Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Intelligent management and control for Wi-Fi small cells

TITULACIÓ: Grau en Enginyeria Telemàtica

AUTOR: David Sesto Castilla

DIRECTOR: Eduard Garcia Villegas

DATA: 9 de setembre del 2016

Títol: Intelligent management and control for Wi-Fi small cells

Autor: David Sesto Castilla

Director: Eduard Garcia Villegas

Data: 9 de setembre del 2016

Resum

Per tal de fer front al creixement exponencial de les transmissions de dades mòbils, des de fa temps es treballa amb el concepte de *small cells*, desplegaments d'alta densitat de petites cel·les per proporcionar una gran capacitat a un número alt d'usuaris.

El projecte SENSEFUL, sota la direcció d'un equip d'investigació de la fundació I2CAT, estudia la utilització d'*small cells* amb tecnologia Wi-Fi, fent un ús compartit dels recursos entre la xarxa d'accés i la xarxa de retorn o *backhaul*. La implantació d'aquest nou paradigma de desplegament de xarxes requereix d'un estudi profund de millores en el rendiment de les xarxes d'accés en termes de mobilitat, a la vegada que s'intenta millorar el comportament de la xarxa *backhaul* mitjançant noves tècniques d'accés al medi compartit.

SENSEFUL ha rebut el finançament de l'*open call* WiSHFUL, iniciada per un col·lectiu d'entitats i universitats, de les quals hem col·laborat, principalment, amb la *Technische Universität Berlin*, degut a la utilització del seu testbed de proves, el TWIST.

Fent ús de tècniques i tecnologies de recent aplicació, com pot ser el paradigma del *Software Defined Networking*, s'aconsegueix el desplegament d'una xarxa intel·ligent que fa una gestió dels recursos de xarxa de forma dinàmica adaptant-se als requeriments del sistema en cada moment. Respecte als dos principals fronts de SENSEFUL, el rendiment de la xarxa de retorn i la mobilitat en xarxa d'accés, les tècniques aplicades són les següents:

Per a la xarxa *backhaul*, s'ha estudiat una proposta de mecanisme d'accés al medi compartit completament innovadora i que encara no compta amb un estàndard per defecte, sinó que són molts els grups d'investigació treballant en aconseguir un sistema funcional per a múltiples escenaris. En aquesta tesi s'estudia el *Hybrid TDMA*, un protocol d'accés al medi ràdio Wi-Fi que utilitza un híbrid de detecció de portadora (*CSMA*) i divisió temporal (*TDMA*) per obtenir els avantatges de tots dos sistemes. Els principals avantatges que *HTDMA* pot aportar són la millor gestió de la qualitat de servei en xarxes sense fil, a la vegada que soluciona problemes endèmics de les xarxes Wi-Fi com poden ser el *node* ocult o el *node exposat*. Per poder treballar en aquesta direcció cal, en primer lloc, un nivell de sincronització elevat entre els dispositius que utilitzaran aquest mecanisme d'accés al medi; és per això pel que els mecanismes de sincronització habituals en xarxes Wi-Fi és un altre dels punts principals dels que es preocupa aquesta tesi.

En quant a la mobilitat en la xarxa d'accés, s'utilitza una nova tècnica, que tot i quedar fóra de l'àmbit d'aquesta tesi, resulta igualment interessant i nova. El *BigAP* consisteix en la unificació de diversos punts d'accés sota un mateix BSSID, proporcionant un *traspàs* imperceptible per als clients realitzant únicament un canvi de canal de transmissió.

Treballant en diferents entorns i escenaris, aquest projecte fa un estudi dels millors mecanismes de sincronització per aquest àmbit. A més a més, es realitza la implantació del sistema *HTDMA* en un petit escenari de proves per tal de començar a analitzar el funcionament d'aquest mecanisme híbrid i el seu rendiment sota diferent condicions, comparant-lo amb el tradicional *CSMA*.

Título: Intelligent management and control for Wi-Fi small cells

Autor: David Sesto Castilla

Director: Eduard Garcia Villegas

Fecha: 9 de septiembre del 2016

Resumen

Con tal de hacer frente al crecimiento exponencial de las transmisiones de datos móviles, desde hace tiempo se trabaja con el concepto de *small cells*, despliegues de alta densidad de pequeñas celdas para proporciona una gran capacidad a un número elevado de usuarios.

El proyecto SENSEFUL, bajo la dirección de un equipo de investigación de la fundación I2CAT, estudia el uso de *small cells* con tecnología Wi-Fi, haciendo un uso compartido de los recursos entre la red de acceso y la red de retorno o *backhaul*. La implantación de este nuevo paradigma de despliegue de redes requiere de un estudio profundo de mejores en el rendimiento de las redes de acceso en términos de movilidad, a la vez que se intenta mejorar el comportamiento de la red *backhaul* mediante nuevas técnicas de acceso al medio compartido.

SENSEFUL ha recibido la financiación de la *open call* WiSHFUL, iniciada por un colectivo de entidades y universidades, de las cuales hemos colaborado, principalmente, con la *Technische Universität Berlin*, debido a la utilización de su testbed de pruebas, el TWIST.

Haciendo uso de técnicas y tecnologías de reciente aplicación, como puede ser el paradigma del *Software Defined Networking*, se despliega una red inteligente que hace una gestión de los recursos de red de forma dinámica adaptándose a los requerimientos del sistema en cada momento. Respecto a los dos principales frentes de SENSEFUL, el rendimiento de la red de retorno y la movilidad en red de acceso, las técnicas aplicadas son las siguientes:

Para la red *backhaul* se ha estudiado una propuesta de mecanismo de acceso al medio compartido completamente innovadora i que todavía no cuenta con un estándar por defecto, sino que son muchos los grupos de investigación trabajando en conseguir un sistema funcional para múltiples escenarios. En esta tesis se estudia el *Hybrid TDMA*, un protocolo de acceso al medio radio Wi-Fi que utiliza un híbrido de detección de portadora (*CSMA*) y división temporal (*TDMA*) para obtener las ventajas de los dos sistemas. Las principales ventajas que *HTDMA* puede aportar son la mejor gestión de la calidad de servicio en redes inalámbricas, a la vez que soluciona problemas endémicos de las redes Wi-Fi como pueden ser el *nodo oculto* o el *nodo expuesto*. Para poder trabajar en esta dirección hace falta, en primer lugar, un nivel de sincronización elevado entre los dispositivos que utilizarán este mecanismo de acceso al medio; es por ello por lo que los mecanismos de sincronización habituales en redes Wi-Fi es otro de los puntos principales de los que se ocupa esta tesis.

En cuanto a la movilidad en la red de acceso, se utiliza una nueva técnica, que pese a quedar fuera del ámbito de esta tesis, resulta igualmente interesante e innovadora. El *BigAP* consiste en la unificación de diversos puntos de acceso bajo un mismo BSSID, proporcionando un *traspaso* imperceptible para los clientes realizando únicamente un cambio en el canal de transmisión.

Trabajando en diferentes entornos y escenarios, este proyecto hace un estudio de los mejores mecanismos de sincronización para este ámbito. Además, se realiza la implantación del sistema *HTDMA* en un pequeño escenario de pruebas con tal de empezar a analizar el funcionamiento de este mecanismo híbrido y su rendimiento bajo diferentes condiciones, comparándolo con el tradicional *CSMA*.

Title: Intelligent management and control for Wi-Fi small cells

Author: David Sesto Castilla

Director: Eduard Garcia Villegas

Date: September 9th 2016

Overview

In order to face the exponential growth of mobile data transmissions, it has been long since the concept of *small cells* is in the table, which provides high density deployments of small cells so as to provide a high capacity to a large number of users.

The SENSEFUL project, being directed by a research team in the I2CAT foundation, studies the use of small cells with Wi-Fi technology, where both the access network and the backhaul share the same radio resource. The deployment of this new paradigm requires a deep study of improvements on the performance of access networks in terms of mobility while, at the same time, trying to improve the behaviour of the backhaul network by means of new techniques to access the shared medium.

SENSEFUL has been granted the funding of the WiSHFUL open call, started up by a collective of entities and universities, of which we have mainly worked with the *Technische Universität Berlin*, due to the use we have made of their testbed, the TWIST.

Using new techniques and technologies, such as the Software Defined Networking paradigm, an intelligent network is deployed, which can manage the network resources dynamically according to the requirements of the system. Regarding both of the fronts of SENSEFUL, the performance in the backhaul network and the mobility in the access network, the techniques that were applied are the following:

For the backhaul network, an innovative proposal of a shared medium access mechanism has been studied. It is not yet standardized, because there are many research teams trying to achieve a functional system that can be applied to multiple scenarios. In this thesis, the *Hybrid TDMA* is studied, a Wi-Fi radio medium access protocol that uses a hybrid of carrier sense (CSMA) and time division (TDMA) in order to benefit from both systems. The main advantages that *HTDMA* brings are a better management of the quality of service in wireless networks, while solving some of the endemic problems of Wi-Fi, such as the *hidden node* or the *exposed node*. So as to work in this direction, first of all, a precise synchronization among the devices that will use this medium access mechanism is required; that is why the usual synchronisation mechanisms in Wi-Fi networks is one of the main topics that this thesis deals with.

Regarding mobility in the access network, a new technique is used, which, despite being out of the scope of this thesis, it is indeed interesting and innovative. The *BigAP* unifies several access points under a shared BSSID, providing a seamless handover for the clients by making only a change on the transmission channel.

Working in different environments and scenarios, this project studies the best synchronisation mechanisms for this field. Moreover, the HTDMA system is installed in a small test scenario so as to begin with the analysis of the operation of this hybrid mechanism and its performance under different conditions, as compared to the legacy CSMA.

Acknowledgement

The original idea of this project has to be granted to Eduard Garcia (director of my thesis at EETAC) and Daniel Camps (member of the I2CAT team). Eduard put me in touch with the research group at I2CAT and guided me through the first steps of the project. After that, he has always been involved in the development of the SENSEFUL project itself, while offering me advice whenever a problem appeared in my part of the project and this thesis.

Moreover, I would like to thank Daniel Camps, Ferran Quer and August Betzler, the members of the I2CAT team with which I have worked, who have been always available to help me with the understanding of the SENSEFUL project and their already-working platform. They were completely essential on my first steps on SENSEFUL, speeding up my incorporation to the project and making sure I had everything I needed to start working.

Finally, I would also like to mention the people at the TKN group in the TUBerlin, who have provided us with some of the tools in which this thesis is based, and have introduced us to the TWIST testbed, keeping periodical teleconferences so as to ensure that the progression of the project kept its expected pace.

Thank you all for this opportunity I was given.

INDEX

FIGURES INDEX

CHAPTER 1. INTRODUCTION

Today's certainty that mobile traffic demand will continue the exponential increase tendency has mobilised lots of researchers and experts in this sector towards developing new techniques or improving some of the existent, in order to be able to cope with the needs of the future 5G networks and users.

Dense small cell deployments seem to be the most effective and reliable way to do so, as they manage to densify access networks by reducing their sizes and having massive deployments. This situation is not something for what current technologies are ready, as efficient mobility and backhauling are just a couple of examples of the pending subjects on this area. Getting into some detail on the backhauling problem (in which this project will be focused), the general idea is that a wide deployment of small cells would make wired connections economically and materially inefficient, as all the access points (APs) would have to be reached using copper or fibre connections. That is why a wireless backhaul network is required for an environment like the one proposed here; however, in such case the backhaul and access networks would compete for the medium resources, so a proper resource management of the wireless channel has to be done.

Software Defined Networking (SDN) and the flexibility it brings to a scenario where centralized management is required, are some of the most novel topics in this area too. SDN tools give the possibility of having intelligent self-managed networks with a centralized controller that can modify the behaviour of the nodes and links according to the feedback collected from the system itself.

This thesis stems from the project SENSEFUL, which has been under development for some time now under the hands of several I2CAT researchers. With the new possibility of working with high-end technologies available through an Open Call to which the project was submitted, we had the opportunity to study and research advanced mechanisms for Wi-Fi networks management. More details on SENSEFUL and the WiSHFUL Open Call are given in the following chapters.

With our own tools, and the experimentation framework provided by the granters of the Open Call, we wanted to develop and test new techniques focused on optimizing wireless network resources in dense scenarios where these resources are shared between the backhaul and access networks. Moreover, regarding the environmental impact of this thesis, it must only be said that the techniques applied in the project are aimed at optimizing network resource management in cellular networks, which can be translated into energetic savings.

As the main objectives for this project, there are several pieces that together converge towards the same idea: building an architecture that allows centralized management using SDN tools and time scheduling with new techniques for a medium access control layer that combines the best aspects of both CSMA and TDMA. The main advantages of this system are that some endemic problems of

CSMA-type networks (for instance Wi-Fi) such as the hidden/exposed node are solved, while enabling the provisioning of QoS.

In order to achieve this general goal, the following objectives were defined:

- Thorough study of synchronisation tools, as they are essential when talking about centralization of TDMA access mechanisms.

- Familiarization with the experimentation environment provided by the WiSHFUL platform, including the preparation of a functional Linux image ready to deploy in TWIST that includes all the necessary modules to make SENSEFUL work in the German testbed.

- Familiarization with the experimental SDN-driven platform provided by I2CAT and over which the SENSEFUL project is built.

- Design and perform experiments focused on the optimization of a hybrid CSMA/TDMA access mechanism.

- Study QoS provisioning over a hybrid CSMA/TDMA access mechanism and compare it to legacy CSMA EDCA quality of service.

1.1. Work in progress and Open Call

For this project I was offered the possibility of working with a research group at I2CAT Foundation [1], helping in the development of a new idea under the name of SENSEFUL, that would be presented to an Open Call for the opportunity of researching and studying advanced network management mechanisms using a testbed based on the latest available Wi-Fi technologies.

1.1.1. Work in progress at I2CAT: the SENSEFUL Project

The research group with which I have collaborated had been working on a new project called SENSEFUL, acronym for "*S*DN driven Joint Access Backhaul coordination for next generation d*ense* Wi-Fi Small Cell networks via WiSH*FUL* APIs". SENSEFUL sets its foundations on the belief that small cells will be the most effective way of dealing with the expected exponential increase in mobile traffic in the following years. Facing the challenges of deploying efficient access networks in the sense of mobility and new backhaul network technologies, this project uses SDN methodologies to research in the area of dense outdoor small cells for future 5G networks.

1.1.2. WiSHFUL Open Call

The WiSHFUL project (Wireless Software and Hardware platforms for Flexible and Unified radio and network controL) [2] is an idea funded by the European Commission's Horizon 2020 Programme that started on January 1st 2015 and aims to ease the research and experimentation of wireless solution developments.

The research group at I2CAT applied to the first WiSHFUL Open Call with SENSEFUL, and the project was granted in March 2016 with the possibility of taking advantage of WiSHFUL tools for its development for 6 months.

Apart from having access to the BigAP [3] architecture and controller developed by a group of researches in the Telecommunication Networks Group (TKN) at the Technische Universität Berlin, we have also been able to work with the TKN WIreless NetworkS Testbed (TWIST) in order to experiment with our project in a wireless indoor environment.

1.2. Document structure

This document is divided into several chapters and sections so as to explain everything that has been done during the project in the clearest and most thoughtful way. First of all, an introduction to the project and its main objectives has been made. In Chapter 2, the reader can find some brief explanations about all the theoretical basis needed to understand the project, going from the present and future of small cells and SDN, to synchronisation tools and the state of the art of TDMA on WLANs. Chapter 3 is devoted to specifying the tools and scenarios that have been used during the development. Chapter 4 talks about the real and final implementation and evaluation of the project. Finally, some conclusions and the direction of future work can be found, alongside with the bibliography consulted for the project and this report, some abbreviations and acronyms and extra information in the annexes.

CHAPTER 2. THEORETICAL BACKGROUND

This second chapter includes the basic background knowledge required to understand the development of the thesis. Starting from a brief review of what Wi-Fi is and some of its endemic problems, it then leads into the basis of small cells and Software Defined Networking. Finally, there is an explanation of some of the most popular protocols and mechanisms for wireless networks' synchronisation, ending with the state of the art of adding TDMA techniques to WLAN environments.

2.1. Wi-Fi

IEEE 802.11 is a set of standards that specify MAC and PHY layers for wireless local area networks in the bands of 900 MHz, 2.4, 3.6, 5 and 60 GHz. Up to now, there are lots of versions and amendments that coexist in the market, as some of them are focused in different areas or specific aspects of WLANs and wireless communications. All this is controlled under the brand of Wi-Fi, the name that the Wi-Fi Alliance gives to any product based on the 802.11 standards.

Getting into some details about its capabilities, the latest IEEE 802.11ac release, which is already being supported by the most recent consumer devices, can reach data rates over the Gigabit per second. That is thanks to the use of MIMO technology, high bandwidth channels (in the 5 GHz band), high order modulations and several spatial streams. In terms of signal range, some hundreds of metres of coverage can be achieved using more reliable modulations, prioritizing bit protection over bit rate.

Now, a brief reminder of some of the basics of Wi-Fi is done, just to understand the importance of keeping working on this field.

2.1.1. Implementation

Each new version of the standard tries, either to focus on its existing problems, or to improve some of the things that seem more important for the users, such as security, data rate, interferences or access range. In fact, the increase of the data rate is one of the things in which the standardization organisation and manufacturers have put more effort. Both the MAC and PHY layers affect the data rate of Wi-Fi. The main considerations in the PHY layer are the available bandwidth, the simultaneous transmissions and the modulation:

In terms of bandwidth, Wi-Fi works in a license-free shared wireless medium, which means that any user can make use of any of the available channels, with the risk of suffering interference that fact brings. In the 2.4 GHz band, the standard defines up to 14 available channels (depending on the regulation domain), with 20 MHz of bandwidth (up to 40 MHz when talking about some of the latest releases, such as IEEE 802.11n) and spaced 5 MHz, as shown in Fig. 2. 1. That means that, for reducing interferences, only 3 or 4 non-overlapping channels can be used in the same range: 1-6-11 (2-7-12 or 3-8-13 alternatively) or 1-5-9-13. However, channel 13 is not allowed in all regulatory domains and its possible removal has been studied.

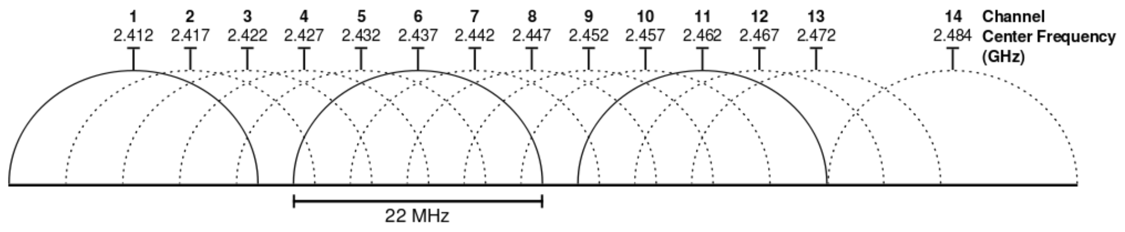

1 2 3 4 5 6 7 8 9 10 11 12 13 14 Channel
2.412 2.417 2.422 2.427 2.432 2.437 2.442 2.447 2.452 2.457 2.462 2.467 2.472 2.484 Center Frequency (GHz)
22 MHz

Fig. 2. 1 Wi-Fi 2.4 GHz band available channels

The 5GHz band has less problems in terms of bandwidth, as there are more available channels, with bandwidths that go from 20 to 160 MHz (as per IEEE 802.11ac).

In the MAC layer, the CSMA/CA protocol is generally applied. Putting it simple, it is a medium access protocol that waits for the channel to be sensed as idle before transmitting. It randomizes transmissions by assigning variable back-off waiting times to each device every time it senses the channel as occupied. Other QoS-oriented MAC techniques such as PCF (Point Coordination Function) have been added to the 802.11 protocol, and despite having a precise management over differentiated services, it has not been successful, and it would not work in multi-hop environments. In PCF, the AP behaves as a point coordinator, polling the connected devices according to their preferences in terms of Quality of Service. In order to ensure that the access to the shared medium is granted to the AP (which will then poll the terminals), it waits for a PIFS (PCF Interframe Space), an interframe space shorter than the DIFS but longer than the SIFS.

In the physical layer, the MIMO (Multiple-Input Multiple-Output) techniques allow transmitting several signals by using multiple antennas in transmission and reception. In the latest amendments, MIMO is also used to enable simultaneous transmission of multiple stations (MU-MIMO)

Last but not least, modulation plays an important role in Wi-Fi too. There have been different modulation techniques that were implemented across the versions and amendments to the standard, here a brief summary of the most relevant ones:

- DSSS: the signal is spread in spectrum and modulated with a pseudo-noise bit sequence (11 bits in the Barker sequence) so that it looks just like noise for all receivers except for the one that must process the signal.

- OFDM: the data is split into multiple subcarriers to transmit it on parallel channels. Each subcarrier is modulated with conventional methods such as QAM or PSK, achieving high data rates at a low symbol rate. Some of the newest amendments currently under development (IEEE 802.11ax) will use OFDM to add a new multi-user layer through OFDMA.

2.1.2. Problems

Wi-Fi networks present some issues that are always identified as the main cause for these networks not to be even more widely spread. Some of these problems are now briefly explained, and this project aims to, as explained in the introduction chapter, to solve or reduce the effects of some of them:

– Performance anomaly: the presence of both fast (usually near to AP) and slow (far to AP) stations in a BSS extremely reduces the overall throughput because slow stations monopolize the medium due to the strict fairness policy CSMA/CA establishes.

– Hidden node: this effect is produced when two nodes that cannot see each other (A and C in Fig. 2. 2) want to transmit to a common neighbour (B), perform the transmission at the same time. In that case, a collision is produced and the information is lost. The RTS/CTS mechanism can be applied to solve it, but it introduces additional overhead.

– Exposed node: this effect is produced when two nodes that can see each other (B and C in Fig. 2. 2) want to transmit to two different stations out of reach of one another (A and D, respectively). Although a simultaneous transmission could be harmlessly carried out, because there would not be a collision, one of the two first stations will sense the medium as occupied and will not transmit.

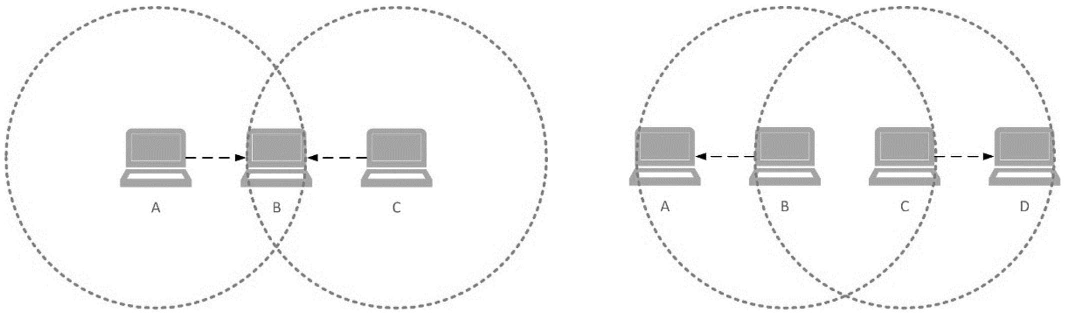


Fig. 2. 2 Hidden (left) and exposed (right) node problematic

2.2. Small cells

The deployment of the so called small cells seem to be the architecture towards which wireless access network providers will have to be moving in order to accommodate the eightfold global mobile data traffic increased expected between 2015 and 2020 (according to the most recent Cisco study [5]).

