Contributions to QoS and Energy Efficiency in Wi-Fi networks

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Daniel Camps Mur

Under the supervision of:
Prof. Sebastià Sallent Ribes

Telematics Department
Polytechnic University of Catalonia (UPC)

Campus Nord, Edici C3
c/ Jordi Girona, 1-3
08034 Barcelona, Spain.

September 2011

Abstract

The Wi-Fi technology has been in the recent years fostering the proliferation of attractive mobile computing devices with broadband capabilities. Current Wi-Fi radios though severely impact the battery duration of these devices thus limiting their potential applications.

In this thesis we present a set of contributions that address the challenge of increasing energy efficiency in Wi-Fi networks. In particular, we consider the problem of how to optimize the trade-off between performance and energy efficiency in a wide variety of use cases and applications. In this context, we introduce novel energy efficient algorithms for real-time and data applications, for distributed and centralized Wi-Fi QoS and power saving protocols and for Wi-Fi stations and Access Points.

In addition, the different algorithms presented in this thesis adhere to the following design guidelines: i) they are implemented entirely at layer two, and can hence be easily re-used in any device with a Wi-Fi interface, ii) they do not require modifications to current 802.11 standards, and can hence be readily deployed in existing Wi-Fi devices, and iii) whenever possible they favor client side solutions, and hence mobile computing devices implementing them can benefit from an increased energy efficiency regardless of the Access Point they connect to. Each of our proposed algorithms is thoroughly evaluated by means of both theoretical analysis and packet level simulations. Thus, the contributions presented in this thesis provide a realistic set of tools to improve energy efficiency in current Wi-Fi networks.

Dedicat a la Miriam, la Lídia, la Teresa i en Joan.

Acknowledgements

Reaching the end of this journey has not been an easy task, and of course it would not have been possible without the help and support of several people to whom I want to express my heartfelt gratitude. I would like to start thanking my professor Sebastià Sallent for his continuous support, his guidance and for countless technical discussions. I also want to thank the NEC Wi-Fi team, specially Youko Omori, for funding and continuously supporting this work.

Big thanks go to Xavier Pérez Costa, for showing me how to work in research and for many years of joint and very interesting work. I also thank Vladimir Marchenko and Manil Gomony who with their excellent master thesis contributed to the work presented in this thesis.

Finally, I thank my family, Lídia, Teresa and Joan, for their love and unconditional support, and I specially thank you Miriam, for being always there. It is much easier to do a PhD. having you in my life.

Contents

1 Introduction and Motivation **1**

1.1 A widening energy gap in mobile computing 1

1.2 The impact of Wi-Fi in mobile computing 2

1.3 Contributions and structure of this thesis 7

2 State of the Art **9**

2.1 Wi-Fi fundamentals 9

2.2 Literature survey 21

2.3 Goals and design guidelines of this thesis 26

3 An Adaptive Triggering Algorithm for Distributed Wi-Fi Power Saving Modes **29**

3.1 Modeling the effect of distributed power saving mechanisms on applications with QoS requirements 31

3.2 Algorithm design 36

3.3 Algorithm analysis 42

3.4 Performance evaluation 52

3.5 Conclusions 64

Appendix 3.A Convergence conditions with uplink triggers 66

4 Leveraging 802.11n Frame Aggregation to enhance QoS and Power Consumption in Wi-Fi networks **67**

4.1 Effect of 802.11n frame aggregation on Wi-Fi QoS and power saving protocols 68

4.2 CA-DFA: Congestion Aware-Delayed Frame Aggregation 76

4.3 CA-DFA performance evaluation 91

4.4 Conclusions 94

5 On Centralized Schedulers for 802.11e WLANs: Distribution vs Grouping of Resources Allocation 97
5.1 Related work on centralized scheduling . . . 98
5.2 Distributed Resource Allocation (DRA) . . . 101
5.3 Grouping of Resource Allocations (GRA) . . . 111
5.4 Performance evaluation . . . 113
5.5 Summary and conclusions . . . 121
Appendix 5.A F function description . . . 122
Appendix 5.B An initial admission control algorithm based on DRA . . . 125

6 Enhancing the performance of TCP over Wi-Fi Power Saving Mechanisms 131
6.1 TCP performance over *Wi-Fi PSM* . . . 132
6.2 BA-TA: An adaptive triggering algorithm for TCP over Wi-Fi PSM . . . 147
6.3 BA-TA Performance Evaluation . . . 152
6.4 Conclusions . . . 161
Appendix 6.A BA-TA convergence analysis . . . 162

7 Designing Energy Efficient Access Points with Wi-Fi Direct 165
7.1 AP power management in Wi-Fi Direct . . . 166
7.2 ASPP: Adaptive Single Presence Period . . . 168
7.3 AMPP: Adaptive Multiple Presence Periods . . . 183
7.4 Summary and conclusions . . . 197
Appendix 7.A Revisiting the assumptions of the 3g bandwidth estimation algorithm . . . 199

8 Conclusions and Future Work 201

List of Figures 205

References 211

1

Introduction and Motivation

1.1 A widening energy gap in mobile computing

Last years have whitnessed a phenomenal growth in the area of mobile computing devices. Indeed, it is envisioned that in the near future a wide presence of broadband access technologies will foster even more the adoption of such devices, bringing to reality the vision of ubiquitous broadband connectivity. A fundamental problem though is to be solved for that vision to come true, that is to fulfill the increasing power demands that newer and faster technologies impose on mobile computing devices. Without the capability of powering light and attractive handheld devices for long periods of time while providing users with a full broadband experience, mobile broadband will struggle to reach its full potential.

Nowadays, batteries used in mobile computing devices are mostly based on the Lithium-Ion (Li-Ion) technology. This technology, although it did revolutionize mobile computing on its adoption in 1991, has not been able to keep pace with the growing demands of the systems embedded in portable electronics (1). This fact is illustrated in Figure 1.1 where the relative improvements since 1991 in energy density of Li-Ion batteries is compared to the relative performance improvements of other mobile computing components like hard drive capacity, CPU speed and the peak data rates of wireless networks. It is clear in the figure that while the relative performance of almost every single component has increased exponentially, the energy density of Li-Ion batteries remains flat, giving raise to an increasing *Energy Gap* between power demand and power delivery in mobile computing (1).

The future of battery technologies seems to point towards *nanostructured materials*, *fuel cells* and *thin film batteries* as promising technologies to eventually replace the Li-Ion technology. In addition, power coming from additional energy sources, like solar cells, might also come into play (7). However, due to a variety of factors, none of these technologies appears as a feasible replacement for Li-Ion in the near future and even when available they will not completely solve the energy problem (1). Therefore, efficient *power management protocols* and *energy oriented architectures* have arised as a mandatory path to follow in order to overcome the existent Energy Gap in mobile computing.

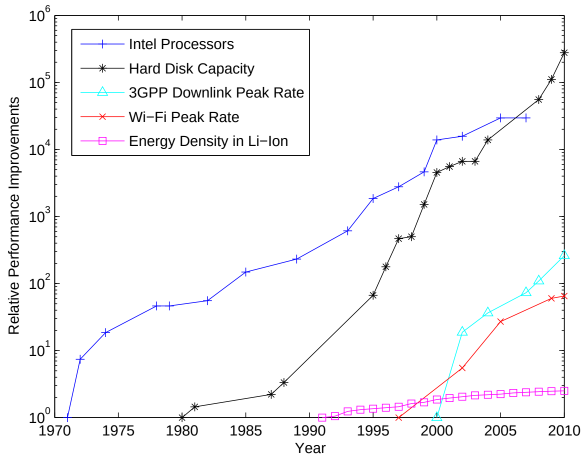


Figure 1.1: Widening Energy Gap in Mobile Computing. Data obtained from (2), (3), (4), (5) and (6).

1.2 The impact of Wi-Fi in mobile computing

One of the main trends driving the increase of this Energy Gap is the adoption in portable devices of multiple wireless technologies that have to be served using a unique power source. Among these technologies, 802.11 (9) or Wi-Fi (10)[1] has enjoyed a privileged position. For instance Figure 1.2 illustrates the forecasted growth of Wi-Fi in handset devices in the upcoming years. Other studies (8) also point out that by 2014 90% of smartphones will be shipped with embedded Wi-Fi.

Among the factors driving the adoption of Wi-Fi in mobile coputing devices are its high data rates which benefit multimedia and business applications, its low roaming costs and its good in-building coverage. Indeed, it is expected that in the near future Wi-Fi will be widely deployed inside the home or in the enterprise, for instance embedded into TV sets, set top boxes or projectors, hence creating opportunities to connect mobile computing devices like mobile phones or cameras with consumer electronics, leading to every time richer user experiences and new business opportunities for carriers. The recently developed Wi-Fi Direct technology (11) is a key step in this direction. In addition, carriers are also starting to heavily rely on Wi-Fi capabilities in smartphones in order to offload traffic from their heavily congested cellular networks into Wi-Fi Hotspots (12).

[1]In this thesis we refer indistincly to the technology described by the 802.11 standard as 802.11 or Wi-Fi.

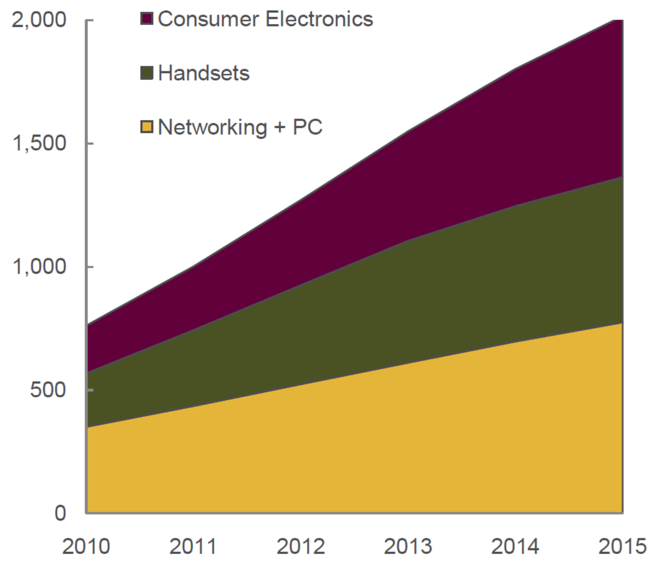


Figure 1.2: Wi-Fi IC Shipments by Device Type, In Millions. Source: ABI Research - 4Q 2010.

The Wi-Fi industry is actively working in order to consolidate and expand its position in the mobile computing arena. Among the major trends expected to steer this technology in the near future are increased data rates above 1Gbps, 802.11ad (13) and 802.11ac (14), the operation in new frequency bands like TV White Spaces, 802.11 af (15) or the 900Mhz ISM band, 802.11ah (16), and the adoption of Wi-Fi in sensor and metering devices with very stringent battery requirements, SmartGrid effort in Wi-Fi Alliance (67) or 802.11ah (16).

Notice that in order to successfully address the previously described use cases and markets, energy efficiency arises as a key factor in Wi-Fi.

1.2.1 How can we save power in Wi-Fi?

An immediate question arises when thinking about Wi-Fi and energy efficiency, and is which is the impact of a Wi-Fi radio in the battery duration of a mobile computing device.

The actual impact on the overall power consumption of a device depends of course on a variety of factors, for instance the presence of other power hungry subsystems like displays. Several studies have been carried out in the literature to get a closer grasp on what this impact actually is. For instance Wi-Fi interfaces have been shown to account for about 10% of the total energy consumption in current laptops (19), up to 50% in hand-held devices and even higher percentages in smaller form-factor prototypes (19, 20). An interesting study is presented in (18), where the battery discharging profile of a modern

HTC Hero smartphone (17) is studied with the Wi-Fi radio turned on and off, and with different Wi-Fi configurations. The obtained results are depicted in Figure 1.3.