The Small Cell Forum [6] is the organization in charge of supporting the wide-scale adoption of small cells, by means of defining standards and promoting their use in several scenarios. The standard definition for a small cell says that it "*is an umbrella term for operator-controlled, low-powered radio access nodes, including those that operate in licensed spectrum and unlicensed carrier-grade Wi-Fi*". In fact, the concept of small cell is simply used to talk about all the possible implementations of femtocells, trying to eradicate the idea of femtocells being only used in residential spaces.

There are different types of smalls cells, depending on their size (they usually have a range from tens to hundreds of meters) and their use cases. According to sizes, in increasing order, we can find femtocells, picocells and microcells, which are all based on the "femtocells technology", although the *pico* and *micro* versions may not implement some self-management capabilities. According to the use cases in backhaul scenarios [7], we can also differentiate cells in four groups, the first two of which are capacity-aimed while the others are intended for coverage:

– Targeted capacity hotspot: a hotspot is deployed in order to increase the capacity of a certain network and fill possible spectrum gaps. Some example use cases may be dense urban deployments (e.g.: Times Square, Oxford Street) or Wi-Fi complements for small businesses (McDonalds, Starbucks).

– Non-targeted capacity: enhance user perceived experience related to service availability instead of capacity. Example: macrocells where peripheral coverage requires QoS.

– Indoor coverage: improvement of indoor public spaces coverage in environments of low mobility and occasional peaks. Examples: dense urban indoor venues (stadiums, convention centres, shopping malls…), dense suburban residences, distributed suburban facilities (individual houses, shops or offices with low interferences) and mobile small cells (indoor coverage in public transport).

– Outdoor coverage: provide coverage simultaneously with existing macrocells. Examples: rural area (isolated areas with no macrocell coverage), distributed suburban environment (terrain or building shadowing) and disaster recovery support (provide fast mobilisation of mobile services when natural disasters happen).

Most mobile operators consider backhaul networks the most challenging part of their infrastructures, and the exponential increase of traffic makes them use other techniques such as improving their networks to 4G technology or Wi-Fi offloading (using complementary networks to distribute data alongside with cellular networks). The Small Cell Forum aims to make them understand that backhaul is not the feared barrier to the growth of small cells it is thought to be. In fact, their studies show that the use of small cells can improve capacity by up to 1600x and macro network performance (if placing several small cells inside a macrocell) by 315%. However, there is not a single solution that can fit all scenarios, and depending on the use cases commented previously, the deployment will change.

In fact, LTE and the future 5G networks are designed to work with self-organizing cells, i.e. femtocells. Moreover, innovations in the area of radio interfaces makes it possible for a base station to work with any current or legacy technology, thanks to the implantation of SDR antennas than can tune their functioning via software.

Bringing the concept of small cells closer to the scope of this project, it must be said that operator-run Wi-Fi small cells are not as popular as other licensed spectrum technologies such as GSM, WiMax or LTE. That is because the use of unlicensed spectrum by Wi-Fi devices makes it more complex to make an efficient and effective use of the medium. However, with the introduction of novel techniques such as the ones that will be presented later in this report, small cells can also turn into a reality in WLAN environments.

2.3. Software Defined Networking

Software Defined Networking, known in short as SDN, is one of the greatest advances in network management in the latest years. SDN takes the network management and application services to centralized platforms that, by means of software programming, can change the configuration of the whole infrastructure by provisioning new architectures or reconfiguring the existing ones. This way, topologies for new services or applications can be deployed within minutes, a huge improvement in comparison to the several days that would require in the past, when physical human work was involved in the modification of the architecture. SDN is a key point in the development of future networks because the evolution of the services and their users are asking for things that this technology provides. The change in traffic patterns, the increase in traffic volume and the popularity rise of cloud services are just some of the trends with which Software Defined Networking can help.

SDN achieves this adaptability and dynamicity by decoupling the control plane (where decisions about traffic forwarding are made) and the data plane (elements where the traffic forwarding actually takes place), and abstracting the underlying infrastructure. The organization in charge of promoting SDN through the development of open standards is the ONF (Open Network Foundation). In fact, as they define it, there are some key points in understanding SDN [8]:

– Direct programmability: the decoupling of the control and data planes makes the network directly programmable.

– Agility: the abstraction of control from forwarding makes it possible to adjust traffic dynamically all over the network according to the needs at each moment.

– Central management: the network intelligence is set in logically centralized controllers that keep a global view and understanding of the network.

– Programmatical configuration: there are open-source automated SDN programs that let SDN network managers configure the network dynamically.

– Open standards and vendor neutrality: SDN simplifies everything related to the network operation because the instructions across the network are provided by neutral SDN controllers, not by vendor-specific tools.

2.3.1. SDN Architecture

The SDN architecture, as Fig. 2. 3 shows, relies on three main stacks:

– SDN Applications: programs that communicate with the SDN controller via APIs. These applications send the network requirements to the controller, and can even have a global view of the network so as to make decisions about the situation of the network. Examples of these applications can be network analytics or network management programs that process the data collected from the controller to configure new policies.

– SDN Controller: logical entity that interconnects the decisions made by the applications with the physical devices that build the network. The controller receives information from the network and provides it to the upper applications in the shape of statistics and events. On the meantime, the previously mentioned applications process all that data and send orders to the controller, which “programs” the network devices.
There are several available SDN controllers, some of which are proprietary and some others open source: NOX, POX, Beacon… However, one of the most popular ones is OpenDayLight (ODL), a Java-based Apache platform that uses OSGi framework and bidirectional RESTful APIs as NBI (Northbound Interface) and OpenFlow as SBI (Southbound Interface).

– SDN Network Devices: they control the forwarding and data plane of the network, processing the data path.

The connection between these three stacks is done with two types of interfaces:

– Northbound (NBI): connect the applications to the controller.

– Southbound (SBI): connect the controller and the network devices.

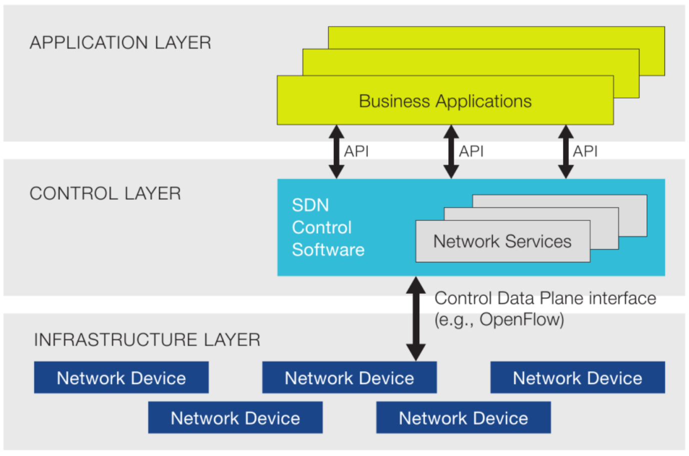


Fig. 2. 3 SDN architecture [9]

2.3.2. OpenFlow: the southbound interface

The OpenFlow protocol [10] is currently one of the key elements of SDN. It is the reference of the open-source southbound interfaces, i.e. a protocol able to program the functioning of network devices according to the rules imposed by the SDN controller.

OpenFlow switches have only the implementation of the data layer, while the control layer is taken care of by an external controller. Each switch has a flow table full of entries that include a set of fields to match (some of them are *ingress port*, *VLAN id*, *MPLS labelling*, *IP and MAC addresses*, *transmission protocol ports*, etc.), a counter updated for every matching packet and an action to perform. When packets arrive that match all the fields on an entry, the corresponding action is done (which could be, for instance, drop the packet, send to a certain interface, modify a field and resend, or just send the packet first to another flow table [Fig. 2. 4]). If a packet that matches no entry is received at a switch, it forwards it to the controller, which then makes the decision of how to deal with it. Generally there are two options, either drop the packet or create a new flow table entry with the actions to perform to packets like that one. Each flow table entry has a couple of timers too, setting the time after which a rule must be removed, under several conditions.

The name given to the southbound interface that connects each OpenFlow switch to a controller is OpenFlow Channel. The messages sent through that interface are formatted according to the standard in the OpenFlow protocol, and a secure transmission based on TLS over TCP through port 6633 is used. The protocol supports three kinds of messages:

– Controller-to-switch: initiated by the controller, its main use is to manage and inspect the state of the switch, asking, for instance, for switching capabilities, configuration parameters, statistics collection, etc.

- Asynchronous: initiated by the switch and destined to the controller, used to notify of packet arrival (*Packet-in*), switch state changes (*Port-status*), flow removal after timeout (*Flow-Removed*) or errors (*Error*).

- Symmetric: unsolicited messages sent in either direction. They can be *Hello* messages, *Echo request/reply* to measure latency/bandwidth/state or *Experimenter* for trying new OpenFlow features.

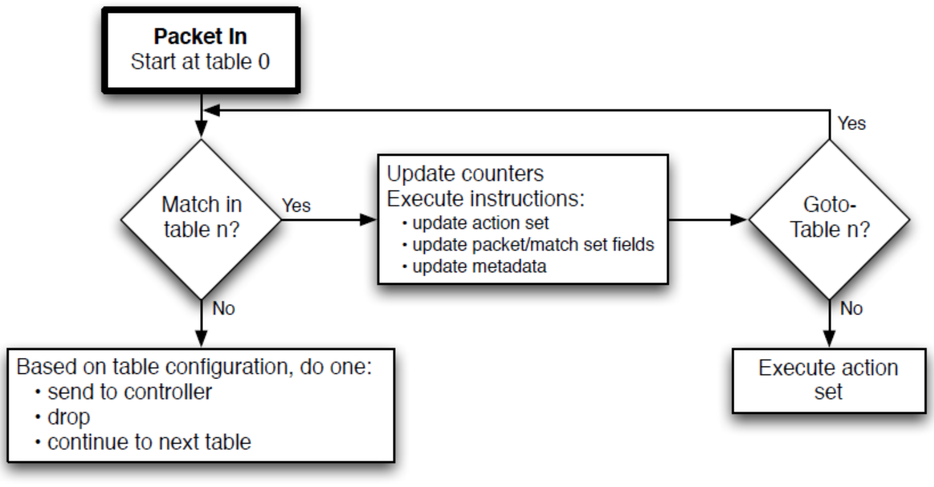


Fig. 2. 4 Flowchart explaining the packet processing when received at an OpenFlow switch

Software Defined Networking seems to be the future of network management. However, an OpenFlow-based implementation of SDN is generally designed for wired networks (for instance Ethernet). When trying to be adapted into wireless networks, some of their typical challenges (such as interference, radio channel variations, etc.) have to be taken into account, and, hence an adaptation is required so as to make the most of all of the potential SDN can bring to a network.

2.4. Synchronisation on wireless networks

As stated in [11], there are many available technologies used to synchronise devices and networks in frequency, phase and/or time; some of which have been studied for the development of this project, and even tested in a real environment. The project in which this thesis is built, relies on a decent synchronisation between devices, as the addition of TDMA techniques in WLAN environments requires that all the devices involved in the procedure share a good timing.

2.4.1. IEEE 1588 Precision Time Protocol (PTP)

PTP provides accurate distribution of time and frequency over a packet network, using several timestamped messages between master and slave devices through which slaves can estimate their offset from the master.

The main messages used by this protocol are: SYNC (message sent periodically from master to slaves, containing its transmission timestamp), FOLLOW_UP (transmitted after every SYNC message, it contains a more precise timestamp which is obtained by measuring the exact time of transmission; only two-step clocks or systems with security protocols need FOLLOW_UP messages, as one-step clocks can perform on-the-fly modification of the timestamp in the SYNC message and therefore the FOLLOW_UP is not needed), DELAY_REQ (slave requests its master to inform the precise time of arrival of the message at the master), DELAY_RESP (response from the master to the previous type of message, containing detailed information of arrival of a DELAY_REQ message; this both last type of messages are used to calculate RTT in the master-slave route).

The use of this four messages provides a collection of 4 timestamps (*t1*, *t2*, *t3*, *t4,* as in Fig. 2. 5), which can then be used to estimate the time offset.

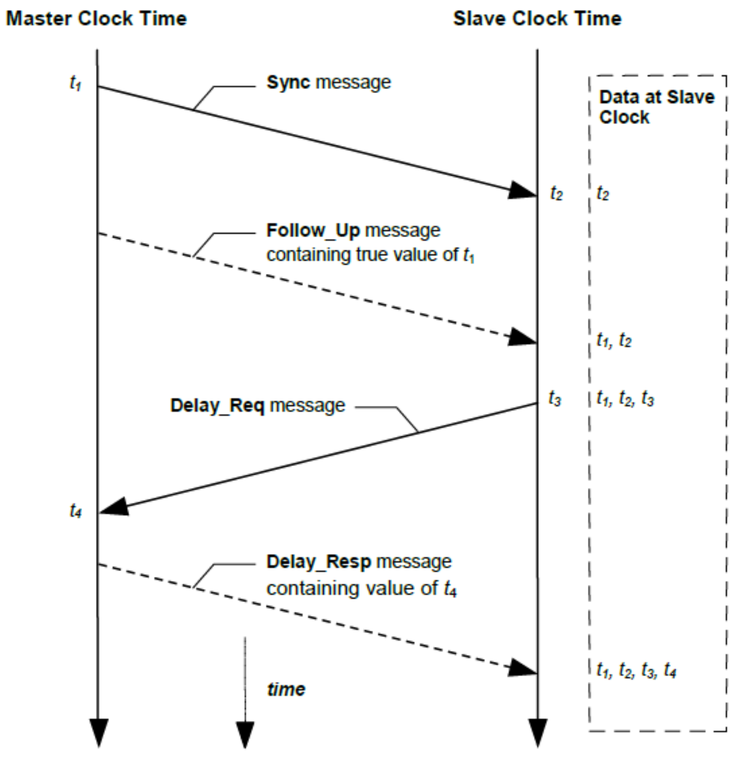


Fig. 2. 5 Functioning of the IEEE 1588 protocol

$$Round\ trip\ delay = (t2 - t1) + (t4 - t3) \tag{2.1}$$

$$One\ way\ delay\ estimate = \frac{round\ trip\ delay}{2} = \frac{(t2-t1)+(t4-t3)}{2} \tag{2.2}$$

$$Slave\ time\ offset\ estimate = t2 - (t1 + one\ way\ delay) = \frac{(t2-t1)-(t4-t3)}{2} \tag{2.3}$$

In (2.2), an estimation of equality in forward and backward one-way delays is assumed.

PTP systems can still suffer from noise that can lead to time and frequency synchronisation error. The sources of noise can be: jitter and wander, timestamping errors at the master or slave, noise at the slave's oscillator, asymmetrical delays, etc. Even though timestamping errors can be solved by using hardware timestamping (which reduces the delays introduced by the software stack that would be in charge of this process) and high precision oscillators, other problems such as PDV (Packet Delay Variation) in the network require additional filtering algorithms in order to achieve an accuracy in the sub-microsecond margin.

So as to reduce the PDV errors, the IEEE 1588-2008 standard defines three ways of providing on-path support:

- Boundary clocks: they recover the clock from the PTP flow and regenerate the flow.

- End-to-end transparent clocks: they forward all messages in the PTP flow transparently while calculating a *residence time* (time the packet has been inside the device) that is later added in a correction field as the packet leaves the device.

- Peer-to-peer transparent clocks: as well as considering *residence time*, delay in network links is also counted by exchanging peer delay messages (Pdelay_req and Pdelay_resp).

The relevance of PTP as a network synchronisation protocol is stated as soon as we learn that IEEE added 1588 into 802.1AS, the Timing and Synchronisation standard that ensures that synchronisation requirements in time-sensitive applications are fulfilled.

Another interesting state-of-the-art technique is the "Fine Time Measurement" (FTM) [12] that is included in IEEE 802.11mc (set of amendments that includes the 802.11-aa, ac, ad, ae and af versions). FTM is nothing but a revision of the already existent Time Measurement (TM) mechanism, but with a finest timestamp resolution and some other minor changes, granting backwards compatibility with TM and IEEE1588. The timestamp resolution with FTM goes down to 0.1ns, and only some minor changes from what is seen in TM are required.

2.4.1.1. *IEEE 802.11v*

The IEEE 802.11 working group added PTP to their standard on the 802.11v amendment ([23]). Under the name of "Timing Measurement", this MAC layer synchronisation option uses the measurement data obtained with higher layer algorithms to synchronise clocks between end devices or synchronize any MAC802.11 system to a common clock.

As can be seen in [24], the Timing Measurement capability uses the same basis as the IEEE 1588 to establish synchronisation between devices. Even the diagrams provided to explain the time references and protocol frame have certain resemblance to what was studied earlier in this section.

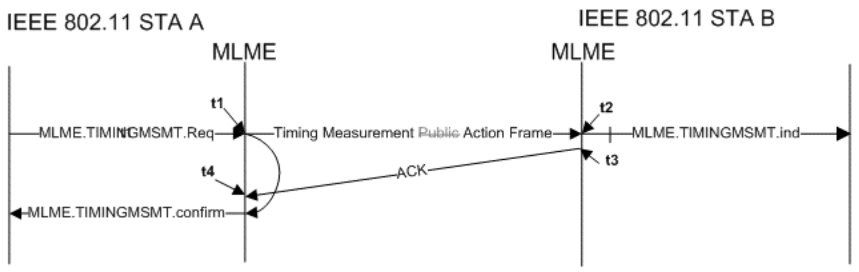


Fig. 2. 6 IEEE 802.11v Timing Measurement protocol functioning ([24])

2.4.2. Network Time Protocol (NTP)

NTP is a client-server protocol typically used to synchronise computer hosts. NTP working strategy is based on a classical clock hierarchy with stratums, where stratum 0 clock is a device such as a GNSS (explained later in section 2.4.3) that provides time information to the stratum below (stratum 1 server). Stratums represent the “topological distance” to the atomic clock where the time reference comes from, and go from 0 to 15 (having a mark of Stratum16 in a server means that it is not synchronised to any time source). The protocol also supports the definition of NTP peers that can automatically define the hierarchy depending on the stratum information that is carried by the protocol.

The client-server mode of this protocol relies on a single request/response message exchange (via UDP on port 123), initiated by the client, which ends up reporting four timestamps (transmission and reception in both the client and the server) that can be used, similarly to what is done in PTP, to calculate the time error from the server. As seen in PTP, the estimation is only accurate for a symmetrical delay in both paths.

While NTP version 3 uses a timestamp of 64 bits (2 fields of 32 bits, which represent the number of seconds since January 1st, 1900; and fractions of a second, respectively), the newest version of the protocol, NTP version 4, works with an extended 128 bit timestamp.
The algorithm NTP uses in order to choose the most reliable server to which synchronise, consists in polling all available servers (the poll period can be modified, and moves between 2^4 and 2^{17} seconds). Then, the last eight samples of each server are ordered in increasing round trip delay (given the fact that the smaller the round trip delay the lower the jitter), the dispersion is calculated (dispersion is defined as the maximum error due to both frequency tolerance and time since the last update) and time and frequency offsets are calculated too. All results from all available servers are compared and the best one is chosen to synchronise the local device clock.

Even though NTP was thought for time synchronisation, it achieves it by aligning clock frequencies between clients and servers. This way, when client clocks are brought to certain margins of *ppb* from the server clock, the poll period increases and viceversa. NTP can achieve better accuracy than that client clocks have inherently, although its performance in the beginning of the process is its main weakness.

2.4.3. Other synchronisation tools

Below, there is a list of alternative synchronisation tools that can be used in a network. Some of them have been considered for our project, although later discarded, while some others are just unfeasible in our case. In Appendix A a more detailed explanation of each of these techniques can be found.

- **DCF77** is a long-wave time signal broadcast from Germany. Its signal is usually used to synchronise clocks, appliances, industrial equipment, etc. It is modulated on amplitude, and coded with pulse-width at 1 bps and is said to have a coverage range over 2000 km from the transmitter, although some factors can make that distance change.
- **SyncE** is built over legacy Ethernet standards and it tries to make it easier to transport clock signals over the physical layer, transmitting the clock directly and continuously over the physical layer on full-duplex scenarios.
- **GNSS** systems used in telecommunication environments (such as GPS, GLONASS or Galileo), were designed to provide accurate time and location references (or only time in bad GNSS-signal conditions at the expense of position being manually supplied) in any point on the planet Earth. The main problem that these systems have is the requirement of *direct sky visibility*, which reduces its functionality in some scenarios.
- **Cellular Network Listen (CNL)** bases its functioning on the same idea with which User Equipment obtain synchronisation, i.e. listening to surrounding cellular base stations of any technology. It is a good idea for the deployment of small cells in highly populated areas, however, the environment has to be studied so as to avoid loops of cells trusting in each other's reference when in fact they all work with CNL.
- **Miniature atomic clocks**, which can meet synchronisation accuracy requirements (from 0.1ppb to 1ppb) without the need of an external reference, are only suitable for Stratum0 clocks or other important base stations, due to their high costs.
- **Hybrid solutions** are also a trend nowadays, so as to get the most out of different available technologies. In Appendix A, some proposals about hybrid technologies can be found.

2.5. TDMA on WLANs

One of the main goals of this project is to study and put into practice the use of TDMA techniques on Wireless LANs, something that is still being under study and development, as there is no standard although it is seen to be of great advantage in many different environments, one of them being the one this project deals with: multi-hop wireless backhaul networks where a guaranteed QoS has to be provided.

2.5.1. TDMA nowadays

Time Division Multiple Access is a channel access method used in networks that share a common transmission medium, and it offers the possibility for different users to work on the same medium and frequency channel, by dividing time into smaller time slots so that each user transmits on its pre-assigned slot without overlapping transmissions.

Nowadays, TDMA is mainly used in 2G cellular networks such as GSM, satellite systems or even the upstream transmissions (end users to operator) over PON networks (downstream PON transmissions are usually broadcast).

The main advantage that TDMA access method provides is that it is quite simple to implement, as it only requires a temporal division and allocation of users. For the duration of the assigned time slot, the radio interface in the user terminal can transmit information, while for the rest of the time it can remain inactive or perform channel measurements in order to achieve a better performance. On the other hand, the TDMA definition itself limits the bandwidth capacity per user of the medium, as only a fraction of the time is available for each user. Moreover, this fraction is usually smaller than $\frac{1}{n}$ (being *n* the number of slots into which the signal was divided), as an optional guard interval is reserved at the beginning and/or end of each time slot so as to avoid overlapping with adjacent slots. These guard periods are especially necessary in scenarios with inaccurate synchronisation, since an error in the determination of the beginning and end of time slots between different machines can lead to an undesired overlapping and its consequent loss of information.

2.5.2. TDMA on wireless multi-hop networks

Up to now, 802.11 networks use random contention algorithms such as CSMA/CA to provide medium access control in wireless environments, and although some other protocols such as PCF have been designed, they have not been a great success. New technologies and adaptations from others are being studied nowadays in order to have a more deterministic access distribution in these scenarios.

The aim of this section of the project is to present the state of the art of TDMA over WLANs, its main advantages and disadvantages, and some existing projects and developments that contribute with good ideas, theories and discoveries to the research in this area.

2.5.2.1. *Introducing TDMA in IEEE 802.11 networks*

The increasing interest of introducing TDMA in IEEE 802.11-based networks has provided some new ideas on how to adapt this protocol for existing wireless technologies, and there are several papers that illustrate the results of the research.

Pseudo-TDMA in MWN [15]

This paper studies the benefits of incorporating a pseudo-TDMA scheme in wireless mesh networks, where CSMA is not enough when it comes to dealing with typical problems such as hidden nodes. Making use of some newer functionalities such as MCCA (Multi-user Controlled Channel Access), which lets nodes in an 802.11 network reserve channel access intervals in advance, or WMP (Wireless MAC Processor), a programmable node architecture, the researches build an unsynchronised multi-hop network that does not require signalling overheads to function.

For the experiments, a new medium access control protocol is built using the WMP hardware functionalities abstraction (i.e. defining an API of hardware actions available at each node) and some simple logic that connects events, actions and conditions. This MAC program can then be transported over the network so that all nodes work with the same set of rules.

Then the MCCA mechanism is added so as to avoid the recurrent “hidden node” problem. It basically tries to distribute channel holding times among groups of nodes in a way that simultaneous transmissions and potential collisions with hidden nodes are avoided. However, it requires a proper node synchronisation and extra signalling over the network, which increases the complexity of the network and reduces its efficiency in terms of bandwidth. That is where the research group goes in and perform some modifications so that the limitations in term of signalling and synchronisation can be avoided.

As defined by its developers, Pseudo-TDMA is a scheme that “*limits the channel access rate at each node, and supports the allocation of different channel holding times to groups of non-interfering stations without explicit negotiation among adjacent nodes*”.

Its operation is in fact quite simple. In short, it performs a first random access CSMA/CA transmission that, if considered successful (i.e. each node received the acknowledgements corresponding to their transmissions) is then repeated every *pseudo-frame* time. An example is given in Fig. 2. 7, where unidirectional flows from each node to its following neighbour are first set randomly. *A* and *C* transmit simultaneously, and only *D* receives the packet with no interference, so then *C* receives *D*’s ACK. Then *C* stops the random access policy and waits for its next *pseudo-slot*, allocated at a distance of a *pseudo-frame* as indicated in the figure. When all the rest of the nodes have their first successful transmission, the final scheduling is repeated periodically.

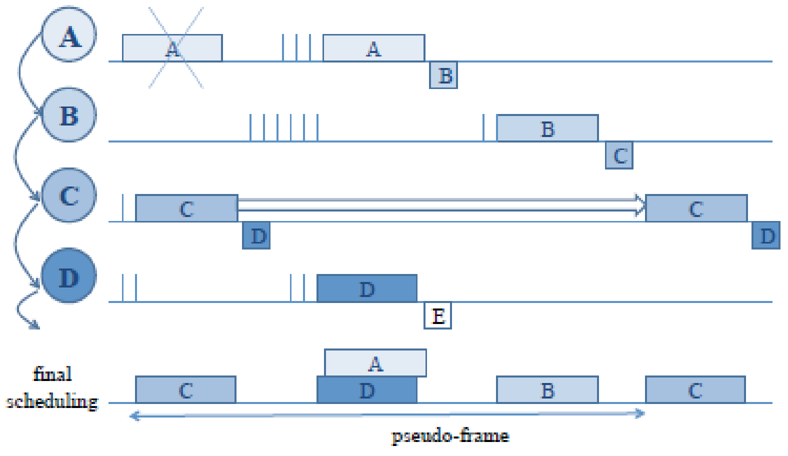


Fig. 2. 7 Example of Pseudo-TDMA access operations in a chain of nodes A-E

The *pseudo-slots* can still suffer collisions before establishing the definitive *pseudo-frame*, which is why the carrier sense is still used before each transmission. That way, each node only needs to know when it has to transmit, not what the neighbours do, and therefore no extra signalling or overheads have to be transmitted over the network.

Several simulations and real experiments show that, although legacy DCF can, under some conditions, achieve a better aggregated throughput, the proposed Pseudo-TDMA scheme obtains a greater fairness rate, which is to be considered one of the most important parameters in this kind of topologies.

Self-organizing TDMA MAC protocol [16]

The SO-TDMA algorithm aims to enable QoS provisioning in delay-sensitive applications by taking the best of both CSMA and TDMA algorithms. Its main idea is the same as the one in [15], with a CSMA start that then converges to TDMA with an adaptive *pseudo-frame*, which changes depending on the channel state obtained by each node of the network. SO-TDMA relies on a distributed mechanism to improve QoS while keeping robustness and scalability untouched.

The group of researches that developed this paper, refer to technologies such as the one in [15] as PTDMA (Pseudo-TDMA), and have studied QoS-aware metrics like the effective capacity to prove that, despite achieving better statistics in saturated traffic scenarios, PTDMA techniques perform poorly under unsaturated conditions. From this, it derives that PTDMA does not adapt properly to changing scenarios.

The SO-TDMA proposal begins just like PTDMA with a first round of CSMA access control that switches to periodic transmissions as soon as all nodes have an assigned *pseudo-slot*. Nevertheless, this protocol then trusts on locally available statistics to adapt each node's transmission length. This way, channel utilization is maximized while providing QoS and improving effective capacity in comparison to pure CSMA or PTDMA environments.

TDMA for today's WLANs [17]

While [15] and [16] propose distributed mechanisms, other systems based on centralized radio resource management have been proposed, as the success of centralized TDMA in cellular networks is a good point to follow. They generally provide a more precise and controlled management, although in these cases, a very good synchronisation between nodes is needed in order to perform a successful scheduling.

In this paper, the OpenTDMF solution is explained, an architecture (Fig. 2. 8) that aims to provide TDMA to commodity WLAN devices, mainly in the RAN access network although its main concepts can be applied to our multi-hop backhaul environment. OpenTDMF's basis is similar to SDN in the sense that it has a decoupled control plane. An OpenTDMF controller manages flow-level access and programmable APs, which can manage packet-level access in both downstream and upstream communications so that all traffic in the network is deterministic.

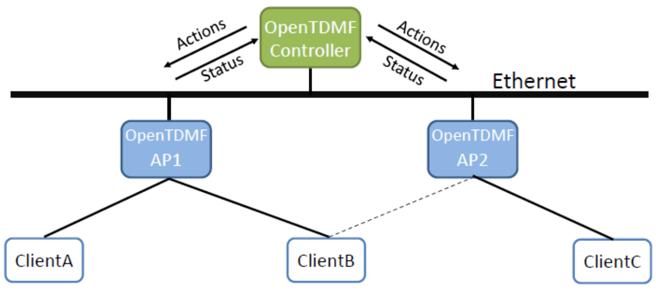


Fig. 2. 8 OpenTDMF architecture

The OpenTDMF controller takes QoS requirements per flow and channels information to calculate actions for each flow on the network, letting the programmable APs coordinate its associated devices without worrying about interference with neighbour APs and clients.

However, as commented before, the first challenge for this idea is the need of a precise synchronisation among APs, because otherwise the scheduling would not be successful. The challenge was solved with the use of existing protocols already explained in this project, such as PTP, and small modifications made to improve its accuracy (reaching the µs level). Secondly, the uplink communications between clients and access points was something to think about too, as it is usually assumed that the clients determine their own medium access, which directly confronts to the concept of centralized time scheduling. This was solved by enabling AP-triggered uplink transmissions.

Through several examples and experiments, the team behind OpenTDMF have proved its benefits in different scenarios with hidden and exposed nodes, QoS requirements, etc.

All in all, the OpenTDMF project has some things in common to the project we are developing, although they are focused in the access network (with APs interconnected via Ethernet) while, in our case, a complex backhaul multi-hop topology is considered.

meSDN: Mobile Extension of SDN [18]

Some of the researchers that wrote this paper want to deal with the problem of providing QoS on WLANs, where the last hop is a shared medium accessed with the CSMA protocol and thus APs cannot coordinate client uplink transmissions.

They propose a framework, *meSDN*, that extends the use of SDN to client devices, using Open vSwitch to manage end-devices' traffic, including additional features such as support for PTDMA and QoS. This extension of the SDN APIs in clients can bring the centralized management of the backhaul network also to the uplink traffic coming from those end devices, achieving a better performance.

To do so, *meSDN* makes use of an architecture composed of three modules that run over the mobile client: the Scheduler (which applies prioritization and limitations in rate to uplink flows), the Flow Manager (an OVS that measures per-flow statistics at the client side) and the Local Controller (which manages both the Scheduler and the Flow Manager, as well as communicating with a Global Controller for the definition of traffic policies and QoS profiles).

So as to solve the synchronisation issues that TDMA obviously brings to a wireless network, they simply define that <<P*TDMA scheduling unit is not per-packet basis but is a larger time window during which a client can transmit and receive multiple packets*>>. That way, a millisecond-level synchronisation is enough instead of the typical microsecond-level required in other scenarios.

HTDMA

HTDMA is the approach to TDMA in WLANs that will be under study in this thesis. It is the technique proposed by the TKN group at the TU Berlin and it is based on the idea of using a hybrid TDMA/CSMA medium access control protocol where outgoing flows are controlled in a first instance by scheduling on the wireless card, while the "real transmission" is later done using the default IEEE802.11 CSMA protocol.

Dividing time in a set of slots on each AP participating in the TDMA system, there is a centralized controller that communicates with the APs using an agent that has to be deployed at each machine. The controller then defines how time slots are distributed according to the information collected by another SDN controller.

The HTDMA implementation that is later reviewed in more detail in chapters 3 and 4, also enables QoS by using 802.11e's EDCA for traffic differentiation.

2.5.2.2. *Synchronisation in centralized scenarios*

As explained in the introduction of this project, and noted also in projects like [17], centralized TDMA scheduling requires a high level of precision between the nodes that will be transmitting in the different time slots. This ought not to be a problem with existent protocols that can provide an accurate synchronisation such as the ones analysed in section 2.4.

The chosen solution in most of the cases (as in ours) is a software-based PTP implementation, which has already been studied to be able to provide µs-level accuracy with some tweaks made over the standard implementation. Taking into account that the transmission time of a WLAN packet can usually move between hundreds and thousands of µs, the accuracy variance is unperceivable.

Aneeq Mahmood et al. have studied the limits of precision with PTP synchronisation in several ways. Paper [28] analyses Software Timestamping in IEEE 802.11 environments in a theoretical way, without further implementations or experiments. It is said that accuracies below 1 µs in terms of jitter can be achieved. Paper [29] analyses Hardware Timestamping over WLANs with real-life experiments, trying to understand the timestamping system instead of treating it as a black-box, and proving that a solution for hardware PTP timestamping over WLAN with a comparable performance to software PTP timestamping over Ethernet is feasible.

2.5.2.3. *Working implementations and off-the-shelf devices*

Although there is still a lot under development in this area, some companies already offer their own proprietary solutions for TDMA in WLANs. The American company Ubiquiti Networks offers 802.11 devices working with their airMax protocol [30], which enables TDMA in order to "*allow each client to send and receive data using pre-designated timeslots [...], eliminate hidden-node collisions and maximize airtime efficiency*". So, basically, they have a real-life implementation of the studies shown before, focused on outdoor scenarios where hidden nodes are a real problem and QoS provisioning is required.

AirMax is a centralized TDMA protocol that keeps track of the activity of all the stations connected to what they call an airMax Sector (translated to standard names, a BSS), preassigning slots dynamically to all active stations. Moreover, intelligent QoS provisioning is also offered, identifying and prioritizing voice and video sessions to provide a lower latency in those delay-aware transmissions.

Despite being a good reference for a real-life implantation of what is being studied in this project, the off-the-shelf devices Ubiquiti Networks offers are a proprietary solution that would work as a black box for us, being then unable to know how they work and make any necessary tweak.

CHAPTER 3. TECHNOLOGIES

3.1. SENSEFUL Project and WiSHFUL platform

As already explained all along this report, the SENSEFUL project uses the resources of the WiSHFUL platform to test new technologies and developments in the field of small cells, aiming to provide a reliable wireless backhaul network that efficiently shares the channel resources with the access network.

WiSHFUL is a platform developed by the Telecommunication Networks Group at the Technische Universität Berlin that aims to provide an easy way to get projects into the experimentation process, by offering a flexible and open source architecture where novel wireless solutions can be tested. WiSHFUL offers a great variety of features that were totally in line of what SENSEFUL was intended to. It provides a node-discovery environment, remote management of the testbed devices (frequency and power management, statistics collection, handover control, etc.) and has both the hardware and software that this project requires, as there are, among other things, an SDN-driven Wi-Fi access network with network-initiated handover support, HTDMA functionalities and a synchronised backhaul.

In this section, a description of all the elements available through the WiSHFUL platform that are being used in the SENSEFUL project is provided. Finally, an example showcase scenario is explained. Although an example topology is shown in Fig. 3. 1, for a better understanding of what is being explained, the reader can go to Appendix B in the annexes, where a figure illustrating the intended scenario can be found.

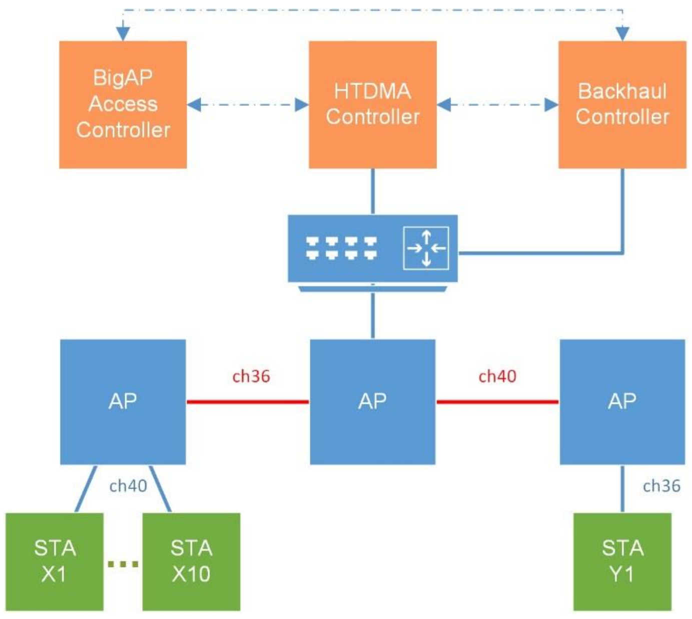


Fig. 3. 1 SENSEFUL showcase scenario

3.1.1. Components

SENSEFUL makes use of the following components:

- Controller nodes: these are the elements in charge of providing the intelligence to the network, as the whole system centralizes its functioning in only some controller devices, following the SDN mentality. SENSEFUL will have three different controller entities, each of which is in charge of a different part of the project. These entities are:

  - BigAP Access Controller: despite being part of SENSEFUL, this branch of the project is out of the scope of this thesis. In short, BigAP [3] is an architecture that provides high network performance and a seamless handover in IEEE 802.11 networks. To do so, it performs below MAC-layer handover making use of DFS (Dynamic Frequency Selection) to force clients to change the AP to which they are connected, while they think they are only changing the operating channel. That is possible thanks to the concept of BigAP: several APs working as a single device.

  - HTDMA Controller: this entity is in charge of assigning the HTDMA scheduling to all backhaul and access nodes.

  - Backhaul Controller: SDN controller working with OpenDayLight that manages the control plane of the backhaul links.

  As expected, all three entities share some interfaces through which relevant information to each of the controllers is transmitted. These interfaces work with queries to the RESTful APIs working on each entity, so that the access (BigAP), backhaul (Backhaul) and medium (HTDMA) controllers can obtain from each other the data they need for decision-making purposes.

- SENSEFUL nodes: wireless elements that transport the client flows across the network to a gateway. In the WiSHFUL platform, these nodes are implemented over Intel NUC devices (their specifications are detailed later on this chapter) , which are equipped with up to three type of wireless interfaces available:

  - Backhaul interface: TDMA-enabled PCI-E interface working with modified ath9k drivers that interconnects backhaul nodes.

  - Access interface: TDMA-enabled PCI-E interface working with modified ath9k drivers that connects the client nodes to the backhaul device.

  - o Scanning interface: USB wireless interface without TDMA capabilities, working with an rt2x00 driver, used to obtain information about the medium status.

- – Client nodes: wireless client stations that generate traffic to be carried by the SENSEFUL network. They share the medium resources and are implemented using TP-Link routers, and their specifications are detailed later on this chapter.

3.1.2. Example scenario

Fig. B. 1 is a good example of a simple scenario where a complete SENSEFUL system is built. In the legend of the image, a brief reference to each type of device can be found.

As the figure illustrates, there are 4 SENSEFUL nodes, only two of them having client nodes attached (10 clients connected to AP2 and 1 client connected to AP3). Furthermore, the stations share the transmission channel with the APs following a schedule that could be similar to the one found in the centre of the image: channel 36 allots more time for backhauling purposes than access, as the channel between AP2 and AP1 is carrying the data of 10 stations, while there is only 1 station (Client STA Y1) working in this channel; on the other hand, channel 40 allocates more time for the 10 active access links than the 2 active backhaul links. This scheduling could change if, for instance, all uplink traffic from stations X is directed towards station Y, then channel 40 would have to increase the time it devotes to the backhaul links so as to improve the performance.

The BigAP controller would be in charge of managing the connection of the client stations to the APs and, for instance, it could initiate the handover of Client STA X10 from AP2 to AP3 just by changing its operating channel from 40 to 36. The HTDMA Controller would be centrally managing the slots at which each backhaul and access node can transmit. Last, the Backhaul Controller would be choosing the most appropriate path for each of the traffic flows coming from the stations.

3.2. Hardware and test scenarios

In order to run all the experiments related to the execution of this project, experiments with different testbeds and scenarios have been run. To begin with, there was a simple home-made topology with some Raspberry Pi devices and laptops. Later, after some successful experiments and with the objective of progressing in the SENSEFUL project, the development progressed into the TWIST testbed. Below, an explanation of the available hardware in each of the cases can be found.

3.2.1. Initial small-scale testbed

The initial testbed was a simple series connection of three devices (three Raspberry Pi at the beginning, but two Raspberry and 1 laptop at the end) connected to a virtual machine running the OpenDayLight SDN controller. In fact, all three devices were connected to a router using their Ethernet ports; the router connected the nodes to the VM via a private virtual network implemented with OpenVPN. The devices were connected among them using their Wi-Fi cards.

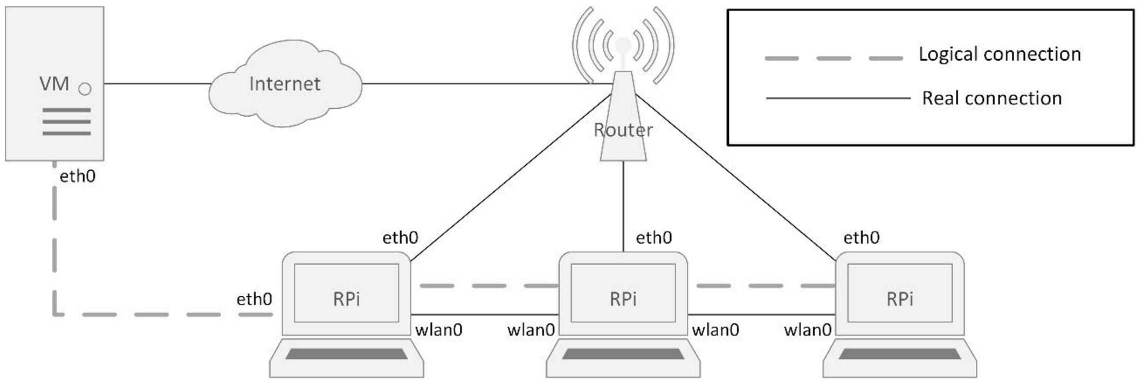


Fig. 3. 2 Initial testbed scenario

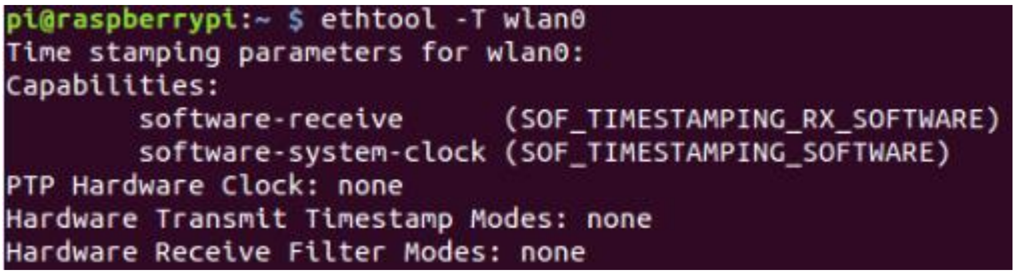
Fig. 3. 2 shows the initial scenario, designed to emulate the network that would later be built in the real testbed. This small-scale testbed was implemented to get in touch with the SDN environment and with all the modules that were already working in the project. In the testbed, each RaspberryPi had a wireless card that interconnected them (emulation wireless backhaul links), and an Ethernet connection used to access the device via SSH through a terminal so as to manage their operation. In the figure, real connections represent the wired or wireless connectivity established among the devices, while the logical connections represent what would in fact be seen by the ODL controller at the VM machine: a controller machine connected to a gateway AP using an Ethernet connection, which then leads to two more APs connected in a wireless environment.

The initial scenario with which the first experiments were performed worked with the following devices:

3.2.1.1. *Raspberry Pi Model B+*

The Raspberry Pi is a cheap single-board computer (SBC) that was originally thought to bring computer learning to schools. Nowadays, it is one of the most demanded SBC devices all over the globe, and is used by students, project developers, etc. The versatility and computing possibilities it offers for such a low budget make it really useful in lots of scenarios.

For this project, some Raspberry Pi Model B+ devices were used to build some simple SENSEFUL topologies and perform some synchronisation experiments and measurements (mainly with NTP, as this model does not support all the timestamping capabilities required for PTP to work [Fig. 3. 3]).



```
pi@raspberrypi:~ $ ethtool -T wlan0
Time stamping parameters for wlan0:
Capabilities:
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
PTP Hardware Clock: none
Hardware Transmit Timestamp Modes: none
Hardware Receive Filter Modes: none
```

Fig. 3. 3 Raspberry Pi Model B+ timestamping options

The main characteristics relevant for this project are:

- CPU: Broadcom BCM2835 with single-core ARMv6 processor
- RAM: 512 MB SDRAM 400 MHz
- Connections:
    - Ethernet port at 10/100 Mbps
    - 4 USB 2.0
    - 40 GPIO pins

The Raspberry Pis ran a Raspbian Wheezy operating system [19], kernel version 3.10 suitable for the specifications of all the software integrated in the SENSEFUL project. Raspbian is nothing but a Debian-based OS specially designed for the Raspberry Pi.

3.2.1.2. *Wi-Fi cards*

Taking into account the needs of the project, we worked with some Wi-Fi cards that would match all the requirements, mainly: having an open Linux driver with *softMAC* support. *SoftMAC*-based devices use the *mac80211* module [20], which takes care of the MAC functionalities.

For the Raspberry Pi, there were some Wi-Fi dongles with Ralink chipset: the Wi-Pi[1], a low-cost wireless interface built for the Raspberry; and Alfa Networks AWUS051NH[2], a bigger wireless USB adapter.

However, in order to take advantage of the feature set offered by the existent WiSHFUL framework (such as the HTDMA MAC), an ath9k-based ([21]) card is needed. Nevertheless, they usually come in the shape of PCI cards, so they would not work with a Raspberry Pi. That is why some work was also performed on an HP laptop with mini-PCI interface and a WLE200NX card with a Qualcomm Atheros AR9280 chipset. Then, the whole Linux image with all the modules for the laptop was built, the new Atheros drivers with some extra modifications made

[1] https://www.element14.com/community/docs/DOC-69361
[2] https://www.alfa.com.tw/products_show.php?pc=67&ps=241

by both the I2CAT team and the TKN compiled (they had added some features to enable their HybridTDMA tools in the card) and finally it was integrated in the initial small-scale test topology.

3.2.2. TWIST testbed

The German TWIST testbed in which the final experiments would be run is part of the iMinds initiative ([22]), a digital research centre that offers a network of experimentation testbeds that can be accessed remotely. There are several testbeds available under this brand, and TWIST was designed to run experiments where an advanced management of Wi-Fi networks is required.

So as to take part in an experiment using an iMinds testbed, reserachers have to be registered in the iMinds Authority[3], an organisation that provides verified identities for accessing their available testbeds. Once the registration process is completed, an SSH key for working with the testbed is granted.