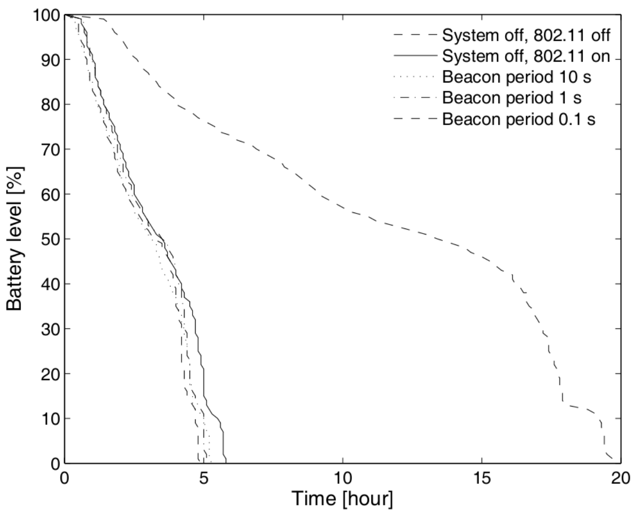


Figure 1.3: Impact of Wi-Fi on battery life. Effect of the Wi-Fi radio on the battery duration of an HTC Hero smartphone.

We can see in Figure 1.3 how the battery duration of the HTC Hero smartphone goes down from about 20 hours to around 5 hours (75% reduction) when the Wi-Fi interface is kept always on[1]. Interestingly, different configurations in the Wi-Fi radio, in particular different Beacon rates that result in different number of transmissions per second, have only a marginal effect on battery duration.

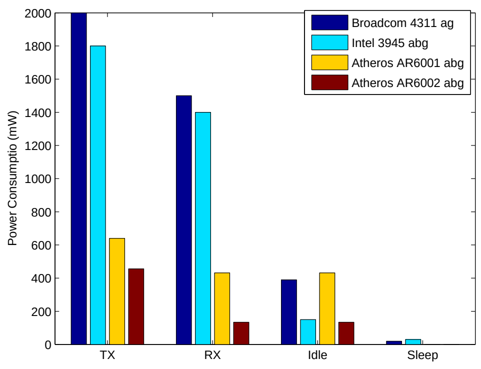
The actual power drained by a Wi-Fi interface depends on the particular activity being carried out by a device. For instance more power is consumed when performing a data transfer than when being in idle mode. In (21) it has been shown that the power spent by a Wi-Fi interface can be safely classified into four different states:

- *Transmit*: The Wi-Fi interface modulates and transmits packets over the air. Usually Wi-Fi transmitters operate with transmit powers around 15dBm.
- *Receive*: The Wi-Fi interface is receiving and demodulating data and passing it to the host CPU, which will then perform further operations like forwarding the received data to the higher layers.
- *Listen*: The Wi-Fi interface is listening to the channel, potentially also receiving data but not forwarding it to the host CPU.

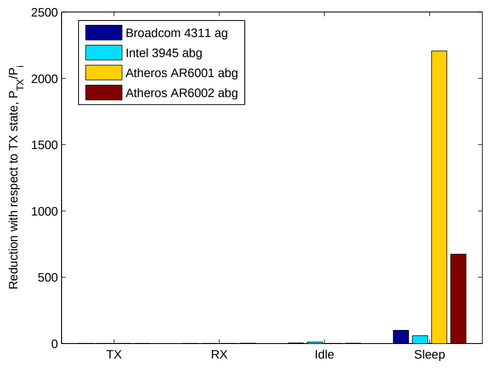
[1]In order to achieve a fair comparison, the display of the device was always on in all these tests.

- *Sleep*: The Wi-Fi interface switches off the majority of its circuitry except certain critical parts. Very small power is consumed by the Wi-Fi interface in this state, but the interface can not transmit, receive or listen to the channel.

Figure 1.4(a) illustrates the power consumed by several popular Wi-Fi chipsets in each of the previous states. In order to reduce the power consumed by a Wi-Fi interface, a first possibility is to reduce the power spent by the wireless interface in each of the previous states, as has been done in the Atheros chipsets depicted in Figure 1.4(a). This can be achieved by means of technological improvements like using more efficient RF power amplifiers, or minimizing the interactions between the Wi-Fi interface and the host CPU[1]. A more detailed description of these technological improvements can be found in (22).



(a) Power consumed by typical Wi-Fi chipsets.



(b) Relative power consumption reduction of each state with respect to the Transmit state.

Figure 1.4: Power Consumption Figures in current Wi-Fi chipsets. Data from (23), (24), (25).

A second approach to reduce power consumption, which complements the first one, is to design power management protocols that switch the wireless interface to the *Sleep* state whenever no transmissions or receptions are required. Figure 1.4(b) depicts for the previous Wi-Fi chipsets, the power consumption reduction achieved in each of the power states with respect to the *Transmit* state. As clearly shown in the figure, the maximum power reduction is obtained when the Wi-Fi radio switches to the *Sleep* state.

Therefore IEEE 802.11 defined power management protocols that rely on the fact that the power consumed by a Wi-Fi interface in the Sleep state is several orders of magnitude lower than the power

[1]For instance processing management frames like Beacons directly in the Wi-Fi chipset, not in the host CPU.

consumed in any other state, and so try to switch the Wi-Fi radio to a Sleep state as much as possible. In a nutshell the core idea behind these protocols, is to let stations sleep while the Acess Point buffers incoming data for the sleeping stations, until a station decides to wake up and trigger the Access Point for its buffered data[1]. In addition, switching to and from the Sleep state can be implemented in current chipsets with very small switching times as further illustrated in Figure 1.5. The previous technological facts open the door to the design of efficient power management protocols in Wi-Fi that can reduce the impact of the technology on the overall power consumption of a mobile computing device.

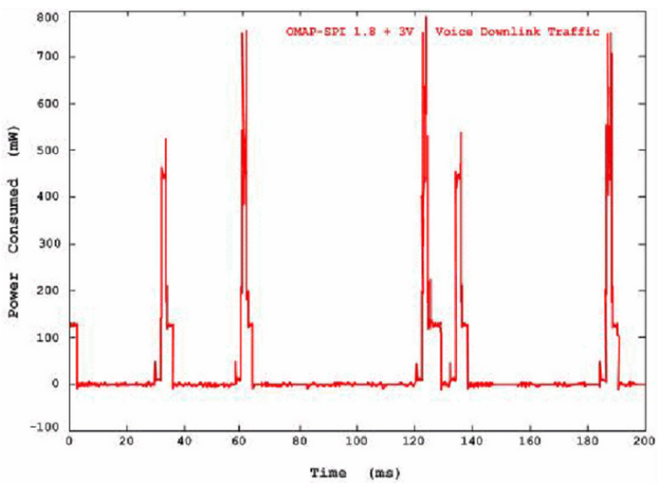


Figure 1.5: Power vs Time during a VoIP call over Wi-Fi. Notice that the different levels depicted in the power consumption profile represent the power spent by the chipset in the sleep, idle, receive and transmit states. Graphic from (22).

Implementing power management protocols though does not come without a penalty. While switching off a Wi-Fi interface reduces power consumption, it also introduces a negative impact on performance or Quality of Service (QoS) of the applications running on the mobile computing device. The reason is that a device in the Sleep state can not receive any transmission, therefore packets transmitted to this device will either be lost or will have to be delayed until the sleeping device comes back to an active state. Power management protocols and algorithms are needed that can successfully manage the trade off between the level of performance or QoS and the power consumption of a device. The fundamental contribution of this thesis, is a set of algorithms that attempt to optimize the previous performance energy trade-off, while considering a variety of applications and Wi-Fi technologies.

[1]These protocols will be described in detail in Chapter 2

1.3 Contributions and structure of this thesis

This thesis is structured in the following way. Chapter 2 reviews the state of the art in the area of energy efficiency in Wi-Fi networks. This chapter describes the Wi-Fi technology fundamentals relevant to the work of this thesis, and presents a taxonomy to classify the contibutions in the literature aimed at improving energy efficiency in Wi-Fi.

Chapters 3, 4, 5, 6, 7, contain the major contributions of this thesis, which can be classified in three different categories: i) energy efficiency with real-time applications, ii) energy efficiency with data traffic, and iii) energy efficient Wi-Fi Access Points.

Our first three contributions address the QoS energy trade-off of real-time applications, our fouth contribution deals with energy efficiency with data traffic and our fifth contribution relates to energy efficient Access Points. These contributions are described as follows:

- ***Contribution 1***. Our first contribution described in Chapter 3 studies how the trigger interval used by a station in power save mode affects the QoS of real-time applications like Voice and Video. In addition, based upon the obtained insights, we present the design of an adaptive layer two algorithm, that runs in a Wi-Fi station and adapts the mentioned trigger interval in order to fulfill the QoS requirements of real-time applications in an energy efficient way. The content of Chapter 3 has been published in (26).

- ***Contribution 2***. Our second contribution described in Chapter 4 considers the same scenario as our first contribution but addresses a different question. With the advent of efficient frame aggregation techniques in 802.11n, energywise it may be beneficial for a Wi-Fi station in power saving to buffer several frames and transmit them using an efficient aggregate. The previous though will increase delay and therefore may jeopardize the QoS of real-time applications. Therefore, our second contribution is a dynamic layer two algorithm that selects the optimum aggregation interval according to the level of congestion in the network. The content of Chapter 4 has been submitted to (27).

- ***Contribution 3***. Unlike our previous two contributions, our third contribution described in Chapter 5 addresses the QoS and energy efficiency trade off of real-time applications but considering the centralized protocols defined in 802.11, i.e. HCCA and S-APSD, that will be thoroughly described in Chapter 2. In this context we propose a novel scheduling algorithm that runs in the Access Point, and is based on the principle of *spreading* in the channel the service periods of the different stations, instead of grouping them as is commonly done in the state of the art. The content of Chapter 5 has been published in (28).

- ***Contribution 4***, described in Chapter 6 addresses the performance energy trade-off of Data traffic in Wi-Fi. In particular, we study the detailed interactions between TCP and the Wi-Fi power saving protocols, and identify that the trigger interval used by a Wi-Fi station should be adapted according to the bottleneck bandwidth experienced by TCP connections. We propose an adaptive layer two algorithm that implements the previous principle and evaluate its performance both with popular data applications. The content of Chapter 6 has been submitted to (29).
- ***Contribution 5***, is described in Chapter 7, where we propose a set of Layer two algorithms to manage the AP power saving protocols defined in Wi-Fi Direct, which are essential if battery limited devices like mobile phones are to implement the Wi-Fi Direct technology. The content of Chapter 7 has been published in (30).

Finally, chapter 8, concludes this thesis and highlights lines of future work.

2

State of the Art

The purpose of this chapter is threefold. First, we describe the Wi-Fi protocols that are the basis of the work presented in this thesis. Second we survey the state of the art on energy efficiency in Wi-Fi and present a taxonomy to classify existent work. Finally, we introduce the design guidelines of this thesis and position our contributions with respect to the work already existing in the state of the art.

2.1 Wi-Fi fundamentals

The today's widely known *Wi-Fi technology* is the result of the standardization work done in the IEEE 802.11 Task Group (31). This group was chartered to design a Wireless Local Area Network, that would allow a set of devices to wirelessly connect to each other and share resources, like Internet access. Thus, the IEEE 802.11 group came up with two distinct architectures for this purpose which are depicted in Figure 2.1: i) the *Basic Service Set* (BSS) which allows a set of *stations* to communicate through an *Access Point* (AP), and ii) the *Independent Basic Service Set* (IBSS), which allows stations to connect directly to each other.

The BSS architecture has been the one enjoying the widest adoption in the market and will be the one considered in this thesis. This architecture offers significant advantages, like the fact that an AP is a central node in range of all its associated stations and can be used to mitigate *hidden node* problems (33), or the fact that centralized support in the AP can be used to improve network discovery and to assist the power saving operation of the connected stations.