TWIST consists of Intel NUC D54250WYKH Embedded PCs and TP-Link WDR4300 routers distributed all along the TKN building. Their main hardware details are:

- TWIST Routers:
    - Atheros AR9344@560MHz SoC
    - Atheros AR9340 2x2 MIMO for 2.4GHz 802.11b/g/n
    - Atheros AR9580 3x3 MIMO for 5GHz 802.11a/n
    - 3 external detachable dual band antennas (RP-SMA)
    - 128 MiB RAM
    - 8192 KiB Flash
    - Atheros AR8327N Ethernet Switch

- TWIST NUCs
    - Intel Core i5-4250U 4. Gen. 2 * 1,3-2,6GHz
    - 4GB RAM
    - SSD SATA 64GB
    - WLAN a/b/g/n Qualcomm Atheros AR928X (PCI-Express)
    - 10/100/1000 Mbps Ethernet
    - 15W TDP

In order to operate the TWIST testbed remotely, we were provided with a software called jFed, which allowed us to deploy the devices we needed with a default or custom Linux image. See section 3.5.3.4 for more details on the jFed environment.

The first experiments in the testbed were run with the same topology as in Fig. 3. 2, but using their NUCs instead of the three Raspberry Pi. The topology shown in Fig. 3. 4, on the other hand, was intended to perform some synchronisation measurements.

[3] https://authority.ilabt.iminds.be/login.php

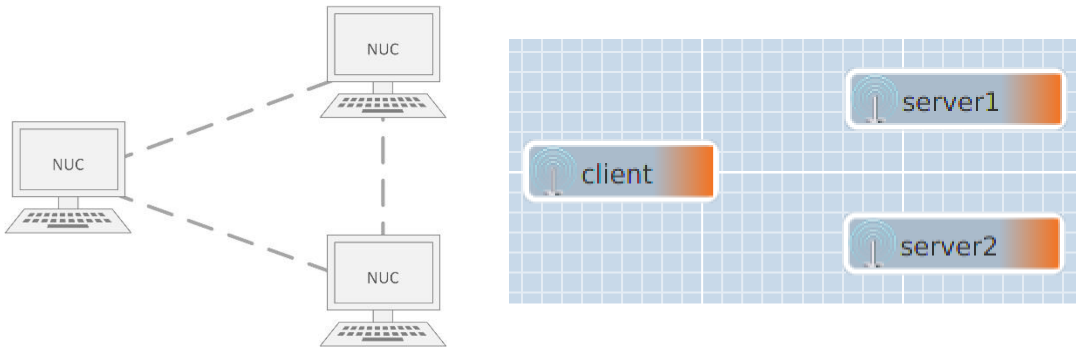


Fig. 3. 4 PTP measurements topology schematics (left) and in jFed view (right)

3.3. Synchronisation tools

This section is intended to work as an explanation of the main synchronisation tools that have been used along the development of the project.

3.3.1. NTP implementation

For the tests with the NTP protocol, the default *ntp* Linux tool was used, which can be obtained through the main repositories. Its functioning is based on modifying the */etc/ntp.conf* file according to the server/client function we want the device to perform:

– **Server:** add the following lines to the configuration file:
  Permit synchronisation to our time source but do not allow the source to query or modify the system:

```
restrict default kod nomodify notrap nopeer noquery
restrict -6 default kod nomodify notrap nopeer noquery #-6 adds IPv6 support
```

  These options were included: *kod* (Kiss-of-Death, send a packet to reduce undesired queries), *nomodify* (forbids changes to the configuration), *notrap* (prevents *ntpdc* control message traps), *nopeer* (no peer association can be established), *noquery* (*ntpq* and *ntpdc* queries will not be answered, although time queries will).
  Allow only machines in your network (192.168.1.0, in our case) to synchronize with the server, and grant all permissions to localhost:

```
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
restrict 127.0.0.1
```

  In case the server is disconnected from its time reference, use the local clock as backup:

```
server  127.127.1.0 # local clock
fudge   127.127.1.0 stratum 10
```

Add log files to see the evolution of the synchronisation:

```
driftfile /var/lib/ntp/ntp.drift
logfile /var/log/ntp.log
```

Finally, after restarting the *ntpd* daemon, we will have a working NTP server.

- **Client:** add the following lines to the configuration file:
  Specify multiple servers in case the first ones can't be reached. Also add the IP address of the NTP server configured previously, with the *prefer* option to set it as the desired source. The *iburst* option sends 8 packets after each poll instead of 1, so that the initial synchronisation can be made faster.

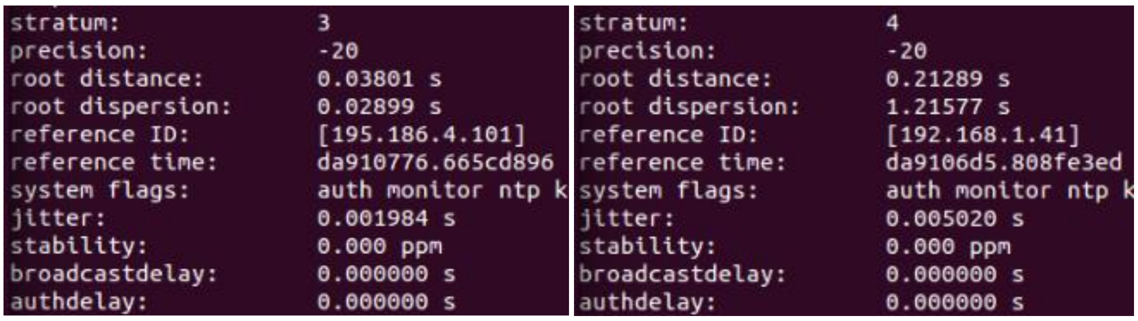
  ```
  server 0.rhel.pool.ntp.org iburst
  server 1.rhel.pool.ntp.org iburst
  server 2.rhel.pool.ntp.org iburst
  server 3.rhel.pool.ntp.org iburst
  server 19.168.1.1 prefer
  ```

  Again, restarting the daemon will get us the NTP client working.

In order to force the synchronisation to the server, you may have to type the following command:

```
ntpdate –u 19.168.1.1
```

Finally, using the commands *ntpq -p* and *ntpdc –c sysinfo* we can obtain information about the synchronisation.



```
stratum:              3
precision:            -20
root distance:        0.03801 s
root dispersion:      0.02899 s
reference ID:         [195.186.4.101]
reference time:       da910776.665cd896
system flags:         auth monitor ntp k
jitter:               0.001984 s
stability:            0.000 ppm
broadcastdelay:       0.000000 s
authdelay:            0.000000 s
```

```
stratum:              4
precision:            -20
root distance:        0.21289 s
root dispersion:      1.21577 s
reference ID:         [192.168.1.41]
reference time:       da9106d5.808fe3ed
system flags:         auth monitor ntp k
jitter:               0.005020 s
stability:            0.000 ppm
broadcastdelay:       0.000000 s
authdelay:            0.000000 s
```

Fig. 3. 5 NTP server (left) and client (right)

For example, in the initial test scenario, configuring a server and a client in NTP, then running the *ntpdc –c sysinfo* command, resulted in Fig. 3. 5. As you can see, the server is in this case a stratum 3 device, while the client (which obviously must be below) is in stratum 4.

3.3.2. PTP implementation

As for the project in which I have taken part, there are different implementation options for the PTP standard.

3.3.2.1. *IEEE 802.11v*

Despite being a standard supported by some consumer devices, we found the IEEE 802.11v (explained in section 2.4.1.1) to be something too specific for the proportions of our project, so we decided to use a different approach in this area.

3.3.2.2. *Higher layer implementations and timestamping: the Linux PTP Project*

As an alternative to low layer PTP implementations, several options for running this protocol in higher layers have appeared. All of them are based on the concept of a “packet timestamp”, i.e. a time reference added to a packet when it is transmitted or received by a device. Although, ideally, we would want the timestamp to be set in the same instant as when the packet is received or sent, this cannot be achieved due to the OSI Reference Model, as the timestamping and packet transmission/reception work in different layers and there is some delay when the packet travels through them. In order to minimize the error between the timestamping and packet physical delivery, there are two different timestamping models:

- Software timestamping: it is implemented either in the Application or Operating System layers and the time reference is obtained from the system clock, so a relatively large error is assumed.

- Hardware timestamping: it is implemented either in the Physical or MAC layers and the time reference is obtained directly from the PHC (PTP Hardware Clock) included in the NIC, so that the error is minimized. This alternative is slightly more complicated as the one based on software, because it requires the synchronisation between two clocks in the same device: the system clock and the PHC.

In Fig. 3. 6 we can graphically see the difference between this two models. Although hardware timestamping is inherently a better option due to its reduced timing error, it must be noted that it requires the functioning of more pieces of software as will later be explain. Moreover, not all connection devices (either Ethernet or Wireless cards) offer software and/or hardware timestamping options, so it is something that needs to be taken into account.

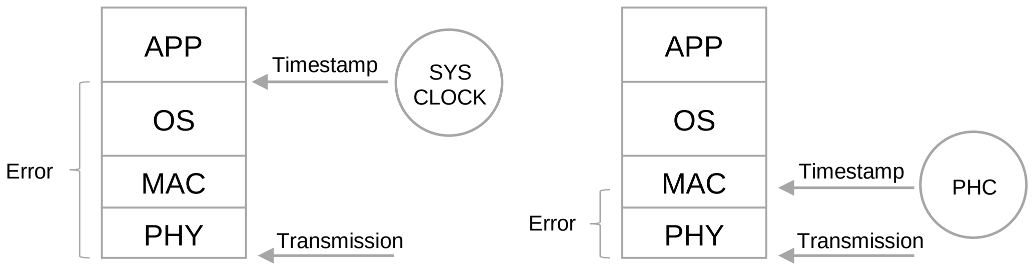
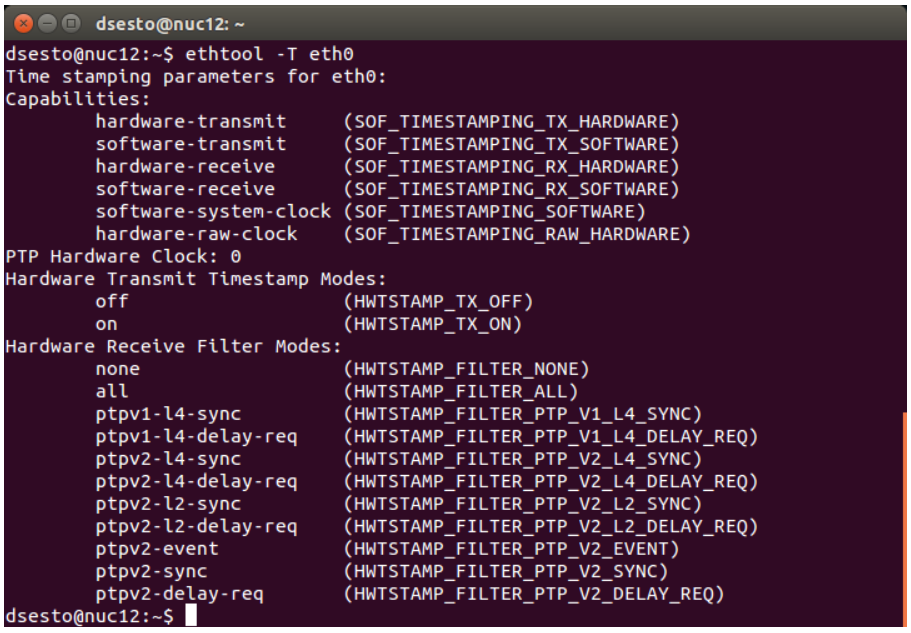


Fig. 3. 6 Software (left) and Hardware (right) timestamping layers comparison

All these higher layer PTP implementations work on user space, provided that the Linux kernel supports some specific features, such as:

– Packet socket timestamping options, generally referred to as SO_TIMESTAMPING. Linux applications as *ethtool* (available through Linux usual repositories) can help to know whether the interfaces we want to use have software and/or hardware timestamping capabilities. For instance, the Intel NUCs being used in the TWIST testbed (Fig. 3. 7) do implement support for both hardware and software timestamping in transmission and reception mode.



```
dsesto@nuc12: ~
dsesto@nuc12:~$ ethtool -T eth0
Time stamping parameters for eth0:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        all                   (HWTSTAMP_FILTER_ALL)
        ptpv1-l4-sync         (HWTSTAMP_FILTER_PTP_V1_L4_SYNC)
        ptpv1-l4-delay-req    (HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ)
        ptpv2-l4-sync         (HWTSTAMP_FILTER_PTP_V2_L4_SYNC)
        ptpv2-l4-delay-req    (HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ)
        ptpv2-l2-sync         (HWTSTAMP_FILTER_PTP_V2_L2_SYNC)
        ptpv2-l2-delay-req    (HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ)
        ptpv2-event           (HWTSTAMP_FILTER_PTP_V2_EVENT)
        ptpv2-sync            (HWTSTAMP_FILTER_PTP_V2_SYNC)
        ptpv2-delay-req       (HWTSTAMP_FILTER_PTP_V2_DELAY_REQ)
dsesto@nuc12:~$
```

Fig. 3. 7 TWIST NUCs timestamping capabilities

– The PHC subsystem should be accessible using system calls such as *clock_gettime* or *clock_settime*.

– The drivers must support timestamping. In the TWIST testbed, NUCs are equipped with Ethernet cards compatible with the e1000e drivers, which are timestamping-enabled.

As the community sees it, Richard Cochran's Linux PTP Project ([26]) seems to be the reference system for PTP over Linux. Many important companies such as Fujitsu, Intel, RedHat ([27]) or SUSE, among others, have shown interest in this project and actively participated in its development. It is based on three main applications that work together to provide a whole PTP-enabled system:

ptp4l

The ptp4l application offers an implementation of the PTP boundary and ordinary clocks. It works with IEEE 802.3, UDP IPv4 or UDP IPv6 transport protocols, and lets the user choose between hardware or software timestamping. When working in software mode it synchronises directly the system clock to the master, while in the hardware mode, it synchronises the PHC to the master clock and another tool will be needed to synchronize the system clock.

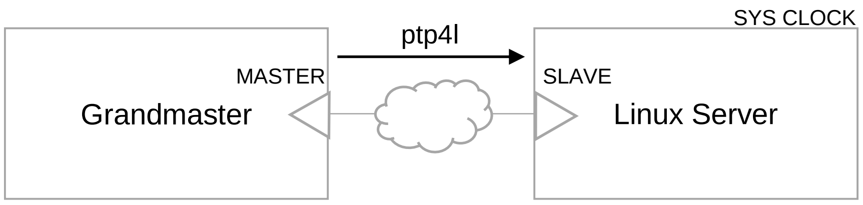
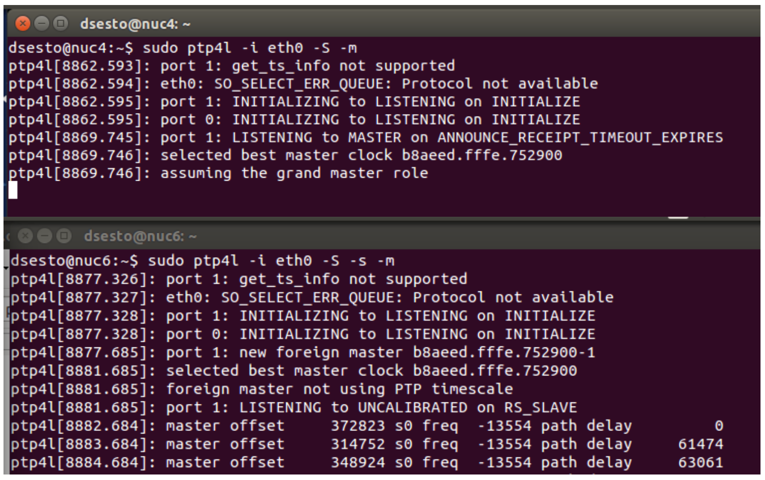


Fig. 3. 8 Software timestamping synchronisation chain

A use example for this program can be found below, where *-i* identifies the interface to be used, *-S* specifies Software Timestamping (*-H* alternatively), *-s* sets the *slave only* mode, where the device cannot work as master clock and *-m* enables the verbose mode so that all messages output are shown on screen. Other options such as *-E* establish the E2E mechanism (*-P* for P2P instead) and *-p* sets the PHC device to be synchronised.

```
# ptp4l -i eth0 -S -s –m
```



```
dsesto@nuc4:~$ sudo ptp4l -i eth0 -S -m
ptp4l[8862.593]: port 1: get_ts_info not supported
ptp4l[8862.594]: eth0: SO_SELECT_ERR_QUEUE: Protocol not available
ptp4l[8862.595]: port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l[8862.595]: port 0: INITIALIZING to LISTENING on INITIALIZE
ptp4l[8869.745]: port 1: LISTENING to MASTER on ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES
ptp4l[8869.746]: selected best master clock b8aeed.fffe.752900
ptp4l[8869.746]: assuming the grand master role
```

```
dsesto@nuc6:~$ sudo ptp4l -i eth0 -S -s -m
ptp4l[8877.326]: port 1: get_ts_info not supported
ptp4l[8877.327]: eth0: SO_SELECT_ERR_QUEUE: Protocol not available
ptp4l[8877.328]: port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l[8877.328]: port 0: INITIALIZING to LISTENING on INITIALIZE
ptp4l[8877.685]: port 1: new foreign master b8aeed.fffe.752900-1
ptp4l[8881.685]: selected best master clock b8aeed.fffe.752900
ptp4l[8881.685]: foreign master not using PTP timescale
ptp4l[8881.685]: port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[8882.684]: master offset     372823 s0 freq  -13554 path delay        0
ptp4l[8883.684]: master offset     314752 s0 freq  -13554 path delay    61474
ptp4l[8884.684]: master offset     348924 s0 freq  -13554 path delay    63061
```

Fig. 3. 9 ptp4l master (top) and slave (bottom) devices

phc2sys

The phc2sys application is only needed when hardware timestamping is chosen, and it takes care of synchronising the system clock to the PHC on the NIC, which is synchronised to a master clock by ptp4l.

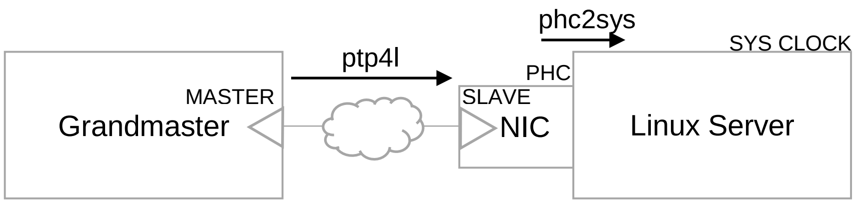


Fig. 3. 10 Hardware timestamping synchronisation chain

A use example for this program can be the one presented below, where *-s* specifies the network's interface PHC, *-c* the slave clock (typically the system clock) and *-w* tells the software to wait for ptp4l synchronisation.

```
# phc2sys -s eth0 -c CLOCK_REALTIME –w
```

pmc

The PTP Management Client (pmc) application sends PTP management messages (as specified in IEEE 1588) to connected PTP nodes. Although GET, SET and CMD actions are available through the definition of the protocol, not all messages are supported by most PTP devices.

LinuxPTP precision

Due to the increasing use of the Linux PTP Project in environments where synchronisation is required, researches have started studying the precision limits to which LinuxPTP can be brought. With complex mathematical models and experiments as the ones used in [28] and [29], accuracies below 1 μs can be achieved, although statistical models and further tools have to be applied in most of the cases in order to reduce the errors produced by the path delay asymmetries. This topic has already been treated in section 2.5.2.1 of this project.

3.4. Hybrid TDMA

The reason why synchronisation is an important part of this project is that it is essential for developing a TDMA-based access system for Wi-Fi networks. TDMA relies on a perfect synchronisation among all the devices in the network, so as to have a good definition of the slots in which the shared resources will be divided.

One of the main focuses of SENSEFUL was to test TDMA techniques in Wi-Fi environments. To do so, we have worked with some pieces of software provided by the TKN group. Their approach to time scheduling in a WLAN is called *Hybrid*

TDMA (HTDMA), a technique that makes use of both TDMA and CSMA for the medium access control in a network. It basically enables the possibility of dividing the transmission queue of an ath9k wireless card into *N* slots of duration *t* microseconds. Then, a set *n* of the slots are configured to allow a given type of traffic and hence, that traffic will only be transmitted for $\frac{n}{N}t$ each cycle, as Fig. 3. 11 shows.

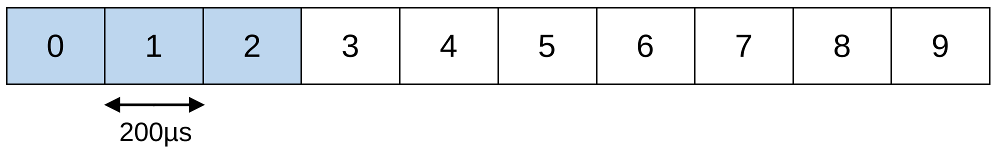


Fig. 3. 11 Hybrid TDMA example with N=10, n=3 and t=200µs

The relevant configuration parameters of the HTDMA implementation are:

– Number of slots per cycle (*N*), i.e. the amount of divisions each cycle has.

– Duration of each slot (*t*), in µs; being *N*t* the total duration of a cycle.

– Traffic allowance in each slot:

  ○ *Allow all:* any kind of traffic can be transmitted, without restriction.

  ○ *Disable all:* no traffic will be transmitted in this slot. This feature is useful when thinking in a TDMA environment, where some nodes remain silent while others are allowed to transmit.

  ○ *Filter by MAC address and ToS:* only the traffic directed to a device with a certain MAC address and with a specific access category is allowed (following the priority levels in Table 3. 1). This feature is useful for the implementation of QoS, as some slots can be reserved for high-priority traffic, for example.

	802.1D		**802.11e**	
Priority	*Type of Service (ToS)*	*Traffic Type*	*Access Category (AC)*	*Designation*
Lowest	1	Background (BK)	AC_BK	Background
	2	Spare	AC_BK	Background
	0	Best Effort (BE)	AC_BE	Best Effort
	3	Excellent Effort (EE)	AC_BE	Best Effort
	4	Controlled Load (CL)	AC_VI	Video
	5	Video (VI)	AC_VI	Video
	6	Voice (VO)	AC_VO	Voice
Highest	7	Network Control (NC)	AC_VO	Voice

Table 3. 1 Priority levels and Access Categories

Regarding the required pieces of software to make this work, first of all, there is a modified version of the *ath9k* driver that enables all the required features for slot division and assignment in the wireless card. Moreover, a *daemon* must be installed, which takes care of the communication with the driver and offers an API that can be accessed remotely (for example by the HTDMA centralized controller) for its configuration. Finally, if we just want to run the protocol locally, we only need a python script with a couple of classes and a *main* function that takes care of creating, updating or deleting the slot system with which the wireless card will be working.

3.5. Other tools

This section aims to explain other important tools that have been used or taken into account during the development of the project, from programming languages to SDN software or deployment tools.

3.5.1. Programming languages

These are the main programming languages that have been used during the development of the project: Bash, Java, C and Python.

3.5.1.1. *Bash*

Command language used by the majority of Linux distributions, with which the user can send actions to the computer using commands. Bash can read commands from a file, called script, and can make use of other software such as *grep*, *aptitude*, or any other Linux default command.

Bash scripts have been really useful when deploying all modules of the project, as a single script was in charge of installing *git* and its keys, downloading chef and the recipes for all the modules, and installing them. Moreover, there was another bash script with which the SENSEFUL scenario could be started up.

3.5.1.2. *Java*

Object-oriented, concurrent, class-based programming language designed with the WORA (Write Once, Run Anywhere) mentality so that a compiled Java program can be run on any platform with Java support without requiring any further recompilation.

The OpenDayLight SDN controller used in SENSEFUL is based on Java, and all the modules implemented in the SESAME project are written in this programming language.