Another relevant part of the Wi-Fi technology, is its channel access protocol. In particular the IEEE 802.11 group adopted a *Listen Before Talk* protocol similar to the one used in Ethernet (34) to arbitrate access to the channel. There is a remarkable difference with the Ethernet protocol though, due to the

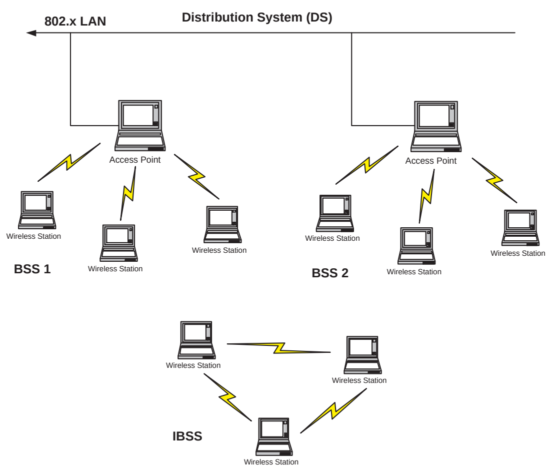


Figure 2.1: Wi-Fi Architecture. In the BSS architecture all communications go through the AP. In the IBSS architecture stations can directly communicate with each other.

fact that wireless stations can not instantly detect collisions. Therefore, a modified protocol was defined known as *Carrier Sense Multiple Access / Collision Avoidance* (CSMA/CA), where stations always acknowledge received frames in order to inform the sender that the frame was received correctly. In order to resolve collisions a binary exponential backoff algorithm was included in the protocol (9).

2.1.1 802.11e: QoS on Wi-Fi

It was soon realized that *over the air* Quality of Service (QoS) was necessary in Wi-Fi. The reason is that Wi-Fi is a shared medium, therefore if all frames are considered equal over the air, a station with low priority traffic could always delay transmissions of high priority traffic. In order to address this concern, the IEEE chartered the 802.11e group with the goal of adding QoS functionalities to the basic Wi-Fi specification.

The result of the work of the 802.11e group was a new ammendment to the 802.11 specification (32) that among other enhancements, defined two new channel access modes with built-in QoS. The first, of these channel access modes was a *Distributed* mode called *Enhanced Distributed Channel Access* (EDCA), and the second one was *Centralized* mode, known as the *Hybrid Coordination Function (HCF) Controlled Channel Access* (HCCA). These two modes of operation are described next.

2.1.1.1 Distributed mode: EDCA

The Enhanced Distributed Channel Access (EDCA) provides prioritized QoS by extending the basic CSMA/CA algorithm defined in Wi-Fi. In particular, EDCA introduces the concept of Access Categories (AC), which represent different priorities to access the wireless channel. Thus, each EDCA device consists of four ACs, namely AC_VO, AC_VI, AC_BE and AC_BK, which run concurrent back-off instances within the device. This architecture is depicted in Figure 2.2, where it is also illustrated how frames coming from the higher layers can be classified within the four defined ACs.

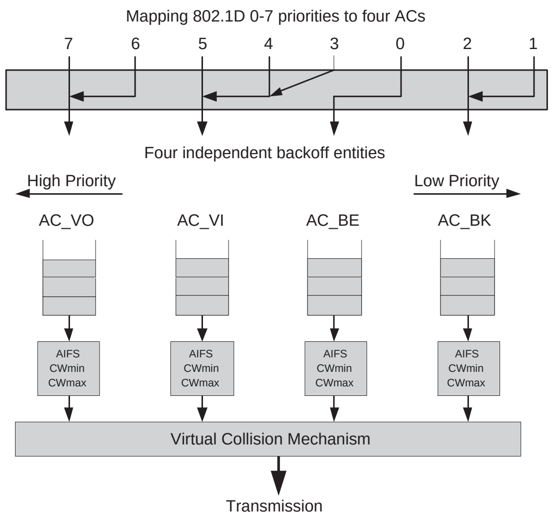


Figure 2.2: Concurrent Backoff Entities in EDCA. Each EDCA device, AP and stations, runs four concurrent Access Categories.

In order to implement the necessary over the air prioritization, EDCA assigns different contention parameters to different Access Categories (ACs). Thus, ACs with smaller contention parameters require shorter times to access the channel, and are hence provided *statistical* guarantees over traffic transmitted from ACs that use bigger contention parameters. The contention paramaters used by stations in EDCA, are dictated by the AP which periodically broadcasts them within the Beacon frame. Figure 2.3 illustrates how over the air priorities can be implemented by using differentiated contention parameters. Another interesting innovation introduced in EDCA is the use of Transmission Opportunities (TXOPs), which allow an EDCA Access Category to hold control of the channel during a certain time, TXOP Limit, in order to transmit multiple frames with a single channel access. The TXOP Limit is another parameter that can be used to provide differentiated service over EDCA. For more details on EDCA the interested reader is referred to (57).

Finally, it is worth to notice that the EDCA technology is currently certified and tested by the Wi-Fi Alliance (10) in the WMM certification (35).

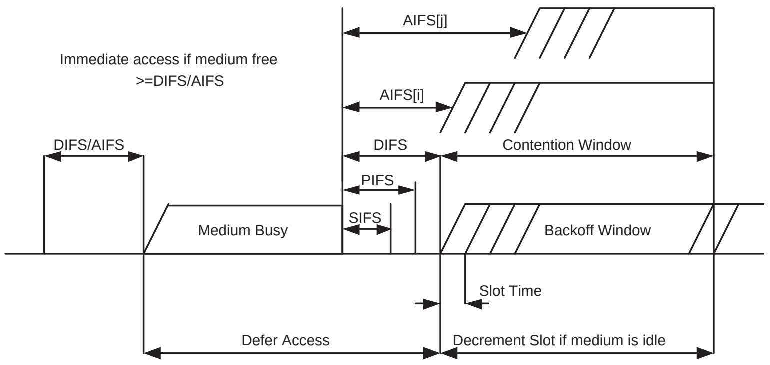


Figure 2.3: Contention Parameters of each AC. Higher priority ACs use smaller contention parameters, i.e. AIFS and Contention Windows, and hence statistically gain earlier access to the channel.

2.1.1.2 Centralized mode: HCCA

The centralized HCF Controlled Channel Access (HCCA) operation is possible in two different ways: the Contention Free Period (CFP), where the Hybrid Coordinator (HC), usually residing in the Access Point (AP), reserves the channel only for HCCA traffic, and the Contention Period (CP) where both EDCA and HCCA traffic can access the channel. In the Contention Period a higher priority is granted to HCCA traffic by allowing the AP to use a shorter inter frame space to access the medium than contention based traffic, the Priority Inter Frame Space (PIFS) illustrated in Figure 2.3.

Within the context of HCCA, MAC Service Data Units (MSDUs) are associated to flows and flows are associated to stations. In order to set up a new flow, a station has to submit a Traffic Specification (TSPEC) message to the AP. In the TSPEC message the station describes its traffic flow[1], and specifies the delay bound (maximum time before which a MSDU belonging to this flow has to be successfully delivered) and the radio conditions[2] under which this flow is expected to operate. Thus, an Admission Control function and a Scheduling function are implemented in the AP in order to guarantee the QoS requirements of the admitted flows.

Figure 2.4 illustrates the operation of HCCA both in the Contention Free Period and in the Contention Period. Notice in the figure how the AP explicitly polls associated stations, using a QoS CF-Poll frame, in order to grant them access to the channel, or directly transmits to them in downlink.

[1]The flow descriptor specified in RFC 2212 (111) is used to define a flow.

[2]Radio conditions are conveyed with two parameters: i) Modulation and Coding Scheme (MCS) and ii) Surplus Bandwidth Allowance (SBA) (which accounts for possible retransmissions).

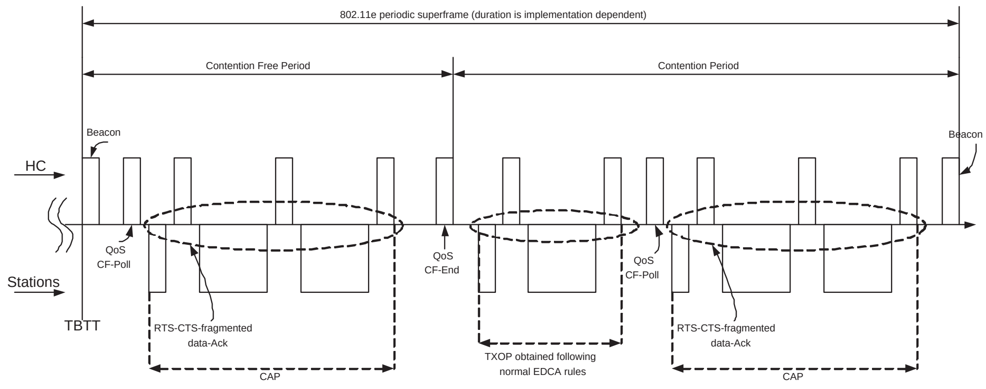


Figure 2.4: Sample of HCCA operation during the CFP and the CP. A Controlled Access Phase (CAP), defines a channel access granted by the Hybrid Coordinator.

2.1.2 802.11n: The next generation of Wi-Fi

Orthogonally to the previously mentioned enhancements, the IEEE 802.11 group recently developed the 802.11n standard (36) which defines improvements to both physical and MAC layers, and is meant to replace the traditional 802.11a/b/g technologies in order to become the baseline technology in next generation WLANs (69).

In particular, specially relevant to the work in this thesis are the aggregation schemes defined in 802.11n: A-MSDU and A-MPDU. These two schemes substantially reduce the overhead introduced by Wi-Fi by allowing a station or AP to aggregate several data frames within a single physical frame, and transmit all of them with a single access to the channel. In particular, the A-MSDU scheme allows to aggregate several *MAC Service Data Units* (MSDUs) coming from the higher layers, into a single A-MSDU and transmit them with a single MAC and Physical header. This scheme provides maximum aggregation efficiency, but has several disadvantages stemming from the fact of considering only one MAC header. For instance, all aggregated frames must be addressed to the same station and must carry the same User Priority (UP) value. In addition this scheme contains only one CRC to protect the whole aggregated frame, which can be counter productive since packet error rates typically increase with the length of the frame (84). Some of the previous drawbacks are addressed in the A-MPDU scheme which creates an aggregate of *MAC Protocol Data Units* (MPDUs), where each independent frame contains its individual MAC header and CRC. However, the A-MPDU scheme is still constrained to aggregate only MPDUs addressed to the same receiver. Figure 2.5 illustrates the two aggregation schemes supported in 802.11n. In addition, a thorough overview of 802.11n MAC aggregation mechanisms can be found in (70).

Besides novel aggregation schemes, 802.11n also defines a full set of throughput enhancements by means of techniques like MIMO or channel bonding. The interested reader is referred to (37) for an overview of the different features introduced in 802.11n.

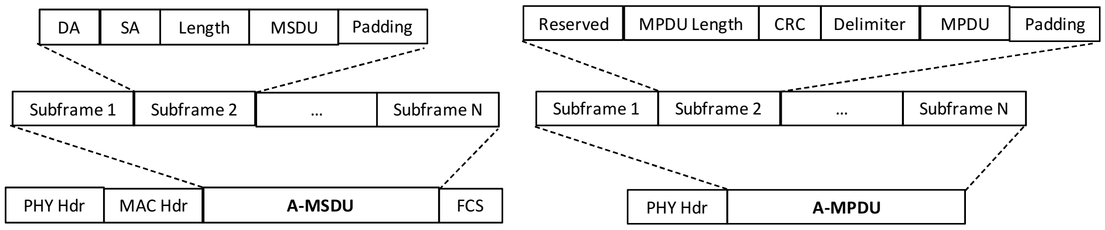


Figure 2.5: Overview of the A-MSDU and the A-MPDU aggregation schemes in 802.11n,. the A-MSDU scheme is depicted on the left and the A-MPDU scheme on the right.

2.1.3 Power saving protocols in 802.11

The 802.11 group soon realized about the need to build in the protocol mechanisms to reduce power consumption, and so it already included a power saving mechanism in the first version of the standard published in 1999 (9). This protocol will be hereafter referred as *Standard Power Save Mode* (StdPSM) or *802.11 legacy power save mode* (802.11 PSM). However, this basic protocol exhibited several problems specially with respect to real time communications, like Voice or Video. Therefore, the 802.11e standard (32), chartered with the goal of adding QoS capabilities to the original 802.11 standard, defined a new power saving protocol, namely *Automatic Power Save Delivery (APSD)*, better suited for real-time communications. These protocols are described next.

2.1.3.1 Standard Power Save Mode

In infrastructure mode (BSS), the power-management mechanism is centralized at the Access Point (AP). APs maintain a power-management status for each currently associated station that indicates in which power-management mode the station currently operates. Stations changing the power-management mode inform the AP by using the power-management bit within the frame control field of the transmitted frames. The AP buffers unicast and multicast data frames destined for any of its associated stations in power save mode. If an AP has buffered frames for a station, it will indicate so in the Traffic Indication Map (TIM), which is sent with each Beacon frame, typically every 100ms.

Stations request the delivery of their buffered frames at the AP by sending a signaling frame called Power Save Poll (PS-Poll). A single buffered frame for a station in power save mode is sent by the AP after a PS-Poll has been received. In the frame control field of the frame sent in response to a PS-Poll, the AP indicates with the More Data bit if there are further frames buffered for this station. The station is required to send a PS-Poll to the AP for each data frame it receives with the More Data bit set. This ensures that stations empty the buffer of the frames held for them at the AP. Mobile stations should also awake at times determined by the AP, when broadcast/multicast (BC/MC) frames are to be transmitted. This time is indicated in the Beacon frame within the TIM element.

Finally, note that the StdPSM functionality does not imply that frames sent from the station to the AP are delayed until the next beacon is received, that is, stations wake up whenever they have data to send and follow the regular 802.11 transmission procedure. Figure 2.6 illustrates the operation of Standard Power Save Mode in infrastructure mode.

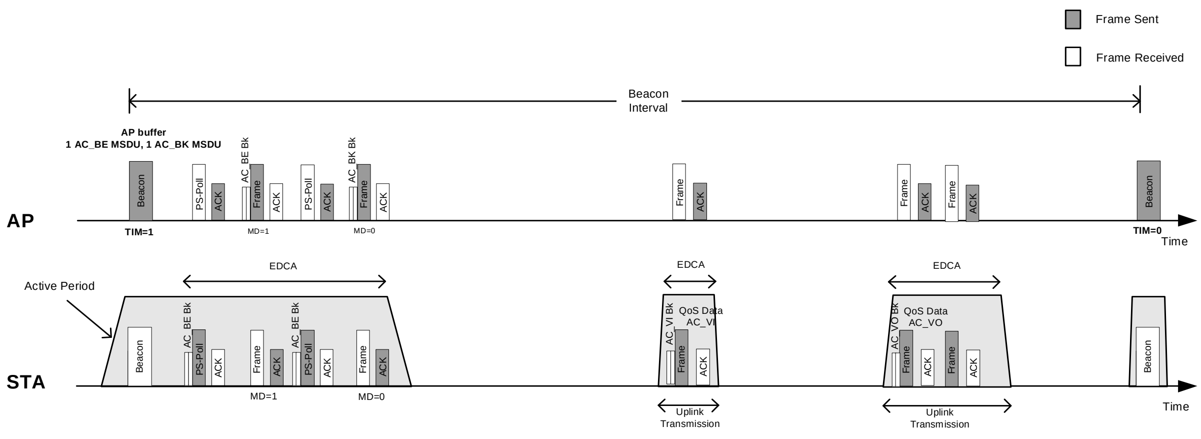


Figure 2.6: Example of Standard PSM operation. The station awakes to listen to every Beacon, Listen Interval=1, and discovers with the TIM bit if there is buffered data for it in the AP. If that is the case the station triggers the transmission of buffered frames using the PS-Poll frame. The station can awake at any time to perform uplink transmissions.

2.1.3.2 802.11e APSD

Automatic Power Save Delivery is the proposed 802.11e extension of the original Standard Power Save Mode protocol. In line with the 802.11e QoS mechanisms, which define the *distributed* Enhanced Distributed Channel Access (EDCA) access to provide *prioritized* QoS guarantees and the *centralized* HCF Control Channel Access (HCCA) access to provide *parameterized* QoS guarantees, APSD defines a distributed power saving scheme, *Unscheduled APSD* (U-APSD), and a centralized one, *Scheduled APSD* (S-APSD). Thus, frames buffered at the AP can be delivered to power saving stations either by using the EDCA access method if U-APSD is selected or by using EDCA or HCCA if S-APSD is chosen. The period of time where a station is awake receiving frames delivered by the AP is defined in APSD as a *Service Period* (SP). A SP is started by a station or an AP depending on the considered APSD mechanism and is always finished by the reception at the station of a frame with the *End Of Service Period* Flag set (EOSP). Next, we review the two APSD modes of operation, U-APSD and S-APSD.

Unscheduled-APSD (U-APSD). The main novel idea behind the U-APSD design is to proactively poll the AP to request buffer frames instead of waiting for notifications in the Beacon frame. Notice that proactive polling can reduce the latencies introduced in Standard Power Save Mode by the Beacon interval, usually 100ms, hence turning U-APSD into an attractive option for real-time communications. However, excessive polling requests from the stations to the AP can introduce congestion and degrade performance, therefore U-APSD was designed with the aim of minimizing the amount of required signaling. Specifically, the data frames sent in the uplink by stations (STA $\rightarrow$ AP) can be used as indications (*triggers*) of the instants when power saving stations are awake. When such an indication is received at the AP from a power saving station, the AP takes advantage of it and delivers data frames buffered while the station was in sleep mode. Because of this specific functionality, U-APSD is specially suited for bi-directional traffic streams even though it provides alternative methods for its usage in other cases.

In U-APSD, each Access Category (AC) of an EDCA station can be configured separately to be delivery/trigger-enabled. If one or more ACs of a station are *trigger-enabled*, when the AP receives a frame of subtype QoS Data or QoS Null of a trigger-enabled AC, a Service Period (SP) is started if one is not in progress. If a station has one or more ACs configured as *delivery-enabled*, when this station starts a SP the AP delivers the buffered frames corresponding to these ACs using EDCA. The configuration at the AP of the different ACs per station as delivery/trigger-enabled can be performed either at association time or through the usage of the Traffic Specification (TSPEC) message.

During a SP one or more data frames of delivery-enabled ACs might be delivered by the AP to a station up to the number of frames indicated in the *Maximum Service Period Length* (Max_SP_Length) value following the rules of an acquired transmission opportunity.

In the case of a station that has no data frame to transmit in the uplink, QoS Null frames[1] can be sent to request the delivery of the frames buffered at the AP. This enables the usage of U-APSD by an Access Category (AC) of a station which does not generate uplink traffic often enough to meet the QoS requirements of an application using this AC.

In order to guarantee backward compatibility of legacy stations that do not support APSD, the TIM element indicates the buffer status *only* of the ACs that are non delivery-enabled. Only in the case that

[1]QoS Nulls are the substitute in U-APSD of 802.11 power save mode PS-Polls

all ACs of a station are delivery-enabled the TIM element indicates the buffer status of delivery-enabled ACs.

Notice hence that an intelligence is needed in a mobile device implementing U-APSD, to decide how to appropriately trigger the AP for buffered data. As usual the 802.11 standard leaves this intelligence undefined to allow for vendor differentation. Figure 2.7 provides an example of the operation of U-APSD.

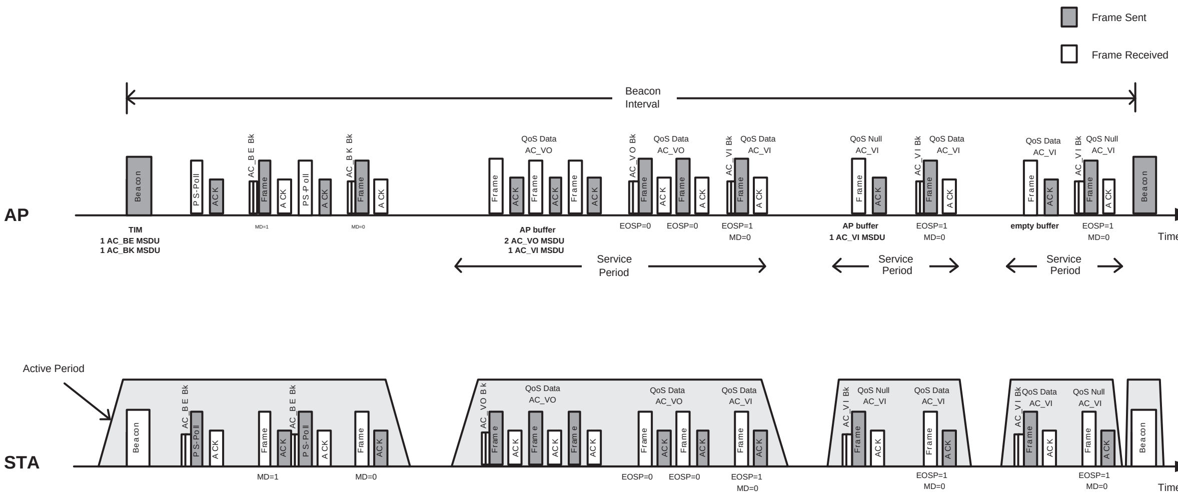


Figure 2.7: Example of U-APSD operation. U-APSD configuration: AC_VO and AC_VI both trigger- and delivery-enabled. AC_BE and AC_BK neither delivery- nor trigger-enabled (i.e., use legacy 802.11 power save mode).

Finally, it is worth pointing out that U-APSD is currently certified and tested within the WMM certification (35) in the Wi-Fi Alliance.

Scheduled-APSD (S-APSD). The main idea behind the S-APSD design is the *scheduling* by the AP of the instants where each station using S-APSD should wake up to receive the frames buffered at the AP.

The usage by a station of the S-APSD delivery mechanism for a traffic stream, in the case where the access policy is HCCA, or for an AC, in the case where the access policy is EDCA, is configured by the transmission of an ADDTS request frame to the AP. In case the AP can satisfy the requested service, it will indicate so in the Schedule Element of the response which will include the *Service Start Time* (SST) and the *Service Interval* (SI) parameters.

The AP is responsible for defining for each traffic stream or AC of a station using S-APSD the SST and the SI necessary for the periodical scheduling of the delivery of frames to the stations. Thus, if a

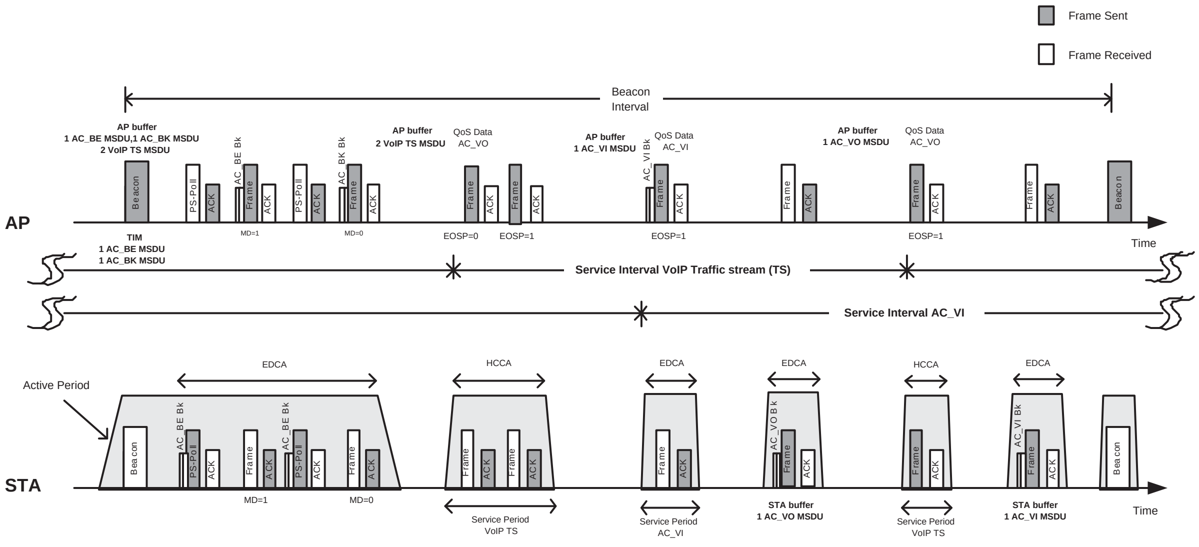


Figure 2.8: S-APSD example of operation. S-APSD configuration: VoIP traffic stream in the downlink configured to use S-APSD with HCCA access mode, AC_VI traffic in the downlink uses S-APSD with EDCA access mode, AC_BE and AC_BK configured to use legacy 802.11 power save mode.

station has set up S-APSD for a traffic stream or an AC, it automatically wakes up at the scheduled time of each Service Period (SP), i.e. $t_k = SST + kSI$.