3.5.1.3. *C*

General-purpose programming language that provides constructs equivalent to a series of machine language instructions. Its run-time requirements are minimal due to its low-level programming paradigm, although this fact also makes its use more complex for some applications.

C offers a really simple access to network sockets, reason why this language was chosen to write the scripts in Appendix C, used to measure synchronisation in a PTP environment.

3.5.1.4. *Python*

General-purpose programming language with a syntax focused on readability that can achieve the same functionalities as other languages with fewer lines of code. It works with some useful and popular libraries such as ZeroMQ, an asynchronous messaging library that uses N-to-N sockets to carry messages through different transport mechanisms.

Python and ZMQ are used by the HTDMA implementation written by the TKN group.

3.5.2. Version control

Version control tools are essential in development teams, as they ease the task of tracking all the parallel progresses while having a permanently available version of the project online. One of the most popular tools, is *git*.

3.5.2.1. *Git*

Version control system used by most of the software development teams in the industry. It works under a distributed topology, which means that in a shared repository, every terminal in which the repository was cloned, is a repository itself, with full tracking capabilities. That way, the server side of the repository is nothing but a cloud version of the directories being tracked. With git, everyone works with their local version of the repository, which then has to be uploaded for example to a GitLab server using commits. In this procedure, merges between users may be needed, when the same file has been modified in two independent local repositories.
Git has been really useful for this project in the sense that all the SENSEFUL scripts required to start it were already available in an I2CAT GitLab repository. For this thesis, some personal branches were also created so as not to interfere with the work under development while performing other experiments. Git also made everything easier when having to deploy the modules to a new device, as everything was available online.

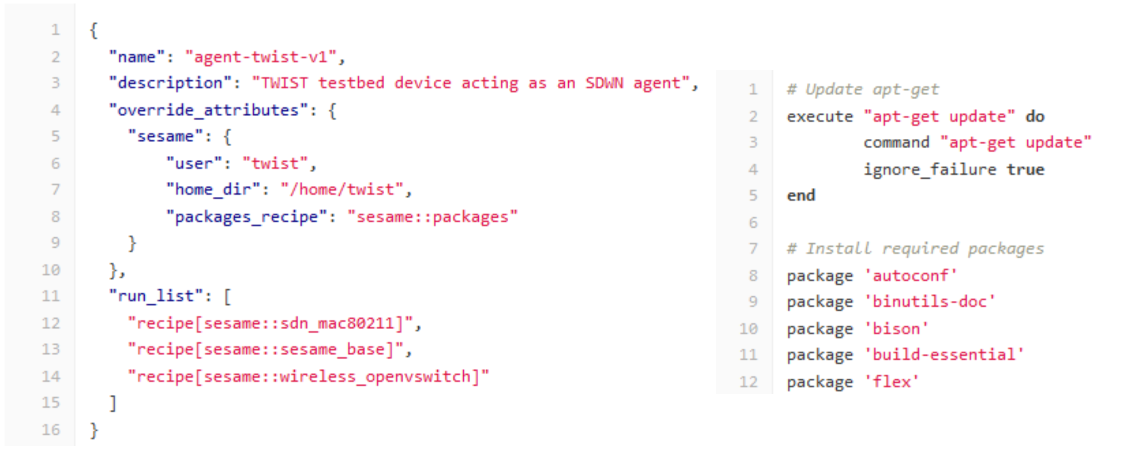
3.5.3. Deployment tools

Under the title of deployment tools, all the software that has been used in the different stages of the deployment of our system is included. *Chef* made it easier to get all the SENSEFUL modules working just by running some scripts. *Ansible* was used to prepare some Ubuntu images for the TWIST testbed. Finally, *jFed* did the “physical” deployment of the images on the testbed.

3.5.3.1. *Chef*

Configuration management tool that is really popular when dealing with multiple identical deployments of a system in several machines. It is also integrated in some of the most popular cloud-based platforms, like Amazon EC2, Google Cloud Platform, OpenStack or Microsoft Azure. With chef, you define a set of *recipes*, *cookbooks* and *roles* that must be executed by the Chef Client (Ruby-based) in order to set a machine to the working state we want it to be. In *recipes* the user defines which packages (for instance *git*, *ssh* or *ethtool*) must be installed and how to configure them, which files must be written or which services running. Then, the developer can group several related *recipes* under a *cookbook*; and define other specifications in a *role*.

For the deployment of the SENSEFUL modules, a bash script that launched a *chef-client* instance with a role that installed all the required packages and modules was used.



```json
{
  "name": "agent-twist-v1",
  "description": "TWIST testbed device acting as an SDWN agent",
  "override_attributes": {
    "sesame": {
        "user": "twist",
        "home_dir": "/home/twist",
        "packages_recipe": "sesame::packages"
    }
  },
  "run_list": [
    "recipe[sesame::sdn_mac80211]",
    "recipe[sesame::sesame_base]",
    "recipe[sesame::wireless_openvswitch]"
  ]
}
```

```ruby
# Update apt-get
execute "apt-get update" do
        command "apt-get update"
        ignore_failure true
end

# Install required packages
package 'autoconf'
package 'binutils-doc'
package 'bison'
package 'build-essential'
package 'flex'
```

Fig. 3. 12 Examples of chef role (left) and recipe (right)

3.5.3.2. *Ansible*

Configuration management tool used by the TKN to build the custom Ubuntu images ready to be deployed in the TWIST NUC devices. With ansible you basically establish a set of roles (a similar meaning to the term used in *chef*, as it represents a set of actions to be executed) that, by means of calls to ansible tasks prepare the configuration the user has programmed. The roles are defined inside ansible *playbooks*, **.yaml* format files that include all the parameters and tasks to be done.

Using a playbook provided by the TKN team (which included a debootstrap task pointing to an Ubuntu Precise image) we could have a "clean" Ubuntu12.02 image with a 3.2 kernel version. Over that image, we would have to install all the required packages and modules using other tools such as *chroot* and *chef*.

3.5.3.3. *debootstrap*

Tool used to install a Debian-based operating system inside a subdirectory of the running OS. The Debian image can be automatically downloaded from any official repository.

Using debootstrap as a task inside the ansible playbook, we obtained the Ubuntu base image we needed for the TWIST testbed.

3.5.3.4. *jFed*

jFed [31] is a software developed by the iMinds organisation (Flemish research institute in charge of promoting ICT companies, authorities and other organisations), the Ghent University (a telecommunications team in Belgium) and Fed4Fire (the Federation for Future Internet Research and Experimentation). It is a Java-based framework designed for testbed federation that, in our case, we use to access the TWIST testbed and deploy our network on their devices. It is really easy to use, and has a friendly interface when talking about the deployment of the nodes. However, it can be a bit rough to work with when problems appear, as there is no feedback when nodes cannot be deployed, so there is no way to know what the problem was.

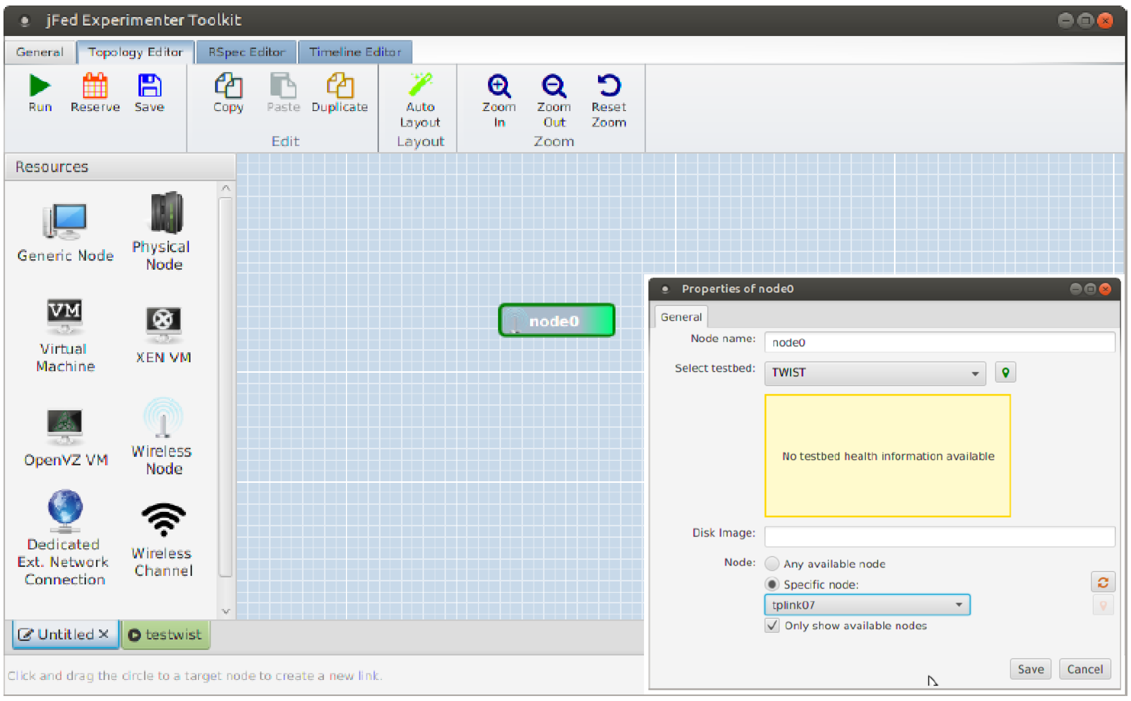
Before getting in jFed, the user must download the certificate granted by iMinds and browse it in the launcher. If the certificate is valid, jFed will show the user, the authority that granted it and its expiration date.

Here there is a brief explanation on how to deploy a node:

1. Click and drag a "Wireless node" to the workspace.

2. Choose the testbed to be used (TWIST in our case), the node to be deployed ("NUCx", were *x* is a figure) and copy the link to an *http* server (note that services offering storage with access via *https* protocol such as Dropbox will not work in jFed, so an *http* storage server has to be used) containing the desired image compressed in a **.tar.gz* file.

3. Run the experiment, and once the node lights in green (red means the deployment was unsuccessful), you will be able to access the node via SSH.



Node name: node0

Select testbed: TWIST

No testbed health information available

Disk Image:

Node: ○ Any available node [● Specific node:] tplink07

[✓] Only show available nodes

Click and drag the circle to a target node to create a new link.

Fig. 3. 13 jFed main screen and node configuration

3.5.4. Drivers

Below, some of the drivers with which this thesis has interacted along the project.

3.5.4.1. *mac80211*

Framework used to modify drivers in SoftMAC wireless devices. SoftMAC devices, in comparison to FullMAC, enable a bigger control over the hardware, as the 802.11 frames can be managed via software. *mac80211* makes use of *cfg80211* for networking registration and configuration, while it uses *nl80211* and other wireless extensions to configure the device.

In the project, a modified version of *mac80211* has been used, which basically collects statistics of the wireless medium, enables QoS and changes VLAN LLID and PLID tags. All these modifications were developed in order to integrate the system with the SDN platform.

3.5.4.2. *ath9k*

The wireless Atheros cards we have used in the project (Compex WLE200NX[4]) ran with the ath9k driver using the Qualcomm Atheros AR9280 chipset. In fact, it

[4] http://www.compex.com.sg/product/wle200nx/

is a modified version that collects information from the debugFS (RAM-based file system designed for debugging) about the hardware outbound queues.

Later, some extra modifications were added to the driver, so as to enable some TDMA capabilities required to make the HybridTDMA module work.

3.5.5. SDN software

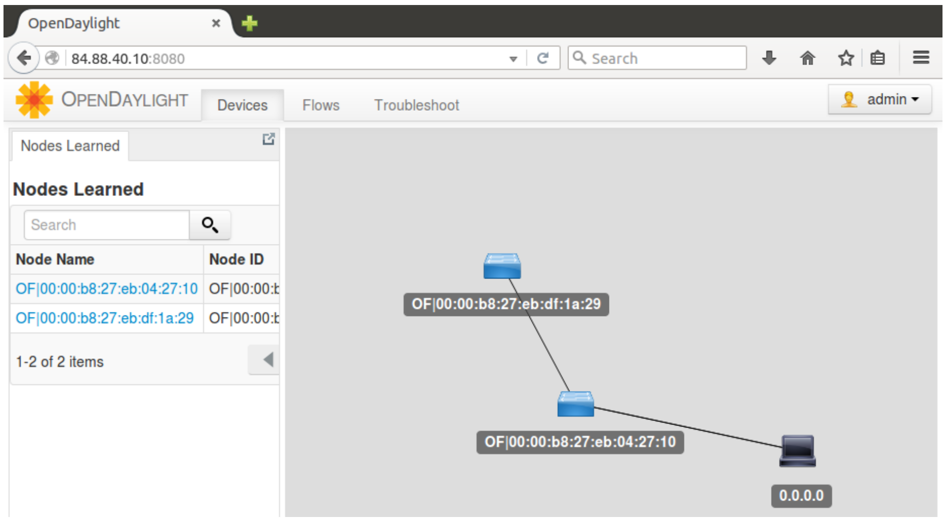
Here the main SDN software used in the SENSEFUL project is described.

3.5.5.1. *OpenDayLight*

OpenFlow controller written in Java that provides the intelligence to the control plane of an SDN network. It can be accessed through several interfaces, the most common of them being a RESTful API and a graphical user interface accessible through a web browser.

OpenDayLight makes use of the OSGi platform to manage Java modules (or *bundles*, using its own terminology). They can be installed during running time and are grouped up under a single virtual machine.

Below there is an example screenshot of the OpenDayLight GUI. The first tab, “Devices”, shows a list of the devices connected to the controller, some information about them, and a graphical representation of the network. The second tab, “Flows”, lists all flows being carried through the network, while showing some of the information that can generally be found in an OpenFlow *flow table*.



Node Name	Node ID
OF|00:00:b8:27:eb:04:27:10	OF|00:00:t
OF|00:00:b8:27:eb:df:1a:29	OF|00:00:t

1-2 of 2 items

Fig. 3. 14 OpenDayLight GUI

3.5.5.2. *Open vSwitch*

OVS is a software implementation of a multilayer virtual switch. It makes use of OpenFlow and OVSDB (OVS Database Management Protocol) to bring together all the virtual machines within a hypervisor instance on a certain server.

3.5.6. User space

Below, the uncommon user space software that was used is detailed.

3.5.6.1. *chroot*

Tool that changes the apparent root directory of the current running OS by the one specified on its parameters when executed (in our case, the second OS installed with debootstrap through the ansible playbook [3.5.3]). Chroot is really useful when building a Linux image inside another Linux system, as required when preparing the TWIST distribution over a running Ubuntu. With the simple instruction below, the system root directory will change to the "child OS" and we will get inside it, and every other command executed will affect to the child system.

```
sudo chroot directory_to_childOS_rootfs
```

However, take into account that the running kernel will still be the one in the parent distribution, so any kernel-related package (such as *linux-image* or *linux-headers*) will be installed according to the running OS version.

3.5.6.2. *ethtool*

User space tool that displays or changes Ethernet card settings, as in Fig. 3. 7, for instance. It must be taken into account that in the Precise (12.04) version of Ubuntu, the most recent available version is 3.1, which does not still include the *–T* (*--show-time-stamping*) tool used to see the timestamping capabilities. That is why, to see the options of the TWIST NUCs we had to work with an Ubuntu Trusty (14.04) or Xenial (16.04) version, which supports later versions of this program.

Its use is really simple, the following order is enough to show the timestamping capabilities:

```
ethtool –T eth0
```

3.5.6.3. *nginx*

NGINX is an open-source HTTP server tool that was used to upload all the Ubuntu images that had been developed for the TWIST testbed, so as to be used by jFed.

CHAPTER 4. DEVELOPMENT AND RESULTS

4.1. Synchronisation evaluation

Due to the importance of synchronisation among nodes in our scenario, we performed some measurements to assess how well some of the techniques and protocols studied in chapter 2 would work in a real-life environment.

4.1.1. Measuring synchronisation

When thinking about how to measure the time difference between two nodes some ideas came up. As a first approach, we considered programing events at a certain frequency, set a timestamp when the event was launched at both machines and then compare the logs and measure the difference. Being this maybe a good idea for occasional measurements, we thought it wouldn't be suitable for a continuous tracking during hours, so we discarded it.

On a second round of thought, we chose the idea of using a three-node scenario (such as the one explained in section 3.2.2's Fig. 3. 4), where a client terminal would send broadcast messages to a couple of receivers, which share a synchronisation protocol (one of the nodes acts as master and the other as slave). That way, and taking profit of the timestamping capabilities of the network cards, a timestamp would be set to every received packet (the actual reception time being identical for both receivers, since the packets were transmitted in broadcast mode). Then, the accuracy in the synchronisation protocol was simply calculated by comparing both measurements.

In order to build this experiment, two scripts were developed: one for the transmitter, sending packets every second for a certain period of time; and one for the two receivers, listening for packets, timestamping at their arrival, translating the timestamp to a human-readable format and writing it to a CSV file for its posterior analysis. The transmission was built over UDP sockets, on a fixed port, and with the timestamping capabilities enabled on the socket (the SO_TIMESTAMP or SO_TIMESTAMPNS flags have to be declared before opening the socket; the only difference is that the first flag counts up to microseconds, while the second goes to nanoseconds).

In the annexes (Appendix C) you can have a closer look at the scripts used.

4.1.2. Performance evaluation

Once the proper functioning of the scripts was tested, the three-node topology was built, first with Raspberry Pis and then in the TWIST testbed. Two of the most popular software-based synchronisation protocols, NTP and PTP, have been tested.

4.1.2.1. *NTP*

NTP is known for being a decent synchronisation protocol in scenarios where precision and accuracy requirements are not extreme. However, we wanted to have a better look at its flaws on the long term. That is why a six hour experiment was run, with a packet being sent every second through the UDP socket explained before, i.e. 21600 samples.

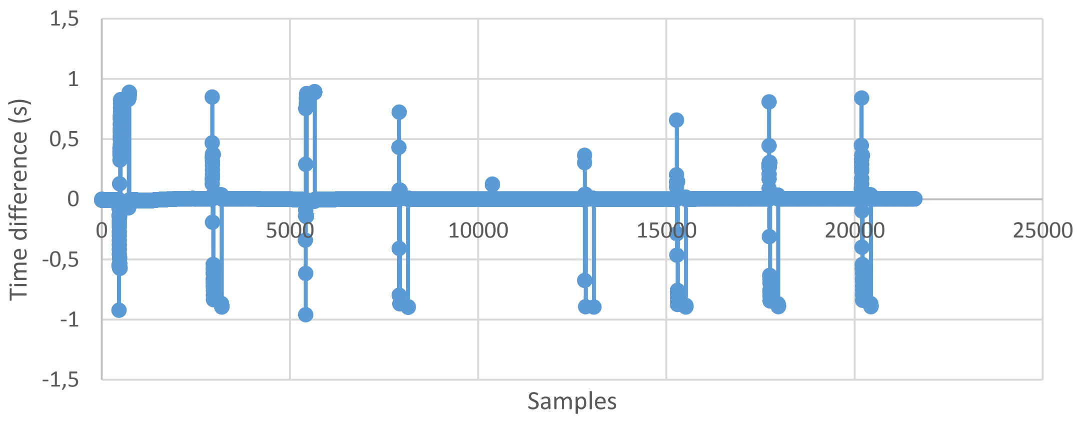


Fig. 4. 1 NTP synchronisation measurements during 6h

As can be seen in Fig. 4. 1, NTP does not seem suitable for the development of a TDMA application, since from time to time, the time difference between two nodes reaches values of nearly 1 second, and that would get even worse in multi-hop scenarios. If we have a closer look at the peaks, we appreciate there's a more or less fixed spacing between them, of about 2500 samples, which translates to 40-45 minutes; however, no explanation to why NTP would behave strangely every that period of time has been found.

Analysing this data deeper, we get to know that:

- Only 52.9% of the samples (11429 samples) have a value smaller than 1ms, which means that half of the time, microsecond accuracies are not achieved.
  In Fig. 4. 2, values greater than 5ms have been put to 0 and highlighted in orange. There you can see that there is some kind of curve evolution for the synchronisation: approximately for the first 1500 samples (25 minutes), NTP does not reach accuracy below 5ms, while later it lowers its values and stabilizes, with some obvious outliers every 40-45minutes.

- The average time difference during the 6h run is 879,4µs.

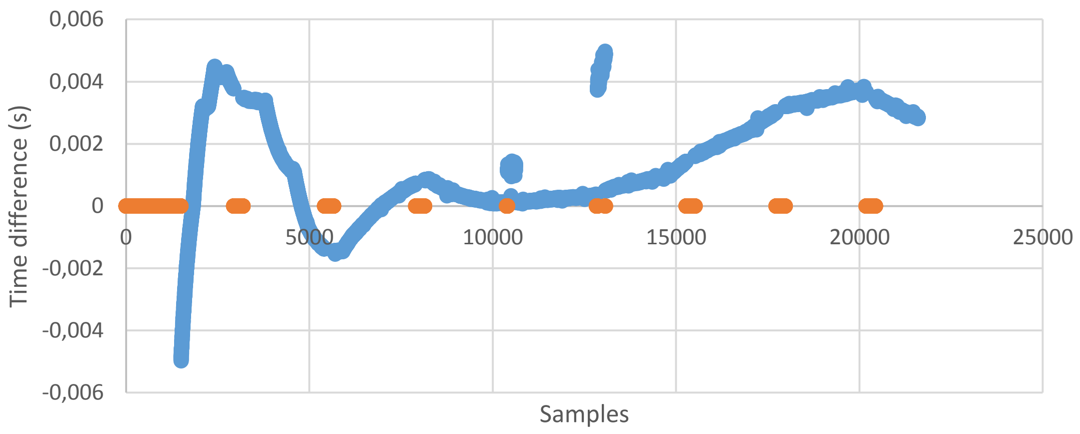


Fig. 4. 2 Measurements with values greater than 5 ms ignored

4.1.2.2. *PTP*

In principle, PTP would be a better option for meeting the requirements of our project. To check this assumption, a 2h (7200 samples) experiment with packets being sent every second and software timestamping was run.

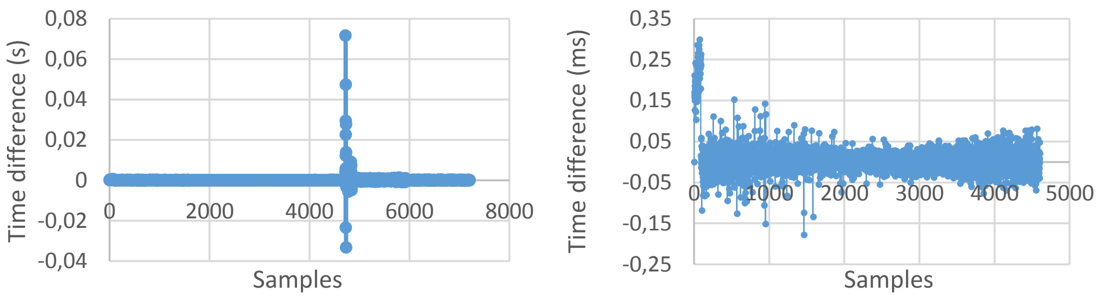


Fig. 4. 3 PTP synchronisation measurements during 2h (left) and only in the first 75 minutes (right)

As depicted in Fig. 4. 3 (left), PTP does not show as many outliers as NTP. There are still some accuracy losses, but they are far away from the nearly 1s suffered in NTP (in fact, they do not reach 80ms). If we consider only the first 4500 samples (75 minutes) (Fig. 4. 3 [right]) we can see that most values are kept under the 100µs accuracy after a transient initial time with peaks under 300µs.

Further analysis of the 7200 samples provides us with the following information:

- 99.0% of the samples have a time difference below the millisecond, 91.1% below 100µs and 58% below 10µs.
- The average time difference during those two hours was of 46.8µs.