The AP may update the service schedule at any time by sending a new Schedule frame to a S-APSD station. A station can also modify the S-APSD service schedule by modifying or deleting its existing traffic specification through ADDTS or DELTS messages.

As in the case of U-APSD, in S-APSD a station remains awake until it receives a frame with the EOSP subfield set to 1. If necessary the AP may generate an extra QoS Null frame with the EOSP set to 1 in order to terminate a Service Period.

Finally, notice that a scheduling algorithm is needed in the AP in order to decide how to multiplex in the channel multiple stations implementing S-APSD. As with U-APSD the 802.11 standard leaves this scheduling algorithm undefined in order to allow for vendor differentiation. In Figure 2.8 an example of the S-APSD operation is provided. For more information regarding the power saving protocols defined in the 802.11 standards, the interested reader is referred to (33).

2.1.4 Power Save Multi Poll (PSMP)

Orthogonal to the previous power saving protocols, the 802.11n standard defined a new protocol extension that allows an AP to very efficiently deliver buffered data to multiple stations in power saving that have triggered the AP. This protocol extension is known as *Power Save Multi Poll* (PSMP).

The basic idea behind PSMP is to let an AP transmit a broadcast frame that contains a schedule for the power saving stations that have triggered the AP and are waiting to receive their data. Thus, having a priori knowledge about the exact time where they will receive their data allows power saving stations to further sleep while the AP is transmitting data for other stations. Figure 2.9 depicts an example of the operation of the PSMP protocol.

It is worth to notice that unlike legacy 802.11 PSM and U-APSD, PSMP is currently not being certified by the Wi-Fi Alliance and to the best of our knowledge has not yet been implemented in the market.

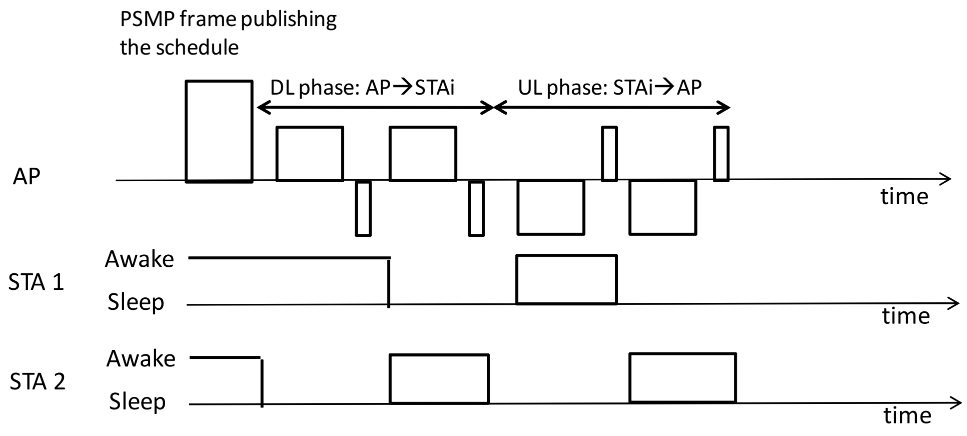


Figure 2.9: Example of PSMP operation. The AP delivers a schedule frame to the stations that trigger the AP to recover their buffered frames. Thus, stations only wake up when they have to transmit or receive data.

2.1.5 Wi-Fi Direct

Finally, we conclude this overview of relevant protocols in the state of the art by introducing the Wi-Fi Direct technology (11). This technology has recently been developed by the Wi-Fi Alliance, and allows Wi-Fi Devices to directly connect to each other without the presence of an Access Point (AP). Hence, Wi-Fi Direct enables novel device centric use cases, like having a mobile phone connecting to a TV set or a wireless printer without an AP.

In order to implement device to device connectivity, Wi-Fi Direct requires devices to implement both the role of a Wi-Fi station and a Wi-Fi AP. Thus, Wi-Fi Direct devices are able to discover each other and then run a negotiation protocol to decide which device will act as AP, referred to as P2P Group Owner, and which device as station, referred to as P2P Client, to then set up a traditional infrastructure network, referred to as P2P Group. Notice that the functionality provided by Wi-Fi Direct certainly overlaps with the original *Ad-Hoc* mode (IBSS) defined in the 802.11 standard (33). However, the Wi-Fi Alliance decided to favor Wi-Fi Direct in front of IBSS due to the following reasons. First, by leveraging the infrastructure mode Wi-Fi Direct allows to immediately reuse all the QoS, security or power saving protocols that have been defined for infrastructure networks. On the other hand, many of these enhancements are lacking in IBSS networks. Second, there are many Wi-Fi devices deployed in the market that only implement infrastructure mode. These devices can seamlessly connect with Wi-Fi Direct devices, since these will simply appear to a legacy device as a traditional AP.

In detail the Wi-Fi Direct functionality is described as follows:

- Two Wi-Fi Direct devices can discover each other by sending Probe Request frames while using a discovery algorithm that alternates between three *social channels*, namely channels 1, 6 and 11, in the 2.4 GHz band.
- After discovering each other, devices run a negotiation protocol to decide which device will act as P2P Group Owner. For this purpose each device can express its intent to become a P2P Group Owner, and a tie breaking rule is defined in case both devices express the same intent. Thereafter the device that becomes the P2P Group Owner starts beaconing and initiates a traditional infrastructure network.

In addition to the previous discovery and negotiation protocols, the Wi-Fi Direct technology also introduces the following new functionalities:

- Layer Two service discovery, that allows Wi-Fi Direct devices to discover which services are available in other devices before setting up a P2P Group. In particular, the Service Discovery protocol allows for instance to embed UPnP (59) or BonJour (60) service discovery querys directly at Layer Two.
- Concurrent Operation, that allows Wi-Fi Direct devices to be part of multiple P2P Groups at the same. For instance a handset device could set up a P2P group with a TV set in order to stream some content, and simultaneously set up another P2P Group with a printer in order to print a document. In addition, Wi-Fi Direct also allows devices to simultaneously create P2P Groups while being connected to an infrastructure Wi-Fi network.
- AP power saving capabilities. If mobile computing devices are to successfully act as P2P Group Owners in Wi-Fi Direct, then power saving protocols have to be defined that allow a P2P Group

Owner to operate in an energy efficient way. Note that 802.11 standards only support power saving operation in the station side. In order to address this concern, Wi-Fi Direct has defined two AP power saving protocols, namely the *Opportunistic Power Save* and the *Notice of Absence* protocols. These protocols are very relevant to the work presented in this thesis and will be carefully described in Chapter 7.

2.2 Literature survey

Besides the work carried out in the 802.11 group in order to define power saving protocols, there has been to date also a significant effort from the research community towards improving the performance of these protocols. In this section we survey the, in our understanding, most relevant contributions in the field and present a taxonomy to classify existent work in the state of the art. In particular our taxonomy contains the following categories:

- The design of energy efficient solutions is often very dependent on the type of application running in the mobile device. Therefore we classify efforts in the state according to: i) ***Generic Protocol Enhancements***, which do not target any specific application, ii) Proposals targetting energy efficiency with ***Real-Time*** applications like Voice and Video, and iii) Proposals targetting energy efficiency with ***Data*** oriented applications, like Web traffic or File transfers.

- We also classify proposals in the state of the art according to whether they are deployed over ***Distributed Wi-Fi protocols***, like EDCA and StdPSM or U-APSD, or over ***Centralized Wi-Fi protocols***, like HCCA and S-APSD.

- Existent work in the state of the art, can also be classified according to whether it is a ***Layer Two solution***, or whether it is a ***Cross-Layer solution*** that relies on functionality implemented in higher layers of the stack.

- Finally, we provide a separate look at the proposals in the state of the art that target ***power saving in the Access Point***, instead of in the station.

2.2.1 Generic protocol enhancements

In a shared network using a listen before talk protocol, like Wi-Fi, energy efficiency can be improved by tuning the MAC protocol parameters in order to minimize the energy required to resolve the contention and access the channel. In (77) an adaptive distributed algorithm was proposed where contending stations monitor the channel in order to measure the current collision probability, which is then used to size the station's contention parameters in order to minimize energy consumption. Following a similar idea, a modification of the DCF protocol was proposed in (76) that minimizes the necessary contention to access the channel and hence maximizes energy efficiency. A drawback of these approaches is that stations need to be all the time awake in order to obtain reliable estimates of the collision probability. A different approach can be found in (75), where the authors propose to let power saving stations sleep during the contention process. In order to implement this method though, changes are needed in the DCF protocol that may result in unfairness between the stations in power saving implementing the proposed mechanism and legacy stations.

Another generic way to increase energy efficiency is to reduce the size of the transmitted packets, hence minimizing the time that a station spends in the Transmit state. In (121) a generic aggregation scheme was proposed, which is not compatible with 802.11n, and the obtained energy savings were evaluated. In addition, (50) proposed a novel aggregation scheme, which unlike 802.11n, allows to aggregate in a single frame MPDUs belonging to different stations, hence increasing aggregation opportunities and further reducing transmission times. Notice though, that aggregating frames addressed to multiple stations in a single physical frame comes at the price of an increased complexity, for instance in order to guarantee that all stations are awake at the same time or in order to let multiple stations deliver their ARQ feedback back to the AP.

Finally, generic improvements in energy efficiency have also been proposed for the centralized protocols defined in Wi-Fi. For instance (79) studied how to schedule in the AP the delivery of buffered data for stations in power save mode, and proposed to embed scheduling information within the TIM element in the Beacon. In (78) a multi-poll scheme and a scheduling policy that maximizes energy saving were proposed where the AP broadcasts scheduling information for the stations in power save mode after the Beacon. Finally, in (80) a centralized scheduled power save mode is proposed, where the AP divides a Beacon frame into non-overlapping time slices and conveys within the TIM element the assignment of slices to power saving stations.

Notice though that the presented generic protocol enhancements require modifications to existent 802.11 standards, and are therefore not directly implementable in existent Wi-Fi networks and devices.

2.2.2 Real-Time traffic

A significant amount of work in the state of the art has been devoted to improve energy efficiency with real-time communications. The reason is that the usually long duration of real-time sessions, like Voice or Video, can heavily affect the battery of a mobile device if such communications are not performed in an energy efficient manner. Next, we describe some of the relevant contributions in this field available in the literature.

First, we start by considering proposals that rely on the Distributed Wi-Fi QoS and power saving protocols, which are the ones enjoying a widest deployment in the market. These works can be classified according to the application they target, i.e. typically Voice or Video, and according to whether the proposed solution is a Layer two solution or a cross-layer solution. Regarding Voice, (45) and (46) evaluated the performance of U-APSD and studied strategies in the Access Point to minimize the energy consumption of Voice stations using U-APSD. In addition, (83) proposes an adaptive layer two scheme where VoIP stations select their sleep intervals based on measurements of the collision probability. Regarding cross-layer approaches to improve the energy efficiency of VoIP over Wi-Fi, in (72) a cross-layer algorithm is proposed where input from the Voice application is used to optimally adjust the station's polling interval according to estimations of the network delay and the Voice playback deadlines. In (82) it is proposed to detect Voice packets in the MAC layer, and adaptively disable MAC layer acknowledgements for these packets in order to reduce energy consumption. Finally in (73) a novel architecture is proposed that leverages the cellular interface in mobile phones in order to improve the energy efficiency of VoIP over Wi-Fi.

Regarding contributions specifically targetting Video, in (43) the authors analyzed how the Standard Power Save Mode defined in the original 802.11 standard can heavily degrade the QoS of streaming applications due to the excessive delay introduced. In addition, the authors proposed a history based approach in the mobile client to estimate the duration of the idle periods in the video stream in order to allow the device to sleep during these intervals. This initial work was expanded by the same authors in (44), where a more sophisticated linear prediction strategy was used instead of the history based predictor. This initial work though focuses mainly in downlink communications (real-time streaming) and does not consider the advanced features of the APSD protocols defined in 802.11e.

Our previous work in (58) and (47) evaluates the performance of U-APSD and legacy 802.11 PSM in a multi-service network with both real-time and data applications. This work illustrates the advantage of U-APSD in front of legacy 802.11 PSM in terms of QoS, energy efficiency and network capacity.

Regarding contibutions targetting centralized Wi-Fi QoS and power saving protocols, i.e. HCCA and S-APSD, in (48) a layer two Enhanced scheduler for APSD (E-APSD) was proposed that adapts the service interval (sleeping period) of stations performing Video streaming according to the amount of traffic buffered in the AP. For that purpose a threshold is defined on the amount of data addressed to a certain station in the power save buffer in the AP, and depending on whether the actual amount of buffered data is above or below the defined threshold the power saving station is allowed to shrink or expand its service interval. The proposed E-APSD scheduler though focuses mainly on downlink communications (only the queue in the AP is monitored) and uses non standard mechanisms to convey the new service intervals to the power saving stations. In (49) a layer two Power Save Feedback Based Dynamic Scheduler (PS FBDS) for HCCA is proposed in order to provide bounded delays for both Voice and Video traffic while ensuring energy savings. The proposed scheduler defines a common polling interval for all flows that is used to poll the different HCCA stations sequentially, and derives the length of the Transmission Opportunities (TXOPs) to be granted to each station by means of a control law based on the queue status in the mobile stations. The proposed scheduler has the drawback of requiring a common service interval for all associated stations, which in general can not be matched to the QoS requirements of each station, hence resulting either in excessive delays if too large service intervals are used or in excessive polling overhead if too small service intervals are used. Finally, our previous work in (68) provides a detailed comparison between the performance of S-APSD, both over EDCA and HCCA, U-APSD and legacy 802.11 PSM, and illustrates how the signaling reduction achieved in S-APSD translates into significant gains in terms of QoS, energy efficiency and network capacity.

2.2.3 Data traffic

There is also a significant body of work in the state of the art that tackles the performance energy trade-off of Data applications in Wi-Fi. The reason is that the research community soon realized how the increased response times caused by the Wi-Fi power saving protocols could heavily degrade the performance of TCP based applications. Next, we provide an overview of some of the most relevant work in this area, which we classify in contributions focusing on Web-like traffic, and contributions focusing on large File transfers. In addition, contributions are also classified depending on whether they consist of a Layer two or a cross-layer solution. Finally, notice that all contributions in this field are based on the distributed Wi-Fi QoS and power saving protocols.

A pioneering work in this area was carried out in (39) where the authors performed an elaborated analysis about the slow down introduced by the Standard Power Save Mode protocol in a short TCP transfer. The authors concluded that although being very good with respect to the achieved power reductions, the Standard Power Save Mode protocol could introduce excessive slow downs that significantly affected the performance of Web traffic. Therefore, an adaptive layer two algorithm was proposed, Bounded Slow Down (BSD), that dynamically selects the awake and sleep intervals of the mobile device in order to bound the slow down introduced by the power saving protocol. This work was extended in (40) where a more energy efficient approach was proposed that achieved the same performance in terms of bounding the additional delay introduced by the power saving protocol. Both (39, 40) can be seen as extensions of Standard Power Save Mode that allow to trade-off a reduction of power consumption with the extra delay introduced by the power saving protocol. However, if small delays are desired the power consumption of these approaches becomes significantly worse than the one of Standard Power Save Mode. Following a similar design philosophy, in (81) an adaptive beacon listening protocol was proposed that adapts the listening intervals of a station based on an estimation of the RTT during the slow start phase of a TCP connection.

Several works in the state of the art rely on cross-layer solutions in order to improve the performance energy trade-off of Data applications. For instance, in (19) a middleware sitting beween the Wi-Fi interface and the applications, known as Self Tunning Power Management (STPM), was introduced. STPM collects information from the applications, like the amount of data to be transmitted or delay tolerances, and based on the collected information configures the wireless interface. Along the same lines, in (41) a Cross-Layer Energy Manager (XEM) was presented that configures the wireless interface to use the appropriate power saving mode depending on the observed traffic patterns from the applications. For instance, the Wi-Fi interface is completely switched off during user think times, and operates in power save mode when non real-time traffic is being transmitted. In order to distinguish user think times from simple idle times XEM exploits information from all layers in the protocol stack. Another interesting cross-layer approach was presented in (38) that focused on large data transfers over TCP, which nowadays enjoy a very wide spread due to popular content distribution services that use TCP for Video streaming (42). Based on the observation that normally not all the bandwidth available in the WLAN network is used by a TCP connection, the authors proposed a cross-layer algorithm where the TCP advertised window field is used to shape TCP transmissions in the WLAN in predictable bursts that can be used by the mobile device in order to gain sleeping opportunities without decreasing the throughput of the TCP transfer. Finally, a similar approach was also proposed in (74) where the authors in addition to the advertised window adjustments, also suggest to switch off the wireless card for a given timeout after detecting that the TCP socket has been inactive for a certain amount of time.

2.2.4 AP power saving

We conclude this survey of the literature by looking at contibutions that have targetted energy efficiency for Access Points instead of Wi-Fi stations. In this context, the authors in (54) and (55) studied the feasibility of having solar powered Access Points, and proposed layer two protocols and algorithms to optimize energy efficiency in this case. In addition, (56) suggests that a Wi-Fi Access Point can deliberately advertise long channel reservations using the duration field in the Wi-Fi header, and use the reserved time to switch off its radio in order to save energy. To the best of the author's knowledge there are to date no proposals in the state of the art proposing algorithms to manage the AP power saving protocols defined in Wi-Fi Direct.

In order to conclude this survey, Table 2.1 summarizes the works described in this section according to the proposed taxonomy.

		Distributed		Centralized	
		Layer 2	Cross-Layer	Layer 2	Cross-Layer
Generic Protocol Enhancement		(75)† (77)† (76)† (121)† (50)†		(79)† (78)† (80)†	
Real-Time Traffic	Voice	(45) (46) (83) (58) (47)	(72) (73) (82)†	(49) (68)	
	Video	(43) (44) (58) (47)		(48)† (49) (68)	
Data Traffic	Web-like Traffic	(39) (40) (81)	(41) (19) (74)		
	File Transfers		(38) (74) (19)		
AP Power Saving		(54) (55) (56)			

Table 2.1: Survey of the State of the Art. The references marked with ()† require 802.11 non-standard protocol modifications.

2.3 Goals and design guidelines of this thesis

Despite the extensive work existent in the state of the art as described in the previous section, there still are significant roadblocks to be overcome in order to turn Wi-Fi into an energy efficient technology. Proof of this fact are the reduced battery lifetimes experienced in current mobile computing devices implementing Wi-Fi, and the fact that most of the approaches described in the previous section are not deployed in the market. Thus, the major goal of this thesis is to propose a set of algorithms that can be readily implemented in existent Wi-Fi devices in order to improve energy efficiency. In particular, we try to address the reasons that hinder deployability of the current approaches in the state of the art, by establishing the following fundamental guidelines in the design of the solutions presented throughout this thesis:

- **Stay in the MAC layer**. While cross-layer solutions have the potential of delivering an enhanced performance, they come at the very high price of having tro re-engineer existent and future applications in order to deliver the required information. The previous is often not possible in modern mobile computing devices where the trend is to provide simplified APIs to ease the work of application developers (51). Therefore, the solutions presented in this thesis will operate exclusively at the MAC layer, and a major challenge to be tackled by our proposed algorithms will be how to obtain in the MAC layer information about the applications in order to deliver a performance comparable to cross-layer approaches.

- **Design within the limits of the current 802.11/Wi-Fi standards**. Another reason that hinders deployability of the solutions in the state of the art, is the need of protocol modifications. It is worth to notice that bringing solutions into standards is a complicated and tedious process, sometimes driven by reasons other than the purely technical ones. In addition, even if the proposed solutions could be standardized, current devices in the market could not benefit from them. Therefore in this thesis we will design solutions that stay within the limits allowed by current standards.

- **Favor Client Side solutions**. Except when designing centralized scheduling algorithms, in this thesis we will focus on client side solutions instead of Access Point (AP) side solutions in order to improve energy efficiency of mobile computing devices. The reason for this design decision is that energy is a big concern specially for the stations. Therefore, since improving energy in the stations does not bring a direct competitive advantage to traditional APs, AP vendors often prefer to concentrate their engineering efforts on other features like enhanced data rates. In addition, deploying solutions on the station side has the advantage that a mobile computing device can operate in an energy efficient way regardless of the AP it is connected to.

Finally, in order to conclude this section, Table 2.2 positions the contributions of this thesis introduced in Chapter 1 within the taxonomy of the state of the art depicted in Table 2.1.

		Distributed		Centralized	
		Layer 2	Cross-Layer	Layer 2	Cross-Layer
Generic Protocol Enhancement					
Real-Time Traffic	Voice	Chapter 3		Chapter 5	
	Video	Chapter 4			
Data Traffic	Web-like Traffic	Chapter 6			
	File Transfers				
AP Power Saving		Chapter 7			

Table 2.2: Positioning of the contributions of these thesis.

3

An Adaptive Triggering Algorithm for Distributed Wi-Fi Power Saving Modes

We have seen in Chapter 2 that the long durations of real-time sessions, like Voice or Video, may severely affect the battery life of Wi-Fi mobile computing devices. It is thus critical to define algorithms and protocols that allow a Wi-Fi device to operate in an energy efficient way without compromising the QoS of real-time traffic.

In order to address this challenge, several authors in the literature have proposed to leverage the regular transmission patterns typically used by real-time codecs, in order to let a Wi-Fi station in power save mode trigger the Access Point (AP) at regular intervals and thus save power by sleeping between transmissions. Indeed, this is the fundamental idea behind the U-APSD protocol described in Chapter 2, and as it was shown in Figure 1.5 current Wi-Fi chipsets are able to implement this idea in a very efficient way.

However, there is a fundamental question to be solved within this framework, that is how can a Wi-Fi station decide on the optimal interval to use in order to trigger the AP for its buffered data. Answering this question is the goal of this chapter. It is worth to notice that the QoS experienced by real-time applications in the downlink direction (AP$\rightarrow$station) will become highly dependent on the algorithms that decide when to poll an AP in order to retrieve the buffered frames. On the one hand, if a station does not generate enough trigger frames, an application can suffer from an excessive delay in the downlink direction. On the other hand, if a station transmits too many unnecessary triggers, power saving is penalized and the congestion level in the network increases. As usual, the 802.11 and 802.11e standards leave open the algorithms to generate trigger frames to allow vendor differentiation.