4.1.3. Conclusions

As we already anticipated, despite being a good solution for deployments without strict requirements of synchronisation accuracy, NTP does not provide us with the precision that a TDMA-enabled environment asks for. Only half of the samples show a time difference below the millisecond threshold.

On the other hand, PTP (with its *linuxptp* implementation and with no further protocol modification) grants better synchronisation, staying below the 100µs nine tenths of the time, values that are much more compliant with the transmission time of WLAN environments (i.e. comparable to frame transmission time).

Finally, it must be said that both solutions studied here show some problems on the long run, with peaks of desynchronisation that could possibly affect the performance until the protocol recovers its normal operation.

4.2. HTDMA

In this section, the summary and conclusions of the experiments run in the HTDMA scenario can be found. There is a review of the process to find the best slot duration, a comparison between HTDMA and pure CSMA, and some other experiments aimed at taking profit of the advantages of this medium access control technique.

4.2.1. Slot times and first steps with the system

As seen in section 3.4, the deployment of a hybrid TDMA MAC is done using a python user-space script that exposes some of the parameters under which the scheduling will be configured. First of all, we wanted to see how the system behaved with different slot durations, and later there was some further work with performance under different slot-assignment conditions and how QoS can be enabled in such a system.

4.2.1.1. *Slot duration*

In order to find the best configuration for an optimal behaviour of the protocol, several experiments were performed, with the first objective of finding the most suitable length for each slot. We found it interesting to see what happened when the duration of the slot corresponded to N times the transmission time of a packet, being N both smaller and greater than 1, i.e. studying the impact of having slots shorter than the real duration of a transmission or slots that can accommodate several packets.

To do so, a scenario with only 2 devices connected in an Ad-hoc network was built. One of the computers would work as a server, while the other would be the client, and would have an active HTDMA MAC on its wireless interface. That way, using tools such as *iperf* and *Wireshark*, we are able to analyse throughput and packet bursts.

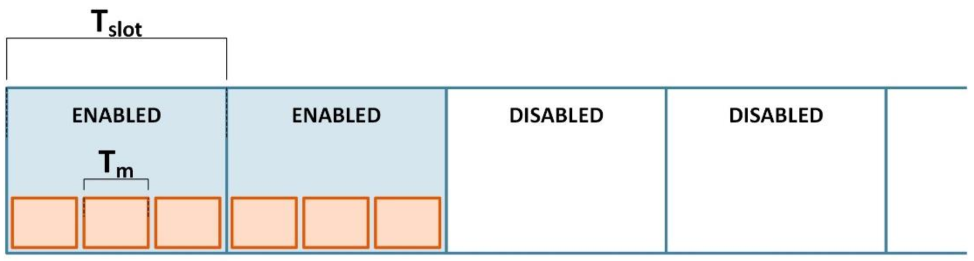


Fig. 4. 4 Slot assignment for the “slot duration” experiments

Fig. 4. 4 illustrates an example slot assignment, where: in blue, the slots where packets could be sent; in white, the slots that were disabled; and in orange, the packets that fit inside a slot, being $T_{slot} = 3T_m$ in this example. For the experiments, time was divided in 20 slots, only the first of them being used for transmission, while the rest were disabled. With that scheme, using traffic analysis software such as *Wireshark* it was easy to detect packet bursts, as they would have to happen only for the duration of 1 time slot every cycle (the duration of a cycle is 20 times the duration of a single time slot). The length of each slot, T_{slot}, was varied to see which value provides the best performance.

Logically, T_{slot} would have to be proportional to the transmission time of a single packet, so first of all, some calculations about transmissions would have to be done. The time required to transmit a frame is shown in (4.1), where some values are constant for the scenario built with 802.11g and a physical bitrate of 54Mbps: *DIFS* = 28µs, *SIFS* = 10µs and δ can be neglected, as it represents the propagation delay (which is really small considering the distances in typical WLAN environments).

$$T_m = DIFS + T_{data} + \delta + SIFS + \delta + T_{ACK} \qquad \textbf{(4.1)}$$

The transmission time of a packet, T_{data}, corresponds to (4.2), where T_H is the duration of the preamble/header and equals 20µs, Ts_{ext} is the signal extension time of 6µs, L depends on the packet length (L includes 36Bytes of MAC and LLC headers plus the payload, L_{DATA}) and r is the transmission rate (54Mbps for normal frames or 24Mbps for ACKs). T_{ACK} equals T_{data} using L = 14 Bytes.

$$T_{data} = T_H + 4\left\lceil\frac{(22+L\cdot 8)}{4r}\right\rceil + T_{s_{ext}} \qquad \textbf{(4.2)}$$

Following these equations, we obtain, for example, T_m for packets of 800 Bytes is 226µs. Then, the slot duration, proportional to this value, could be half the transmission time of a packet, 1 time that value, 2, 3, 5, 10, 20 and 30 times (enough variety so as to see a tendency in the behaviour).

After several experiments, the results were:

- If $T_s < 10T_m$, 8 to 10 packets are transmitted on a single slot.

- If T_s = 10, 20 or $30T_m$, on average, 10, 20 or 30 packets are transmitted on a single slot.

That behaviour can be explained due to a hardware limitation introduced by the wireless card. Apparently, the Qualcomm Atheros AR9280 chipset has a hardware queue that fills even though the open slot does not theoretically let so many packets through, so the minimum possible transmission corresponds to those 8 to 10 frames.

Experiments were run with other packet lengths, and the same results were obtained, so in conclusion we can assume that, for the scheduling to work as close as possible to its theoretical model, slots should be able to admit at least 10 frames. Otherwise, slots may overlap since, at the end of the slot, there may be pending frames in the hardware queue.

4.2.1.2. *Performance evaluation*

Once the first doubts regarding the operation of HTDMA are solved, it would be interesting to study the performance of this MAC protocol in terms of throughput and jitter (measurements that can be easily obtained with a client-server topology and *iperf*).

To do so, experiments with three variables were run:

- Frame size: 1400, 800 and 100 Bytes

- A cycle of 10 slots, with different amounts of open slots: 1/10, 2/10, 5/10 or 10/10.

- Number of frames per slot (i.e. slot length): 10, 20 or 30 frames.

Launching *iperf* with the command below, a UDP flow at 25Mbps with the desired packet length (subtract 28 Bytes corresponding to IP and UDP headers) is obtained. Moreover, the HTDMA python script had to be modified according to the slot duration depending on the amount of packets to fit in a slot.

```
iperf –u –c @IPclient –b 25M –l L –t 30 –i 3
```

The results are summarized in the figures below (due to space restrictions, here we show only a couple of representative examples of throughput and jitter measurements; Appendix D includes a more complete set of results), comparing the values obtained against legacy CSMA.

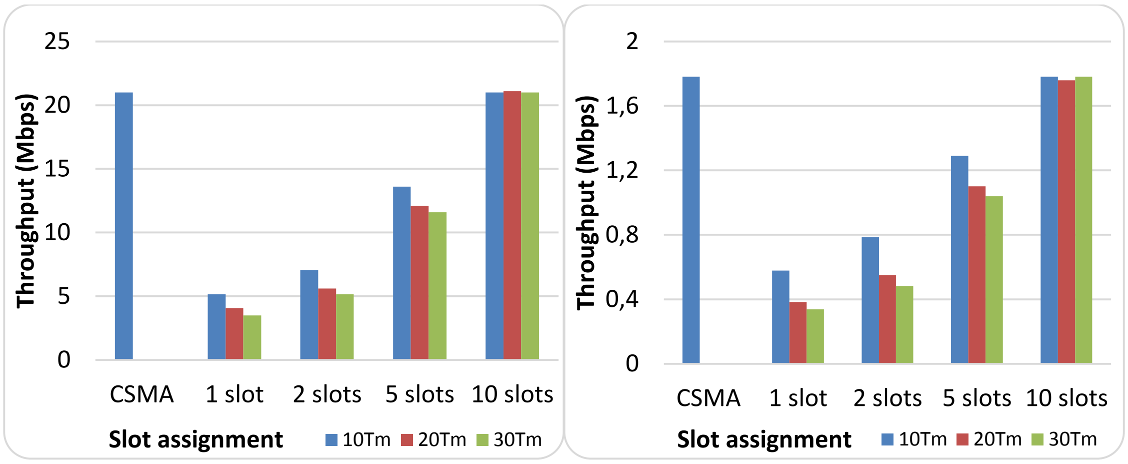


Fig. 4. 5 Throughput for packets of 1400 Bytes (left) and 100 Bytes (right)

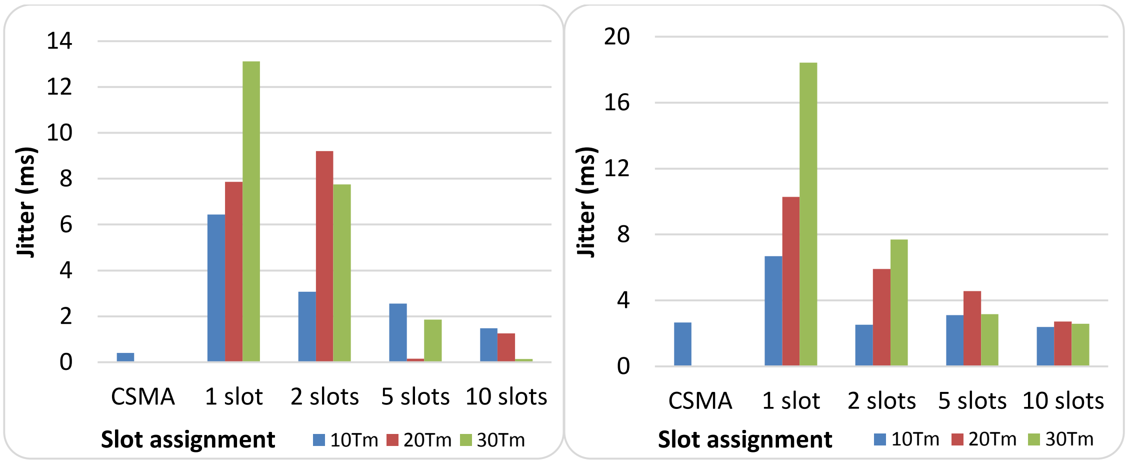


Fig. 4. 6 Jitter for packets of 1400 Bytes (left) and 100 Bytes (right)

In Fig. 4. 5 there is a clear representation of how smaller time slots achieve a better performance than bigger ones in terms of bandwidth, regardless of the proportion of open slots; having a slot duration of 10 frames is preferable to it being 30 times the frame transmission time both in terms of throughput and delay.

Jitter will obviously be worse with fewer active slots and longer slots, as the variation in the arrival of frames under those two conditions will be larger. That is exactly what Fig. 4. 6 illustrates. Taking as an example the single slot measurements, we can see that jitter is nearly directly proportional to slot duration. That is because the duration of those 9 inactive slots is longer in each case.

In order to confirm this behaviour, some more experiments were run, where the proportion of open slots (the open slots were consecutive too) kept invariable: 2 out of 10, 4 out of 20 and 8 out of 40. Packet size of 1000 Bytes, a bandwidth of 25Mbps and slots of size equivalent to 10Tm were used.

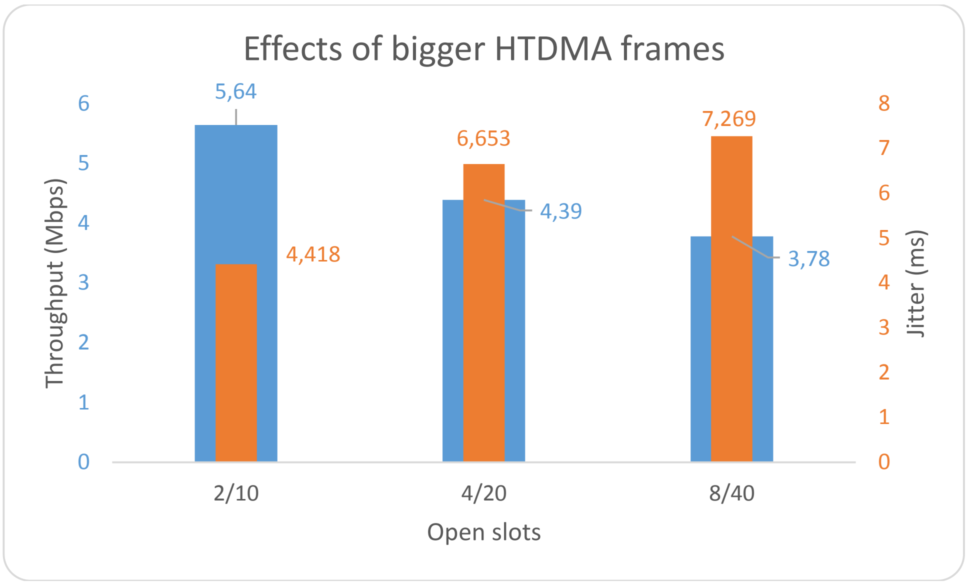


Fig. 4. 7 Throughput and jitter with a constant proportion of open slots

Fig. 4. 7 illustrates how jitter increases and throughput decreases as long as the HTDMA cycle increases its size, despite having the same proportion of usable time.

As a conclusion, we can assume that the smaller the slot the better the performance, taking into account the limitations imposed by the hardware.

4.2.2. QoS over HTDMA

Quality of Service is generally enabled in legacy CSMA by means of IEEE802.11e's Enhanced Distributed Channel Access (EDCA). This system sets four different priorities, called Access Categories (AC), each having different transmission parameters in order to set different priorities. Concretely, the maximum and minimum contention window sizes, inter frame spacing (Arbitration Interframe Space, AIFS) and duration of a transmission opportunity (TXOP) are defined for each AC. These parameters have some default values, which in a Linux-based device can be found (and thus modified) in a *mac80211* kernel module file[5].

The HTDMA system makes it possible to have a more flexible QoS provisioning by means of defining the time slots available for each type of traffic. A dynamic approach to the scheduling can result in a big improvement on the performance of the wireless network.

So as to test the possibilities of HTDMA regarding QoS provisioning, some experiments were run where various simultaneous UDP data flows were sent from a single client to a common server. Each of the flows has a different EDCA

[5] *mac80211_driver_directory*/net/mac80211/util.c

definition (levels 2, 3 and 5 in Table 3. 1; they can be defined in *iperf* with the *–s* flag followed by the code of the AC) and the same offered traffic of 20Mbps.

Then, two tests were performed, one using the legacy CSMA MAC, and another one using HTDMA. The experiments consisted, at first, in three simultaneous flows with ToS definitions AC5, AC3 and AC2, respectively; and then, after one minute, two new AC3 flows are added. The experiments worked as follows:

– Legacy CSMA works assigning bandwidth depending on the Access Category and its default configuration parameters. The results are shown in Fig. 4. 8.

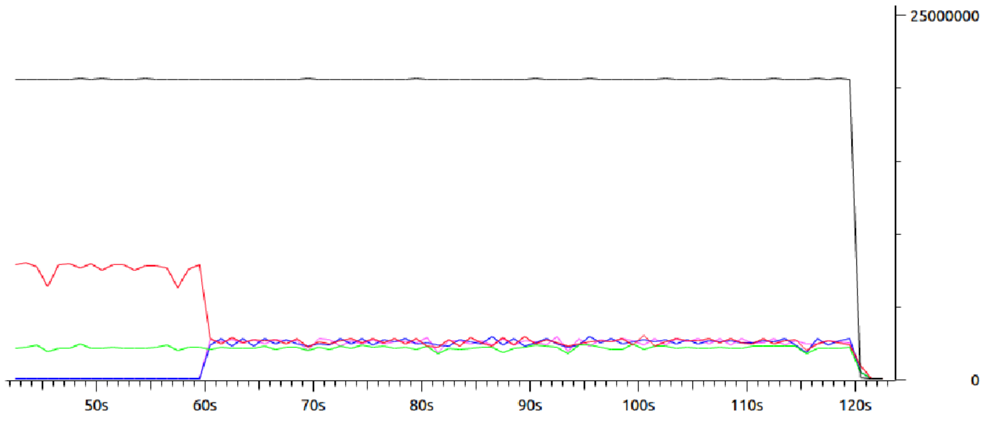


Fig. 4. 8 Throughput measuremens in QoS environament using CSMA

– With HTDMA, there was a manual change on the slot assignments (as in Fig. 4. 9), emulating an ideal dynamic scheduling: for 3 flows, a time frame of 12 slots with an assignation of 8, 3 and 1 in descending AC priority; for the case of 5 flows, 18 slots, assigning of 8, 9 and 1 to give more resources to AC3 flows. The results are shown in Fig. 4. 10.

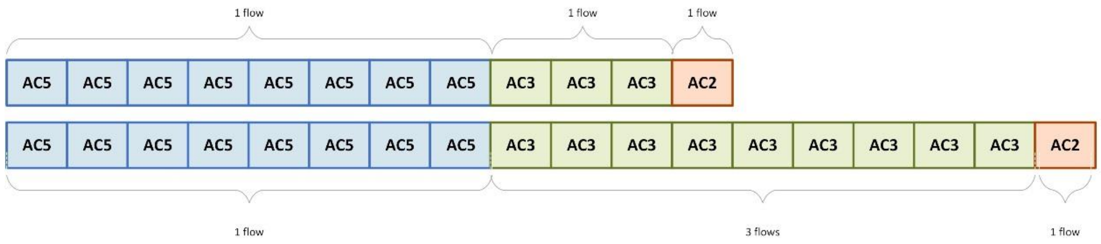


Fig. 4. 9 Slot assignment in the HTDMA QoS experiment

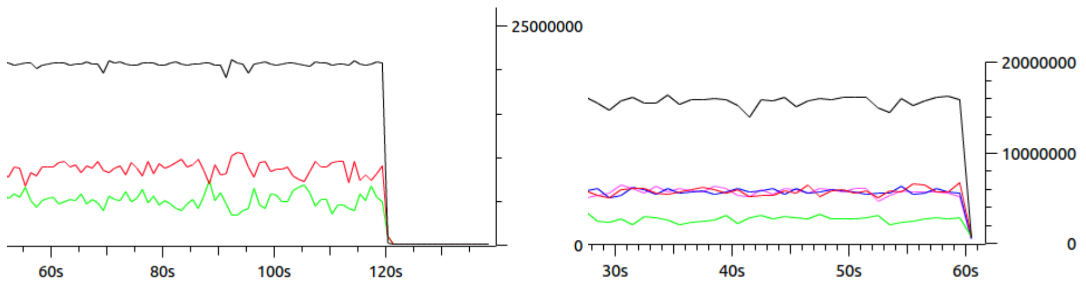


Fig. 4. 10 Throughput measurements with 3 (left) and 5 flows (right)

It is interesting to note that legacy CSMA does not let us dynamically adapt the traffic preferences, as it performs its own prioritization according to the values of the parameters configured in the driver. That is why the same example run with CSMA adapts worse to the varying traffic demands: the highest priority traffic is kept stable at maximum bitrates, while the traffic of medium and low priority are brought to really similar values of throughput, which is *unfair* in terms of individual flows. Meanwhile, the assignment performed in HTDMA is much *fairer*, as AC5's throughput is slightly reduced to allocate the two new AC3 flows. The *fairness* of the system can be calculated with a variation of the *Jain's fairness* index (4.3), where f_i is the actual throughput achieved by each of the *i* flows and F_i is the Max-min fair throughput for each flow, considering its weight as compared to the rest.

$$J = \frac{\left(\sum_i^n \frac{f_i}{F_i}\right)^2}{n \cdot \sum_i^n \left(\frac{f_i}{F_i}\right)^2} \qquad \textbf{(4.3)}$$

That way, for the case of 5 flows sharing the resources, the *Jain's fairness* index for CSMA is 0.77, while for HTDMA it is 0.96.

Nevertheless, we can think of different scenarios where, for example, fairness is not important and traffic AC5 is critical, so that it needs maximum throughput. With HTDMA, this can also be achieved by scheduling more transmission time to AC5. That is why HTDMA can be used in many different situations, enabling a wider range of QoS policies through different scheduling algorithms and different optimization parameters.

In conclusion, EDCA is a decent technique for QoS provisioning in standard scenarios, but newer techniques such as HTDMA enable a more precise provisioning with more options so as to dynamically accommodate the medium to the requirements of the system.

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

With regards to the development of this thesis, the overall valuation is positive, as the main objectives of the project have been accomplished. First of all, there was the study of potential synchronisation tools to be used in the project, both in terms of theoretic research and practical performance in small test scenarios; the experiments that were performed with synchronisation protocols provided useful information about the real capabilities of PTP and NTP, and how much better the former is. Later, and with the help of the research team in I2CAT I was able to get familiar with the development of the SDN SENSEFUL platform, which led me then into preparing my own Linux-based image to deploy the system in the WiSHFUL TWIST testbed, making use of some of the tools provided by the TKN team. Finally, the HTDMA mechanism was studied, by means of understanding how it worked, installing it on my local testbed and the image for the TWIST, and performing some initial tests on its performance comparing it to legacy CSMA MAC. Some of these experiments have resulted in learning about optimal slot durations, QoS and throughput performance, etc., which is really useful for the project itself.

In a more personal note, the development of this project has been really gratifying, as a lot of the knowledge acquired during my studies have become useful while working on it. Moreover, it has introduced me into fields that are not so deeply explored in the degree, such as synchronisation or Software Defined Networking, while letting me learn about the latest researches in some other areas such as the introduction of TDMA in Wireless LANs. While the main objectives where fulfilled, and each of them meant a challenge in some sense, some of them were more a bit of a problem than others. The first steps of the project, understanding and catching up with the SENSEFUL project, were difficult, as it was something really big that had been under development for long, although I had the help of the team for anything I needed. Then the pace slowed down as everything started working and my test scenarios were set up. Finally, the adaptation of all the work that had been done into the German testbed was also a big challenge, as lots of specifications and requirements had to be met regarding both the hardware itself and the SENSEFUL platform, and the software used to interact with TWIST did not provide enough information whenever problems occurred, so lots of mail exchanges and teleconferences became necessary. However, and seen from the present, having been able to fulfill all the expectations that were set up at the beginning of the project, I am really happy with the result.

5.1. FUTURE WORK

Despite having done everything that was once planned for this thesis, there is still some more work to do regarding the SENSEFUL project. The team, together with the TKN group in Berlin, are planning the topology and hardware requirements to run some of the experiments which may result in more interesting conclusions.

Synchronisation is still to be tested in bigger multi-hop scenarios too. It has been studied in this thesis that PTP can provide a good level of synchronisation among nodes, but it has still to be tested whether this level of accuracy is enough for a system that relies on the perfect timing between slots.

The state of the art of TDMA MAC over WLANs shows that there are many research groups interested in this area. It was shown that there are already consumer solutions that offer scheduling in hybrid medium access mechanisms. However, the tendency of the sector is to have software implementations that can be applied over any kind of hardware, so alternatives such as the proposed HTDMA seem to be the future.

In terms of HTDMA, there are still several tests which we want to perform, so as to learn more about this mechanisms and its real capabilities. We want to test throughput and jitter in topologies were several machines use this MAC, also take profit of the hybrid composition of the system to assign the same slots to several devices so that they work both under TDMA and CSMA, etc.

Finally, there is the provisioning of QoS using HTDMA and compare it to the standard CSMA's EDCA tool. QoS over HTDMA implies a big challenge, as slot reservations have to be very dynamic in order not to waste transmission opportunities with slots that are empty because of waiting for a future flow or once a flow ends. However, once everything is set up and working correctly, its performance should be much better –and way more configurable– than the capabilities of today's Wi-Fi.

As long-term objectives for SENSEFUL, the team wants to prove all the advantages of this Wi-Fi micro-cell system with a shared medium between backhaul and access network, focusing, above all, on the dynamicity in resource management thanks to SDN, the benefits of *BigAP* regarding handover, and how HTDMA can solve some of the endemic problems of Wi-Fi.