Different approaches to solve this problem are considered today by Wi-Fi mobile device vendors. These approaches can be mainly classified into static and dynamic cross-layer approaches. Static approaches aim at bounding the delay of applications in the downlink using a *fixed polling interval*. Such a static approach though can not be optimal if different applications with different QoS requirements are used in the same mobile device. Dynamic cross-layer approaches can overcome this problem by re-configuring the MAC's layer polling interval every time a new application starts in order to meet its specific requirements. However, several issues need to be considered that limit the suitability of the cross-layer approach for multi-purpose Wi-Fi devices in practice:

- No generic framework is available for the application layer to become aware of the QoS requirements of the applications.
- No generic interface is available for communication between the application layer and the MAC layer.

The UPnP Forum (59) started some work in the direction of achieving a standard cross-layer communication framework. However, this is a complex task that will not be completed in the near future because it requires the agreement of both software and hardware vendors in a technical specification and a certification program in order to ensure interoperability. Currently, due to the absence of a generic cross-layer framework, the dynamic cross-layer solution is not scalable because new functionality has to be added in the devices every time a new application needs to be supported. In addition, applications that dynamically vary their frame generation rate according to network congestion, like Temporal Scalable Video Coding (52) or Skype (53), make the design of such a cross-layer framework even more challenging.

In order to solve the previous problem, in this chapter we introduce the design of an adaptive algorithm that runs in a Wi-Fi station in power save mode, and using information available at the MAC layer, is able to adapt the used trigger interval according to the QoS requirements of the applications. The content of this chapter has been published in (26) and its main contributions are as follows:

- We present an analytical model that captures the effect of the trigger interval used by Wi-Fi distributed power saving mechanisms on the delay and jitter experienced by real-time applications.
- We design a layer two adaptive algorithm based on the steepest descent method that can be used in a Wi-Fi station in order to configure a distributed power saving mechanism such that QoS is provided without requiring knowledge from the applications.

- We derive an analytical model that studies the stability and convergence properties of our proposed algorithm.
- Finally, we present a thorough simulative study that evaluates the advantages of our approach with respect to other proposals available in the state of the art.

This chapter is structured as follows. Section 3.1 presents an analytical model of the effect of distributed power saving mechanisms on QoS. This model is then used in Section 3.2 to design a layer two algorithm that adapts to the characteristics of the applications. The properties of the adaptive algorithm are analytically studied in Section 3.3 and a thorough performance evaluation is provided in Section 3.4. Finally, Section 3.5 summarizes the results and concludes this chapter.

3.1 Modeling the effect of distributed power saving mechanisms on applications with QoS requirements

This section introduces an abstracted model that will allow us to quantify the effect that distributed power saving mechanisms have on the delay and jitter experienced in the downlink direction (AP$\rightarrow$station) by applications with QoS requirements. The dependencies derived with this model will be used in the next section to design an algorithm that adapts to the traffic characteristics of the applications in order to meet their QoS needs. Although our focus is on guaranteeing that applications with QoS requirements can be run satisfactorily in mobile devices using a Wi-Fi distributed power saving mode, we will also show that the same algorithm can improve the performance of applications with lower QoS requirements, e.g., Web browsing and FTP downloads.

The main characteristics of Wi-Fi distributed power saving mechanisms are:

i) Frames addressed to power saving stations are buffered in a network entity.

ii) Stations are regularly informed through signaling messages about whether frames have been buffered for them.

iii) Stations can request the delivery of their buffered frames at any time by generating signaling triggers and in the U-APSD case also by reusing data frame transmissions in the uplink.

iv) Stations' uplink data transmissions are not affected by the power saving mechanisms.

In order to simplify the derivation of an analytical model of the delay and jitter introduced in the downlink by Wi-Fi distributed power saving mechanisms we make two approximations. The first approximation is that the traffic characteristics in the downlink of applications with QoS requirements can be modeled as a stream of packets, of fixed or variable size, that arrive at the AP with a constant interarrival time. Note that codec-based applications would fit in this model. In Section 3.3 we relax this assumption by considering jitter and drops in the incoming stream. The second approximation is that the delay experienced by a downlink frame is the time since the data frame was inserted in the AP's buffer until a station sent the corresponding trigger frame, i.e., the time to get access to the channel is negligible compared to the buffering time and the station trigger generation interval. This last approximation holds true when the level of congestion in the channel is moderated which is the working area of interest when QoS requirements need to be satisfied. The effect of congestion in the Wi-Fi network will be further studied in Section 3.4.

Two different cases are distinguished in the analysis. The case where uplink data frames can not be used as trigger frames (802.11 power save mode) and the case where uplink data triggers can be used as trigger frames (U-APSD).

3.1.1 No uplink data triggers

Let us consider the case where downlink frames arrive at the buffering entity with an interarrival time represented by Δ_{DL} and stations in power saving poll the AP with a trigger interval equal to Δ_{trig}.[1]

In order to model the delay experienced by the downlink stream the following can be derived. Given a trigger frame sent by a station and assuming that only one frame was retrieved from the AP as a result of that trigger, the delay experienced by this frame can be expressed as $d(k) = t_{trig}(k) - t_{arr}(k)$. Where $t_{arr}(k)$ represents the arrival time of the k-th downlink data at the AP, and $t_{trig}(k)$ represents the reception time of the trigger sent by the station.

To understand the dynamics of $d(k)$ we consider the delay experienced by a reference frame in the downlink, d_0, and observe that no delay above the station's polling interval, Δ_{trig}, can be obtained in our model. Hence the delay experienced by the downlink frames can be expressed like:

$$d(k) = (d_0 + k\epsilon) \bmod \Delta_{trig} \tag{3.1}$$

Where k represents the k-th downlink frame arrived at the AP after the reference frame and $\epsilon = \Delta_{trig} - \Delta_{DL}$, represents the difference between the polling interval used by the station and the interarrival time of the downlink stream.

[1]In 802.11 PSM a station would generate extra PS-Polls when a frame with the More Data bit set is received. In the presented analysis we collapse these extra triggers into a single one, which is the case of U-APSD.

In order to prove Equation 3.1 consider that the k-th downlink frame is retrieved by the l-th trigger frame generated by the station. Applying the modulo function:

$$\begin{aligned} d(k) = t_{trig}(k) - t_{arr}(k) &= d_0 + l\Delta_{trig} - k\Delta_{DL} \iff d(k) + (k-l)\Delta_{trig} = \\ &= d_0 + k\epsilon \iff d(k) = (d_0 + k\epsilon) \bmod \Delta_{trig} \end{aligned}$$

Equation 3.1 can be interpreted in the following way. First, it can be seen that the delay experienced by the downlink stream increases or decreases linearly with a slope equal to ϵ. Second, the downlink stream never experiences a delay above the polling interval used in the station, Δ_{trig}, resulting in a saw-tooth pattern. Figure 3.1(a) shows the downlink delay obtained in a simulation experiment[1] where a single Wi-Fi station using U-APSD receives a CBR stream in the downlink. The results corroborate the correctness of Equation 3.1.

Based on Equation 3.1, the jitter in the downlink direction can be expressed as $|j(k)| = |d(k) - d(k+1)| = |\lfloor \frac{d(k)+\epsilon}{\Delta_{trig}} \rfloor \Delta_{trig} - \epsilon|$. The following bounds can be obtained considering $|\epsilon| < \Delta_{trig}$ [2]:

$$|j(k)| = \begin{cases} |\epsilon| & \text{if } d(k) < \Delta_{DL}, d(k) \geq |\epsilon| \\ \Delta_{trig} - |\epsilon| & \text{if } d(k) \geq \Delta_{DL}, d(k) < |\epsilon| \end{cases} \tag{3.2}$$

Where the two different conditions on $d(k)$ represent the cases of $\epsilon \geq 0$ and $\epsilon < 0$ respectively. Figure 3.1(a) depicts the downlink jitter experienced in the same experiment previously described. Notice in the figure how the spikes in jitter match the saw-tooth pattern observed in the downlink delay, and correspond to the instants where the station generates a trigger frame and retrieves more than one data frame from the AP.

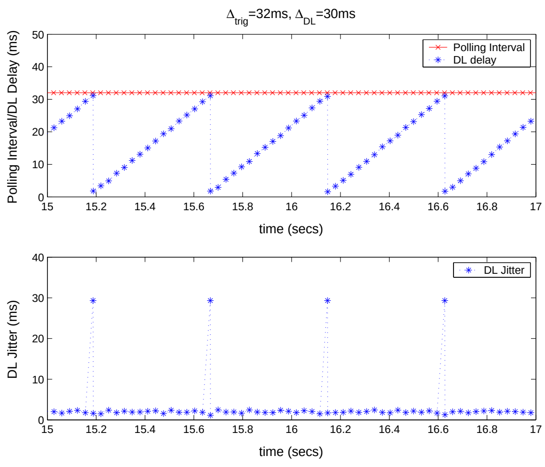
3.1.2 Uplink data triggers

Consider now the case where a station reuses uplink data frames as triggers. In this case a periodic trigger generation solution turns out not to be optimal since trigger frames would be generated more often than Δ_{trig}. Thus, in order to avoid the introduction of unnecessary signaling, a *re-scheduling* mechanism should be used as it was proposed in our early work in (47). This solution consists in scheduling a periodic generation of signaling triggers and re-scheduling the pending signaling trigger transmission every time a data trigger is sent in the uplink.

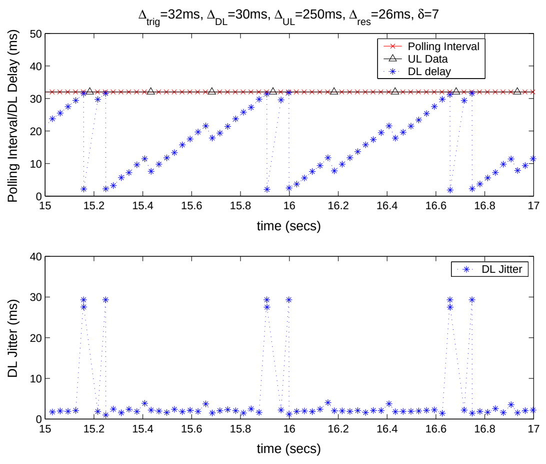
In order to model this solution, let us consider a Wi-Fi station generating data triggers in the uplink every Δ_{UL}, with $\Delta_{UL} > \Delta_{DL}$ and $\Delta_{UL} > \Delta_{trig}$ since otherwise the re-scheduling mechanism would

[1]The simulation framework used in this chapter is detailed in Section 3.4.

[2]Note that this is the usual region of operation of an adaptive power save algorithm as the one presented in section 3.2.



(a) DL stream's delay/jitter without UL data triggers.



(b) DL stream's delay/jitter with UL data triggers.

Figure 3.1: DL stream's delay/jitter under a constant polling interval, Δ_{trig}.

avoid the transmission of signaling triggers, obtaining then a trigger generation pattern like in the previous section[1]. Thus, besides the uplink data triggers, a station schedules the transmission of signaling triggers every Δ_{trig} as shown in Figure 3.2.

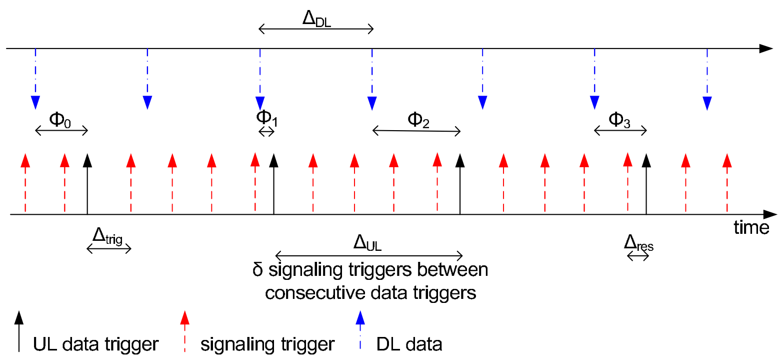


Figure 3.2: Data and signaling trigger generation employing re-scheduling.