Finally, regarding the environmental impact of this thesis there is not much to say, as the current legislation makes sure that WiFi works within non-dangerous limits of transmission power. Moreover, the concept of the project itself is aimed at minimizing interference by using scheduled transmissions, optimize network performance and thus producing energy savings.

All in all, this thesis covers the most relevant aspects of future SDN-driven small cell networks, the present of synchronization in wireless networks and the future of medium access mechanisms. Experiments were run so as to prove the points made in the theoretical chapters, and there is a finishing brief explanation of what is still to come.

BIBLIOGRAPHY

[1] I2CAT Foundation website - http://www.i2cat.net/en

[2] WiSHFUL Project website - http://www.wishful-project.eu/

[3] A. Zubow, S. Zehl, A. Wolisz, “BIGAP–Seamless Handover in High Performance Enterprise IEEE 802.11 Networks”. Technical Report TKN-15-004, Telecommunication Networks Group, Technische Universität Berlin, 2015

[4] TWIST testbed website - https://www.twist.tu-berlin.de/

[5] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015-2020 White Paper - http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

[6] Small Cell Forum website - http://www.smallcellforum.org/

[7] Small Cell Forum, “Backhaul technologies for small cells: use cases, requirements and solutions”, February 2013

[8] Software Defined Networking’s Open Networking Foundation definition - https://www.opennetworking.org/sdn-resources/sdn-definition

[9] Software Defined Everything’s (SDx) SDN architecture definition- https://www.sdxcentral.com/sdn/definitions/inside-sdn-architecture/

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, “OpenFlow: Enabling Innovation in Campus Networks”, March 2008

[11] Metro Ethernet Forum, “LTE Synchronisation v075.06.01”, pages 1 - 28, February 2014

[12] K. Stanton, C. Aldana, “Addition of p802.11-MC Fine Time Measurement (FTM) to p802.1AS-Rev: Tradeoffs and Proposals”, IEEE 802.1 Plenary, March 2015

[13] “Distribution of timing information through packet networks”, ITU-T Recommendation G.8264, October 2008

[14] “Network Limits for Time Synchronization in Packet Networks”, ITU-T G.8271.1/Y.1366.1, July 2013

[15] I. Tinnirello, P. Gallo. "Supporting a pseudo-TDMA access scheme in mesh wire-less networks." Wireless Access Flexibility, pages 80-92, Springer Berlin Heidelberg, 2013

[16] Y. Khan, M. Derahkshani, S. Parsaeefard, T. Le-Ngoc, "Self-Organizing TDMA MAC Protocol for Effective Capacity Improvement in IEEE 802.11 WLANs." 2015 IEEE Globecom Workshops (GC Wkshps). IEEE, 2015

[17] Z. Yang, J. Zhang, K. Tan, Q. Zhang, Y. Zhang, "Enabling TDMA for today's wireless LANs." Computer Communications (INFOCOM), 2015 IEEE Conference on. IEEE, 2015

[18] J. Lee, M. Uddin, J. Tourrilhes, S. Sen, S. Banerjee, M. Arndt, K. Kim, T. Nadeem, “meSDN: Mobile Extension of SDN”

[19] Raspbian - https://www.raspbian.org/

[20] Documentation for Linux wireless modules - http://www.linuxwireless.org

[21] Ath9k drivers - https://wireless.wiki.kernel.org/en/users/drivers/ath9k

[22] iMinds - https://www.iminds.be/en

[23] IEEE 802.11v - http://www.ieee802.org/11/Reports/tgv_update.htm

[24] G. Venkatesan, “Timing Measurement”, Intel Corporation, July 2008

[25] K. Ichikawa, “Precision Time Protocol on Linux – Introduction to linuxptp”, Fujitsu Limited, LinuxCon Japan 2014

[26] The Linux PTP Project - http://linuxptp.sourceforge.net/

[27] Linux PTP RedHat documentation - https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/ch-Configuring_PTP_Using_ptp4l.html

[28] A. Mahmood, G. Gaderer, “Towards High Accuracy in IEEE 802.11 based Clock Synchronization using PTP”, Institute for Integrated Sensor Systems, Austrian Academy of Sciences

[29] A. Mahmood, R. Exel, T. Sauter, “Delay and Jitter Characterization for Software-Based Clock Synchronization Over WLAN Using PTP”, IEEE Transactions on Industrial Informatics, Vol. 10, Nº 2, May 2014

[30] Ubiquiti Networks, “airMAX TDMA Technology Datasheet”, California (EEUU) 2014

[31] jFed website - http://jfed.iminds.be/

ABBREVIATIONS AND ACRONYMS

AC	Access Category
ACK	Acknowledgement
AGNSS	Assisted Global Navigation Satellite System
AIFS	Arbitration Interframe Space
AP	Access Point
API	Application Programming Interface
BSS	Basic Service Set
CDMA	Code Division Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSV	Comma-Separated Values
DCF	Distributed Coordination Function
DIFS	DCF Interframe Space
DFS	Dynamic Frequency Selection
DSSS	Direct-Sequence Spread Spectrum
E2E	End To End
EDCA	Enhanced Distributed Channel Access
FTM	Fine Time Measurement
IEEE	Institute of Electrical and Electronics Engineers
GLONASS	Global’naya Navigatsionnaya Sputnikovaya Sistema
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HRM	Hypothetical Reference Model

HTDMA	Hybrid TDMA
MAC	Media Access Control
MWN	Mesh Wireless Network
NBI	Northbound Interface
NIC	Network Interface Controller
NUC	Next Unit of Computing
OFDM	Orthogonal Frequency-Division Multiplexing
OFDMA	Orthogonal Frequency-Division Multiple-Access
OS	Operating System
P2P	Peer To Peer
PCF	Point Coordination Function
PHC	PTP Hardware Clock
PIFS	PCF Interframe Space
PON	Passive Optical Network
PSK	Phase-Shift Keying
PTP	Precision Time Protocol
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
RAN	Radio Access Network
RTS/CTS	Request to Send / Clear to Send
SBI	Southbound Interface
SDN	Software Defined Networking
SDR	Software Defined Radio
SENSEFUL	*S*DN driven Joint Access Backhaul coordination for next generation d*ense* Wi-Fi Small Cell networks via WiSH*FUL* APIs
SIFS	Short Interframe Space

TCP	Transmission Control Protocol
ToS	Type of Service
TDMA	Time Division Multiple Access
TLS	Transport Layer Security
TM	Time Measurement
TWIST	TKN WIreless NetworkS Testbed
TXOP	Transmission Opportunity
UDP	User Datagram Protocol
WiSHFUL	Wireless Software and Hardware platforms for Flexible and Unified radio and network controL

ANNEXES

APPENDIX A. OTHER SYNCHRONISATION TOOLS

A.1. DCF77

The DCF77 is a standard long-wave time signal broadcasted with an availability of 0.997 by a radio station located near Frankfurt am Main, Germany, which is controlled by the German nation physics laboratory Physikalisch-Technische Bundesanstalt (PTB). Working 24/7, and with an agreed downtime of 26.28 hours per year (corresponding to its availability), it radiates a 50kW of nominal power signal at 77.5 kHz generated by local atomic clocks.

The DCF77 is used to broadcast the German time and date, and the signal has mainly been used to synchronise clocks, appliances, industrial equipment or radio and TV stations.

The signal is modulated on amplitude, and coded with pulse-width at 1 bps. It is repeated every minute and the packet carries the following information: current date and time, leap second flag, announcement bit (change CET-CEST), CET/CEST flag, abnormal operation flag and parity bits. The time is represented in binary and the time transmitted corresponds to the following real minute. The transmission goes as follows: seconds 0-20 for special flags, 21-28 for seconds, 29-34 for hours and 36-58 for the date, leaving a last bit for another flag.

The DCF77 is said to be received over 2000 km from the transmitter, although interferences, signal propagation and other factors can make that distance longer or shorter. However, in our case we would be working within the margins of operation, so it was an option that was considered.

DCF77 receivers can easily be found and purchased[6], and there are many developers that have open-source software[7] ready to decode the signal and synchronise the device to which the receiver is connected.

A.2. Synchronous Ethernet (SyncE)

The Synchronous Ethernet (SyncE) ITU-T standard is built over legacy Ethernet standards and its main aim is to make it easier to transport clock signals over the physical layer. To do so, transmit clocks in SyncE devices are not synchronised to their own crystal oscillator, but to an external traceable Stratum-1 clock which, ideally, should be unique for the whole network. SyncE is built on three ITU-T recommendations:

[6] http://www.conrad.com/ce/en/product/641138/DCF-receiver-module-Compatible-with-C-Control

[7] https://blog.blinkenlight.net/experiments/dcf77/the-clock/

- G.8261 “Timing and synchronization aspects in packet Networks”: establishes an introduction to SyncE, while referring to the network architecture and wander performance.
- G.8262 “Timing characteristics of a synchronous Ethernet equipment slave clock”: defines SyncE clock equipment.
- G.8264 “Distribution of timing information through packet networks” ([13]): describes the Synchronisation Status Messaging (SSM).

A.2.1. *Clock distribution on the physical layer*

While the synchronisation methods summarized up to now rely on packets in order to transmit the time information all over the network, this requires several filtering algorithms so as to avoid any error that would affect the time count. SyncE does not have this kind of problem, as it transmits the clock directly and continuously over the physical layer on full-duplex scenarios. However, that is exactly what makes it impossible to implement in a project like ours, because this standard depends on the continuous transmission of slow Ethernet frames (as defined in [13] section 11 ‘*SSM for synchronous Ethernet*’), requiring a full-duplex environment unavailable in a Wi-Fi network.

A.3. Global Navigation Satellite Systems (GNSS)

The most typical GNSS system used in telecommunication environments is GPS, which was designed to provide accurate time and location references (or only time in bad GNSS-signal conditions at the expense of position being manually supplied) in any point on the planet Earth.

GPS works with 24+7 satellites (24 permanent and 7 additional ones that can be placed in the system at will) that provide CDMA spread-spectrum signal in the 1.2 and 1.5 GHz bands with low bit rate transmissions and a theoretical accuracy of up to 14 nanoseconds (although real-system conditions set 100 ns as a more realistic figure with which to work).

The main problem GPS has is that *direct sky visibility* is required for it to work in most of the cases, which reduces its functionality in some scenarios. That is because the general GNSS signal strength at the surface of our planet is around -130 dBm, what makes it about 30dB under the general noise floor. However, some vendors have started to offer devices that work with non-direct visibility using multi-path signal reception, guaranteeing a time reference with a 500ns accuracy. Moreover, some regions and countries have developed their own GNSS deployments (for example, GPS is the American version, GLONASS the Russian one, and the European Union has Galileo), and new devices are appearing that can work with several signals at the same time so as to improve the time and location accuracy.

Its high accuracy and global functioning makes GPS one of the most used systems for primary synchronisation, and in fact lots of Stratum 0 clocks are based on GPS time and frequency references. However, the direct sky visibility restriction, bad urban penetration, and it being very susceptible to interferences, reduces its use cases in real scenarios.

A.4. Cellular network listen (CNL)

CNL bases its functioning on the same idea with which User Equipment obtain synchronisation, i.e. listening to surrounding cellular base stations of any technology, such as GSM, LTE, etc. This way, small cells can simulate the work of a UE to synchronise their time base to those of the adjacent cells.

CNL is then a good idea for the deployment of small cells in highly populated areas, where it will be at range of many others cells and then it will be able to perform its functioning correctly. However, before a CNL deployment, the environment should be studied carefully, because we cannot allow a scenario where several small cells rely on each other (without a good primary reference) because they all work with CNL.

A.5. Miniature atomic frequency references

Stratum 0 clocks typically rely either on GNSS or on atomic clocks, which can meet synchronisation accuracy requirements (from 0.1ppb to 1ppb) without the need of an external reference.

However, this kind of technology is only suitable, for example, for important base stations, as its costs multiplies several times the one offered by other options. Moreover, only frequency synchronisation is achievable this way, not time or phase.

A.6. Hybrid technology options

As shown in this section, there are a lot of different technologies that can be used in small-cell scenarios in order to provide synchronisation options. Each of those has its own pros and cons, and none of them is the most suitable choice for all kind of scenarios. That is why nowadays, combinations of several of these technologies is a better option when designing a system that requires of synchronisation capabilities.

There is a huge amount of possible combinations, but just to set some examples of how the joint of different technologies can provide new advantages to the system, here there are some alternative approaches:

- Network solutions and AGNSS: as explained in the GNSS section the low strength of the signal makes it compulsory for GNSS receivers to recover the signal from the noise by means of correlation with a correct code, which will vary depending on the frequency modulation of the Doppler effect of the transmitter in the moving satellites. That is why AGNSS receivers are more common these days. They use extra information deployed over the utility network so as to speed up the search of the proper code. If network synchronisation solutions such as PTP or NTP are incorporated into the system, the whole process can be improved by adding a constant frequency and time reference (which can be also useful against outages in coverage loss periods).

– PTP and SyncE: ITU-T G.8271.1 recommendation ([14]), titled '*Network limits for time synchronisation in packet networks*', provides itself an example of joint SyncE and PTP synchronisation network to provide frequency, time and phase to a certain end application (which for instance could be an eNodeB in an LTE network). This way, using an infrastructure as the one presented in the image below (Fig. A. 1), SyncE can be used to synchronise each of the up to 21 PTP nodes (20 Boundary Clocks *T-BC* and 1 Grand Master *T-GM*) that form the chain that feeds the end application.

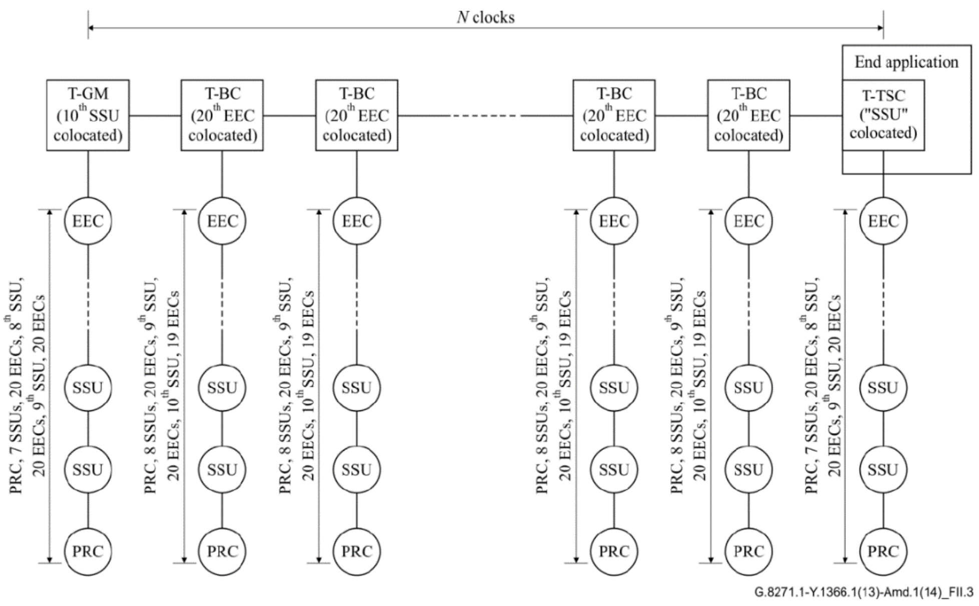

N clocks
End application
T-GM (10th SSU colocated)
T-BC (20th EEC colocated)
T-BC (20th EEC colocated)
T-BC (20th EEC colocated)
T-BC (20th EEC colocated)
T-TSC ("SSU" colocated)
EEC
SSU
SSU
PRC
PRC, 7 SSUs, 20 EECs, 8th SSU, 20 EECs, 9th SSU, 20 EECs
PRC, 8 SSUs, 20 EECs, 9th SSU, 20 EECs, 10th SSU, 19 EECs
G.8271.1-Y.1366.1(13)-Amd.1(14)_FII.3

Fig. A. 1 HRM with physical layer frequency support ([14] page 16)

This way, each of the PTP nodes plus the end application itself have a reliable frequency reference, fact that reduces the overall dynamic time error that can be accumulated in the usage of long synchronisation chains. As a side advantage, in the case of the chain breaking, the end application could be able to maintain time precision for a certain time until the reference is recovered.

APPENDIX B. SENSEFUL SHOWCASE SCENARIO

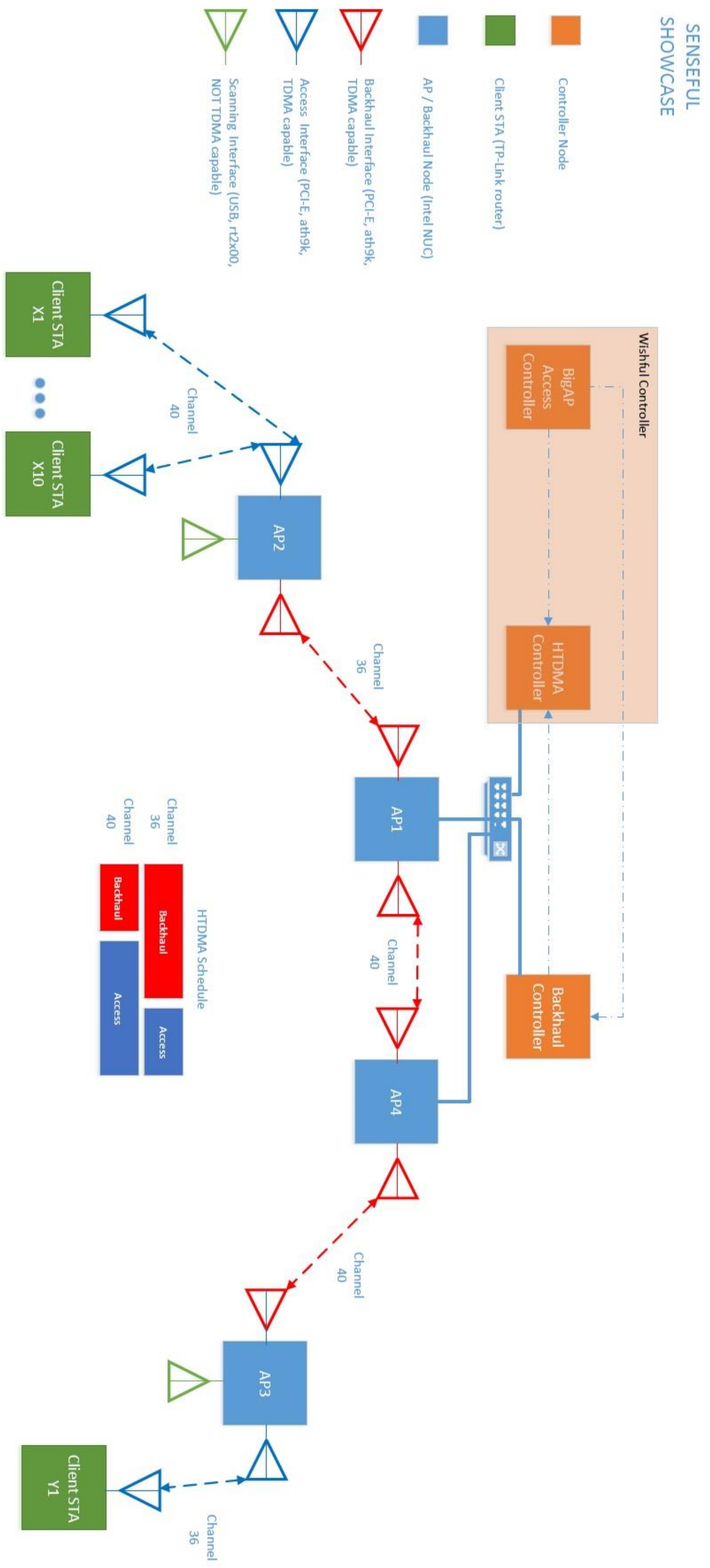
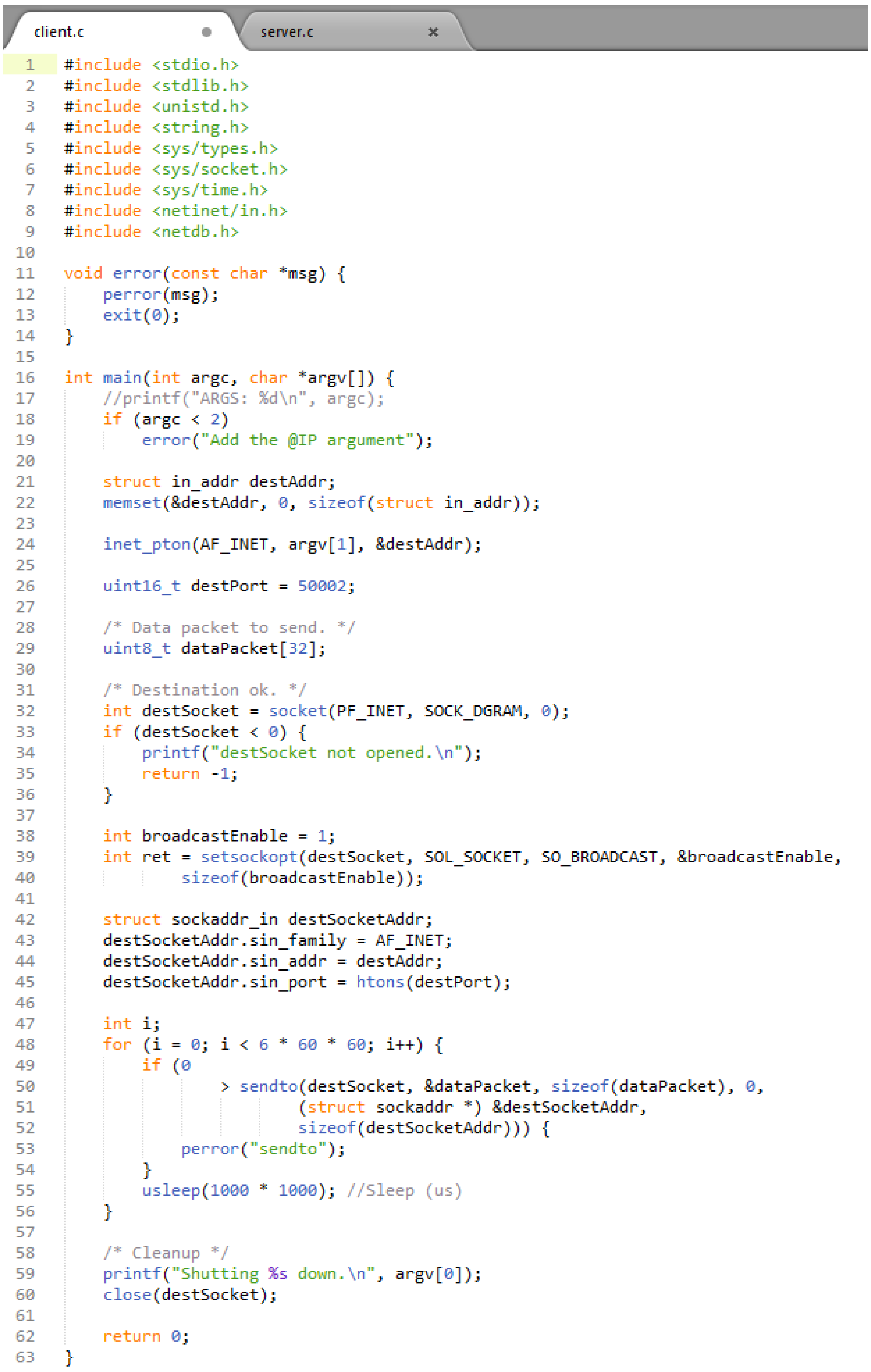


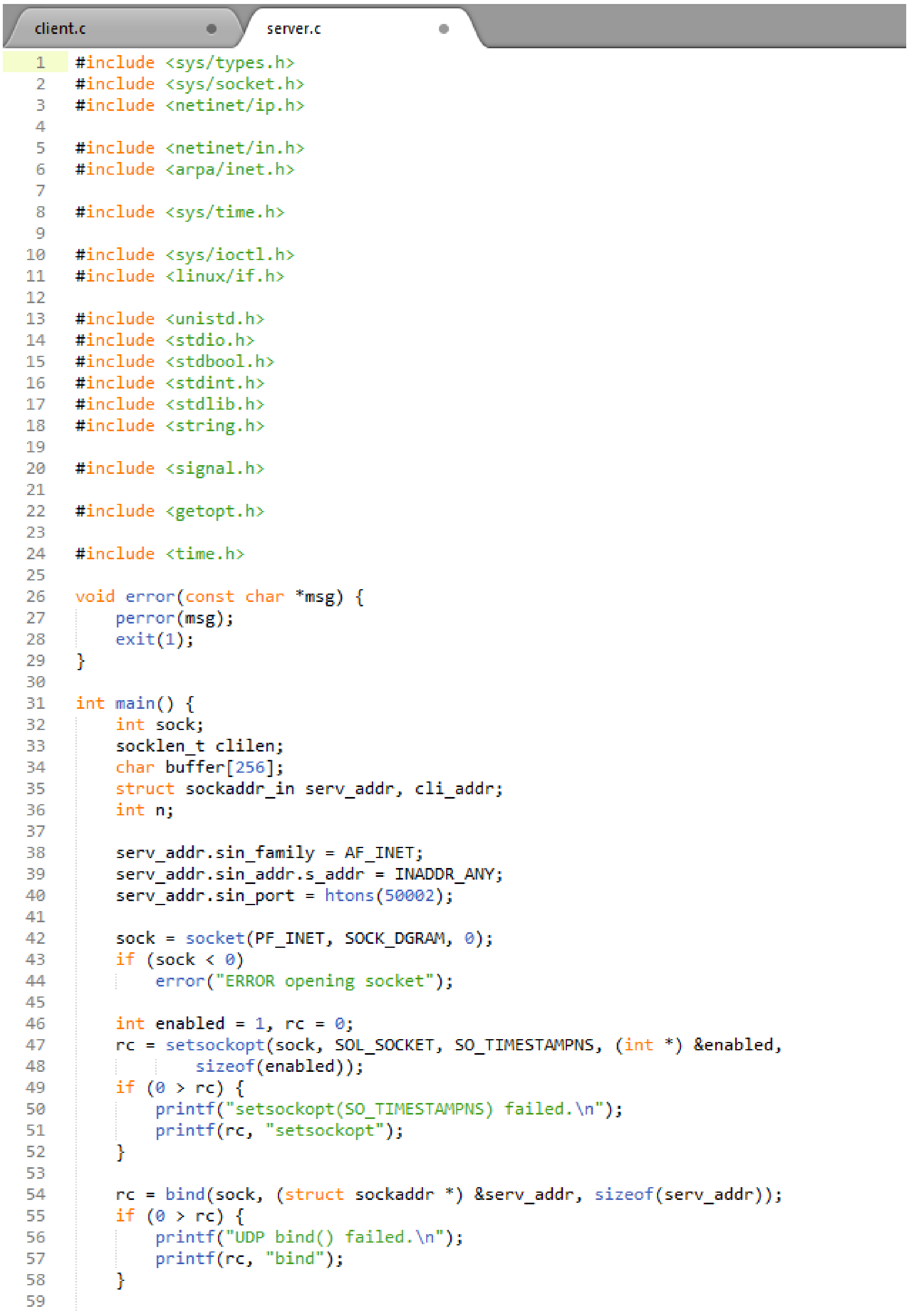
Fig. B. 1 SENSEFUL showcase scenario

APPENDIX C. SYNCHRONISATION MEASUREMENT

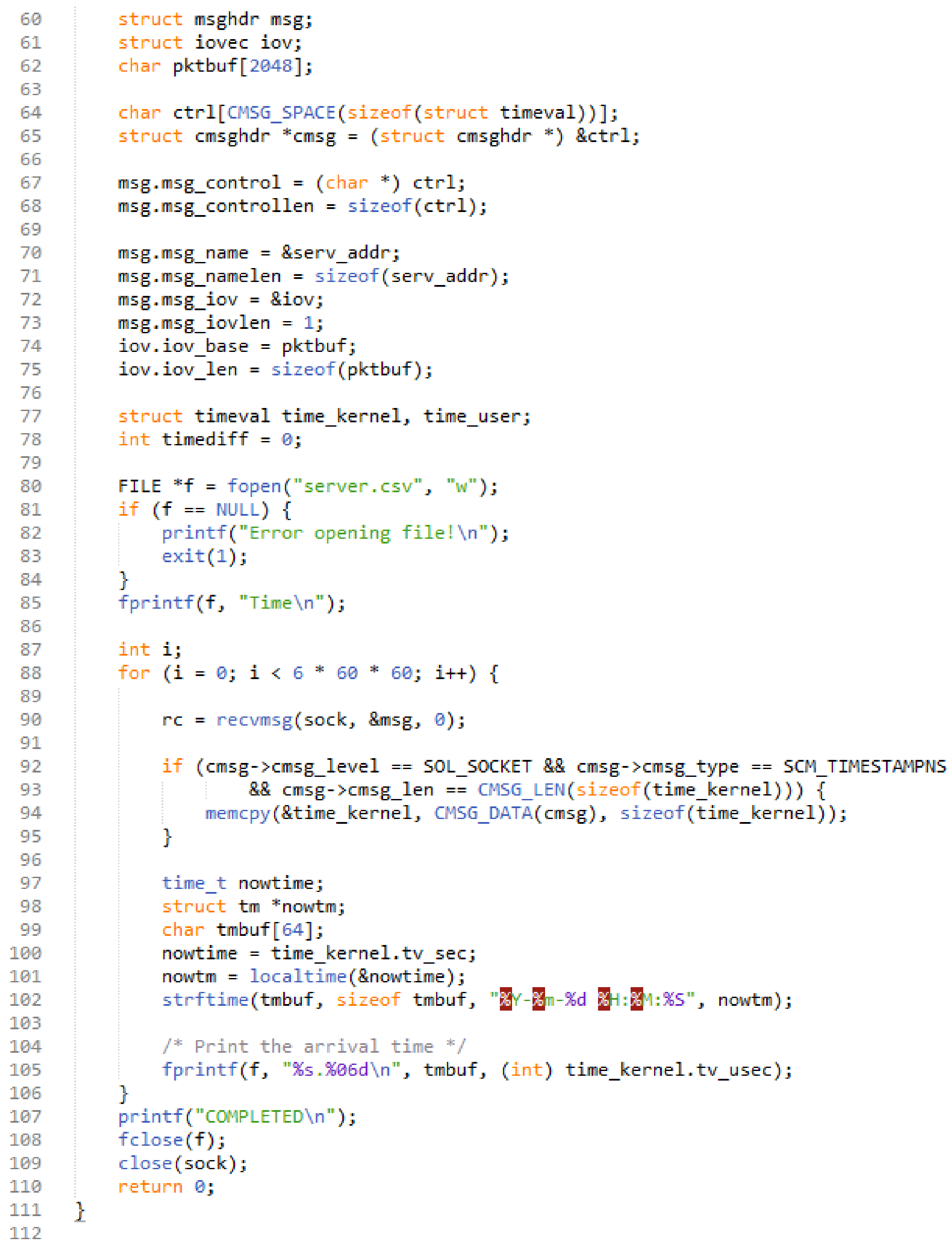


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <netdb.h>

void error(const char *msg) {
    perror(msg);
    exit(0);
}

int main(int argc, char *argv[]) {
    //printf("ARGS: %d\n", argc);
    if (argc < 2)
        error("Add the @IP argument");

    struct in_addr destAddr;
    memset(&destAddr, 0, sizeof(struct in_addr));

    inet_pton(AF_INET, argv[1], &destAddr);

    uint16_t destPort = 50002;

    /* Data packet to send. */
    uint8_t dataPacket[32];

    /* Destination ok. */
    int destSocket = socket(PF_INET, SOCK_DGRAM, 0);
    if (destSocket < 0) {
        printf("destSocket not opened.\n");
        return -1;
    }

    int broadcastEnable = 1;
    int ret = setsockopt(destSocket, SOL_SOCKET, SO_BROADCAST, &broadcastEnable,
            sizeof(broadcastEnable));

    struct sockaddr_in destSocketAddr;
    destSocketAddr.sin_family = AF_INET;
    destSocketAddr.sin_addr = destAddr;
    destSocketAddr.sin_port = htons(destPort);

    int i;
    for (i = 0; i < 6 * 60 * 60; i++) {
        if (0
                > sendto(destSocket, &dataPacket, sizeof(dataPacket), 0,
                        (struct sockaddr *) &destSocketAddr,
                        sizeof(destSocketAddr))) {
            perror("sendto");
        }
        usleep(1000 * 1000); //Sleep (us)
    }

    /* Cleanup */
    printf("Shutting %s down.\n", argv[0]);
    close(destSocket);

    return 0;
}
```

Fig. C. 1 Client script



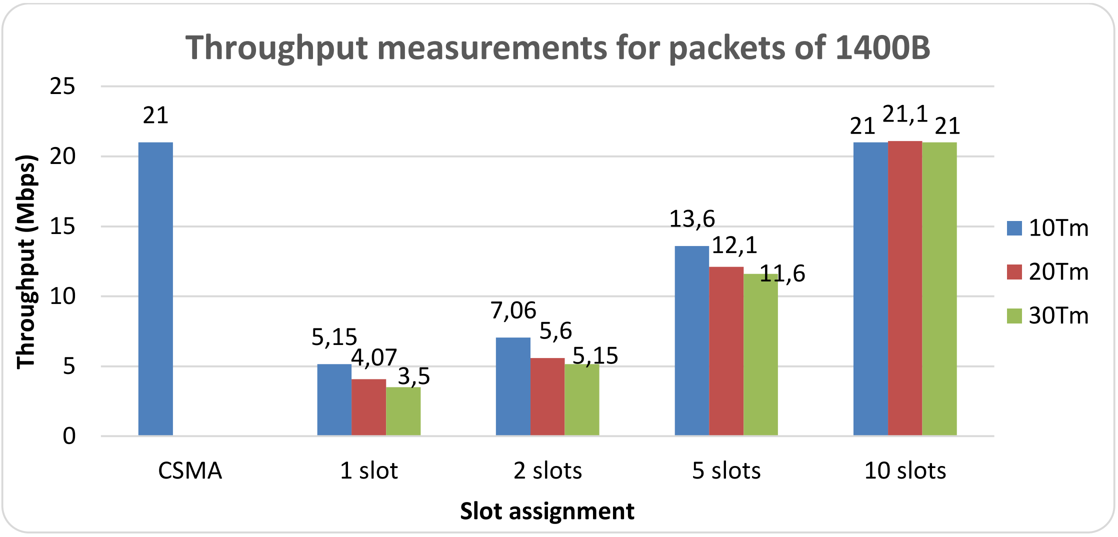
```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/ip.h>

#include <netinet/in.h>
#include <arpa/inet.h>

#include <sys/time.h>

#include <sys/ioctl.h>
#include <linux/if.h>

#include <unistd.h>
#include <stdio.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>

#include <signal.h>

#include <getopt.h>

#include <time.h>

void error(const char *msg) {
    perror(msg);
    exit(1);
}

int main() {
    int sock;
    socklen_t clilen;
    char buffer[256];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(50002);

    sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (sock < 0)
        error("ERROR opening socket");

    int enabled = 1, rc = 0;
    rc = setsockopt(sock, SOL_SOCKET, SO_TIMESTAMPNS, (int *) &enabled,
            sizeof(enabled));
    if (0 > rc) {
        printf("setsockopt(SO_TIMESTAMPNS) failed.\n");
        printf(rc, "setsockopt");
    }

    rc = bind(sock, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
    if (0 > rc) {
        printf("UDP bind() failed.\n");
        printf(rc, "bind");
    }

```

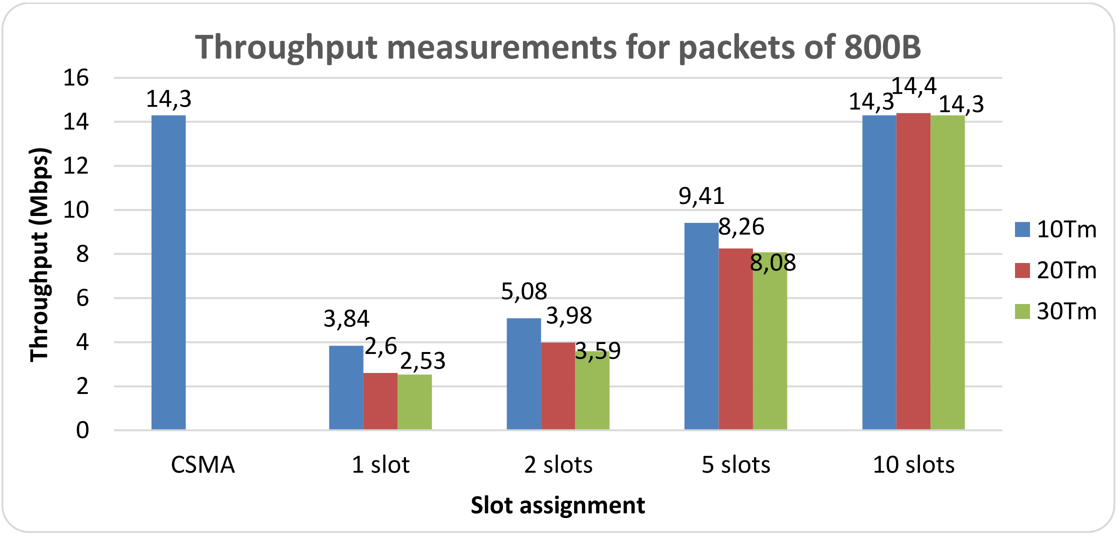


```c
    struct msghdr msg;
    struct iovec iov;
    char pktbuf[2048];

    char ctrl[CMSG_SPACE(sizeof(struct timeval))];
    struct cmsghdr *cmsg = (struct cmsghdr *) &ctrl;

    msg.msg_control = (char *) ctrl;
    msg.msg_controllen = sizeof(ctrl);

    msg.msg_name = &serv_addr;
    msg.msg_namelen = sizeof(serv_addr);
    msg.msg_iov = &iov;
    msg.msg_iovlen = 1;
    iov.iov_base = pktbuf;
    iov.iov_len = sizeof(pktbuf);

    struct timeval time_kernel, time_user;
    int timediff = 0;

    FILE *f = fopen("server.csv", "w");
    if (f == NULL) {
        printf("Error opening file!\n");
        exit(1);
    }
    fprintf(f, "Time\n");

    int i;
    for (i = 0; i < 6 * 60 * 60; i++) {

        rc = recvmsg(sock, &msg, 0);

        if (cmsg->cmsg_level == SOL_SOCKET && cmsg->cmsg_type == SCM_TIMESTAMPNS
                && cmsg->cmsg_len == CMSG_LEN(sizeof(time_kernel))) {
            memcpy(&time_kernel, CMSG_DATA(cmsg), sizeof(time_kernel));
        }

        time_t nowtime;
        struct tm *nowtm;
        char tmbuf[64];
        nowtime = time_kernel.tv_sec;
        nowtm = localtime(&nowtime);
        strftime(tmbuf, sizeof tmbuf, "%Y-%m-%d %H:%M:%S", nowtm);

        /* Print the arrival time */
        fprintf(f, "%s.%06d\n", tmbuf, (int) time_kernel.tv_usec);
    }
    printf("COMPLETED\n");
    fclose(f);
    close(sock);
    return 0;
}
```

Fig. C. 2 Server script

APPENDIX D. HTDMA PERFORMANCE EVALUATION





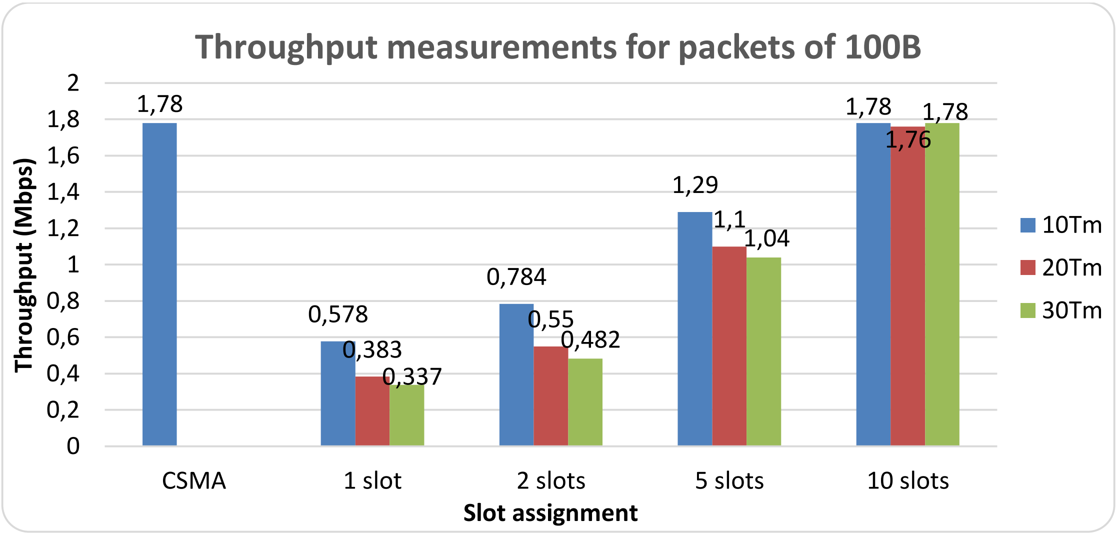
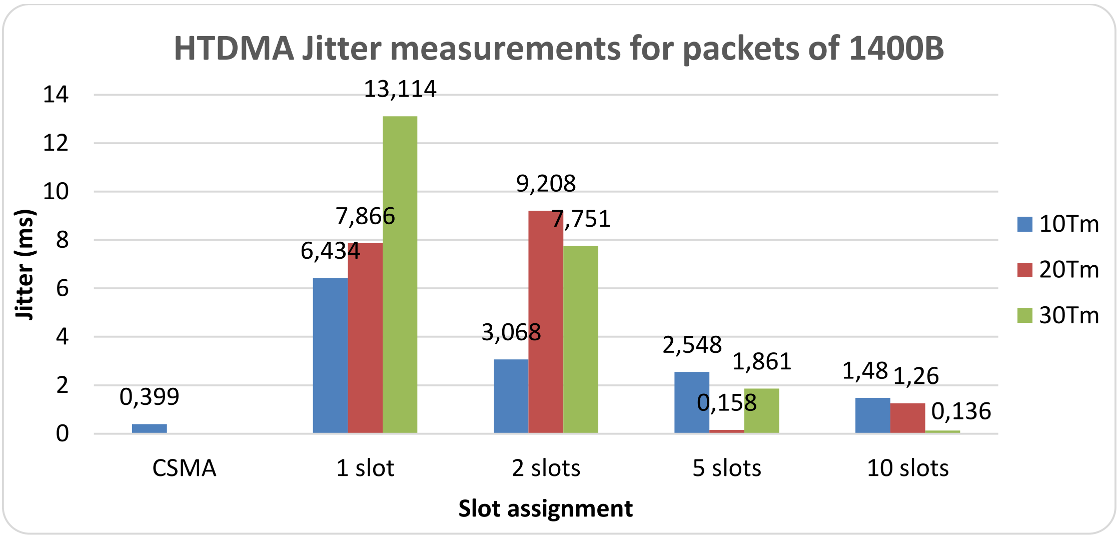
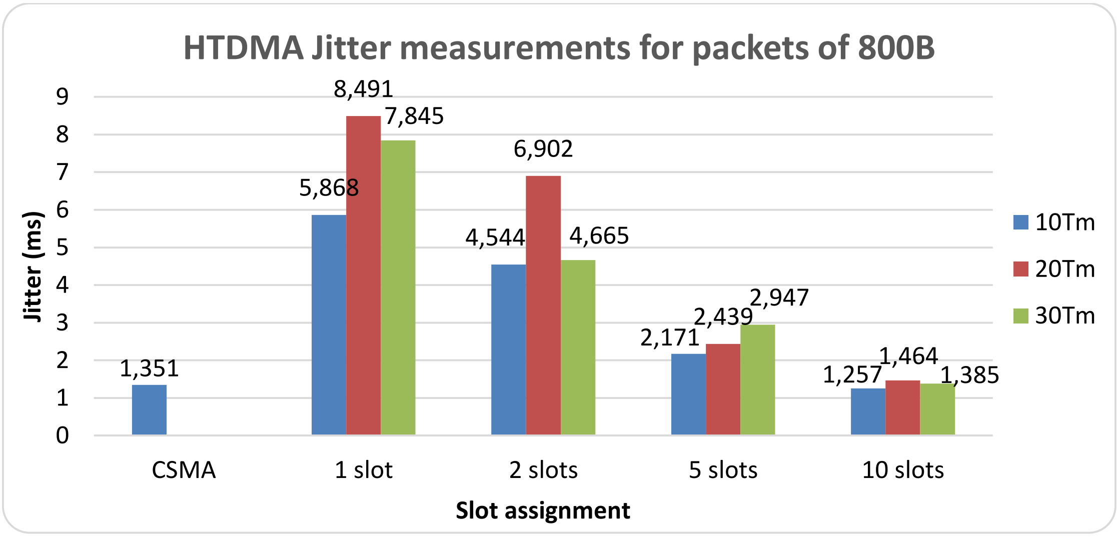


Fig. D. 1 Throughput measurements





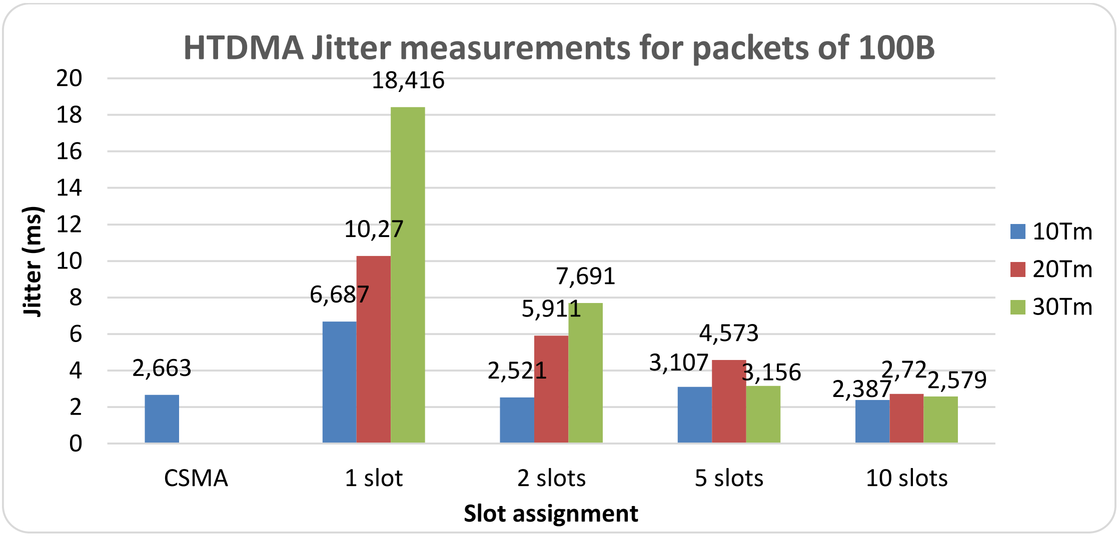


Fig. D. 2 Jitter measurements