Note that when the re-scheduling mechanism is applied, $\delta = \lfloor \frac{\Delta_{UL}}{\Delta_{trig}} \rfloor$ signaling triggers are always generated between two consecutive uplink data frames. In addition, the time between a signaling trigger and an uplink data trigger is $\Delta_{res} = \Delta_{UL} \bmod \Delta_{trig}$, with $0 \leq \Delta_{res} < \Delta_{trig}$.

[1]A practical example of this setting could be a Video application generating periodic RTCP updates in uplink or a Voice conversation using different codecs in uplink and downlink.

Although the introduction of uplink data triggers increases the complexity of modeling the dynamic behaviour of the downlink delay $d(k)$, based on the results from the previous section it can be seen that the delay experienced by the downlink stream is still bounded by Δ_{trig} and linearly increases or decreases between uplink data frames, with a slope equal to $\epsilon = \Delta_{trig} - \Delta_{DL}$. Whenever an uplink data frame is sent in the uplink, the downlink delay varies according to $\epsilon_{res} = \Delta_{res} - \Delta_{DL}$. Figure 3.1(b) shows the same experiment considered in the previous section but adding a periodic generation of data triggers in the uplink. Note that although Δ_{trig} is above Δ_{DL}, Δ_{res} is not and therefore some uplink data triggers find no packets in the AP's power save buffer.

Regarding the jitter of the downlink stream, the following can be said. The jitter continues to be constant and equal to ϵ between consecutive uplink transmissions, and varies according to ϵ_{res} when a station sends an uplink data trigger. A new situation has to be considered though, if an uplink data trigger finds no data buffered in the AP. In this case, the jitter is $|j(k)| = |d(k) - d(k+1)| = |d(k) - (d(k) + \Delta_{res} + \Delta_{trig} - \Delta_{DL})| = \Delta_{res} + \epsilon$, which could be above the maximum value found in the previous section. Figure 3.1(b) shows the downlink jitter in the experiment previously described.

Another interesting metric to consider in this case is the amount of uplink data triggers actually retrieving data from the AP. The former is a measure of how efficiently a station reuses these data frames as triggers. Ideally, we would like all data frames to be reused as triggers in order to minimize the amount of generated signaling triggers.

From Figure 3.2 it can be observed that an uplink data trigger retrieves a frame from the AP if and only if $\phi_i < \Delta_{res}$. Where ϕ_i represents the time between an uplink data trigger transmission and the previous corresponding downlink frame that arrived at the AP.

In order to find how many uplink data triggers will find a downlink frame to download, it can be seen that the compound sequence of uplink triggers and downlink frames is periodic, with period $T = lcm(\Delta_{UL}, \Delta_{DL})$, where lcm stands for least common multiple.

Being such sequence periodic, the sequence of ϕ_i values will also be periodic, with period $N = \frac{T}{\Delta_{UL}}$. Indeed, the actual ϕ_i values can be expressed as:

$$\phi_i = (\phi_0 + i\Delta_{UL}) \bmod \Delta_{DL} \tag{3.3}$$

Where ϕ_0 is any of the values of the sequence taken as initial reference.

Since Equation 3.3 is a linear congruence, the *linear congruence theorem* can be applied to find that there will be integer solutions on i if and only if $gcd(\Delta_{UL}, \Delta_{DL})$ divides $\phi_i - \phi_{min}$. Where gcd stands for greatest common divisor and $\phi_{min} = \phi_0 \bmod gcd(\Delta_{UL}, \Delta_{DL})$. Therefore, the elements contained in the set $\{\phi_i\}$ are the same elements contained in the set $\{\phi_j\}$ defined as:

$$\phi_j = \phi_{min} + j \cdot gcd(\Delta_{UL}, \Delta_{DL}) \quad j = 0...N-1 \tag{3.4}$$

Recalling now that uplink data triggers will find data frames buffered in the AP's power save buffer if and only if $\phi_i < \Delta_{res}$ and considering Equation 3.4, the number of uplink data triggers that, within a period T, retrieve frames from the AP is:

$$M = \lceil \frac{\Delta_{res} - \phi_{min}}{gcd(\Delta_{UL}, \Delta_{DL})} \rceil \quad 0 \leq M \leq N \tag{3.5}$$

3.2 Algorithm design

We have motivated the need to configure the Wi-Fi distributed power saving mechanisms according to the traffic characteristics of the applications based only on information available at the MAC layer and have shown in Section 3.1 how the delay and jitter experienced by an application in the downlink depend on the polling interval used by a station. In this section, to overcome the limitations of a static approach and avoid the problems that arise when using cross-layer mechanisms, we propose an adaptive algorithm that estimates the interarrival time of a downlink stream, Δ_{DL}, and generates signaling trigger frames accordingly.

The reason for choosing this approach is that, according to the model derived in Section 3.1, if we would poll the AP at intervals equal to the downlink interarrival time, Δ_{DL}, the downlink delay would be kept constant, bounded to the Δ_{DL} value, and jitter would be eliminated. Note that the estimated value could also be used to poll the AP at multiples of the interarrival time depending on whether a reduction on delay/jitter or signaling load is preferred. The larger the polling interval the lower the signaling load but the larger the delay and jitter. Thus, an algorithm that estimates the downlink interarrival time allows to efficiently address the trade-off between delay/jitter and signaling load.

We define the following objectives for the design of our adaptive algorithm:

1. Bound the delay of the downlink frames according to their interarrival time at the AP.

2. Minimize the amount of signaling load required to achieve step 1.

3. Rapidly adapt to variations in the interarrival time.

4. Minimize the complexity, in terms of the number of updates required to follow changes in the interarrival time.

Given that the downlink interarrival time at the AP is unknown at the MAC layer of a station and can not be conveyed by the AP using standard mechanisms, an algorithm in a station that could meet the proposed objectives would be one that counts all the frames received from the AP, n_fr_rcvd, during a time window, T_w, to afterwards update the downlink interarrival time estimation by computing $\frac{T_w}{n\_fr\_rcvd}$. The bigger T_w, the better the updated value of the estimation since more frames will be received from the AP. The problem of increasing T_w though, is that sudden changes in the value of the interarrival time can not be followed immediately. In order to achieve better precision using small time windows, a moving average could be considered, but the problem again would be to quickly follow changes of the input because average filters introduce memory.

Therefore, instead of using a fixed T_w value, we propose an algorithm that updates the current value of the estimation according to the occurrence of specific events defined based on the following observations:

- If a station generates trigger frames at the same rate that downlink frames arrive at the AP, each trigger frame will always result in a single frame delivered by the AP.
- If a station generates trigger frames at a rate that is above the rate of arrivals at the AP, each trigger frame will result in either one or no frames delivered by the AP.
- If a station generates trigger frames at a rate below the rate of arrivals at the AP, each trigger frame will result in either one or more frames delivered by the AP.

According to the above observations, two types of events can be defined that univocally identify whether the estimation in the station is above or below the actual interarrival time of the downlink packets at the AP.

- *No Data* event: If a trigger is sent by a station and no frame is delivered by the AP. This event implies that the station holds an estimation below the actual downlink interarrival time.
- *More Data* event: If a trigger is sent by a station and more than one frame is delivered by the AP. This event implies that the station holds an estimation above the actual downlink interarrival time.

These two events can be recognized at the MAC layer of the stations using any of the distributed power saving mechanisms available today. The *No Data* event can be recognized because the AP delivers an empty frame, QoS Null, when a trigger is received but no data is buffered. In addition in (61) we proposed a more efficient method to signal this event, where the need to send explicit signaling is avoided by making use of a bit available in the ACK frames. Regarding the More Data event, it can be recognized by simply counting the number of frames received before getting a frame with the EOSP bit[1] set to 1 in the case of U-APSD, or monitoring the More Data bit in the case of 802.11 power save mode.

An algorithm that aims to estimate the downlink interarrival time can be therefore proposed that increases its current estimation whenever a No Data event occurs, and decreases its current estimation whenever a More Data event occurs. The problem hence is to investigate how the current value of the estimation, $\Delta_{trig}(n)$, has to be updated every time an event occurs in order to converge to the actual downlink interarrival time, where $\Delta_{trig}(n)$ represents the polling interval used by a station after the n-th estimation update, $n \in \mathbb{Z}$.

Our proposal is to use an *steepest descent* algorithm to update the estimation and converge to Δ_{DL}. The idea behind the steepest descent algorithm is to estimate at each iteration the error between the current estimation, $\Delta_{trig}(n)$, and the actual downlink interarrival time, Δ_{DL}, and use this information to update $\Delta_{trig}(n)$ in order to get closer to Δ_{DL}.

A steepest descent algorithm for a single variable, in this case the variable Δ_{trig}, can be expressed as (62):

$$\Delta_{trig}(n+1) = \Delta_{trig}(n) - \gamma \frac{\partial}{\partial \Delta_{trig}} J(\Delta_{trig}(n)) \tag{3.6}$$

Where $\Delta_{trig}(n)$ represents the current value of the estimation, $\gamma > 0$ is the adaptation step that controls the speed of convergence and $J(\Delta_{trig})$ is a derivable and convex error function that has a minimum at the point where the algorithm converges. To avoid converging to a value other than Δ_{DL} we select an error function that has only one minimum:

$$J(\Delta_{trig}(n)) = (\Delta_{trig}(n) - \widehat{\Delta_{DL}})^2 \tag{3.7}$$

Where $\widehat{\Delta_{DL}}$ is an estimator of Δ_{DL}, that has to be used because the actual value of Δ_{DL} is the unknown value we are looking for. Thus, the successful convergence of the algorithm will depend on whether $\widehat{\Delta_{DL}}$ is a good estimation of Δ_{DL}. We consider:

$$\widehat{\Delta_{DL}} = \frac{\Delta_t(n)}{n\_fr\_rcvd(n)} \tag{3.8}$$

Where $\Delta_t(n)$ is the time measured at a station between the current event and the previous one and $n\_fr\_rcvd(n)$ is the number of frames that a station has retrieved from the AP during $\Delta_t(n)$, using a polling interval $\Delta_{trig}(n)$.

[1]The End Of Service Period (EOSP) bit signals the end of a Service Period and allows a station to go back to sleep mode in U-APSD.

The routine described in Algorithm 3.1 illustrates our proposed implementation. This routine is executed by the power saving stations every time a service period (SP[1]) completes.

Several points deserve special attention in Algorithm 3.1. First, it can be observed in lines 20 and 28 that a different value for the adaptation step is used depending on whether the station experiences a More Data event, γ_{MD}, or a No Data event, γ_{ND}. The reason for doing so, is that the value of γ can be used to control whether the estimation, $\Delta_{trig}(n)$, converges to Δ_{DL} remaining always in the More Data zone, i.e., the zone where $\Delta_{trig}(n) > \Delta_{DL}$, or in the No Data zone, i.e., the zone where $\Delta_{trig}(n) < \Delta_{DL}$[2]. The choice of configuring the algorithm to operate in the More Data zone or in the No Data zone represents a trade-off between the possibility of generating an excess of signaling load, if it operates in the No Data zone, or temporarily allowing a delay for some frames above the downlink interarrival time, if it operates in the More Data zone. In order to avoid useless signaling that might increase the congestion in the network and harm the power consumption, we propose to configure the algorithm to operate in the More Data zone.

Based on the previous equations, our proposed algorithm will update the estimation every time an event occurs according to:

$$\Delta_{trig}(n+1) = \Delta_{trig}(n) - \gamma(\Delta_{trig}(n) - \frac{\Delta_t(n)}{n\_fr\_rcvd(n)}) \tag{3.9}$$

Second, it can be observed between lines 8 and 17, that a special behaviour is considered if no frames have been received between the current event and the previous one ($n\_fr\_rcvd = 0$), or if more than two frames have been retrieved from the AP during this SP ($m > 2$). Both cases represent the situation of having a current estimation, $\Delta_{trig}(n)$, far from Δ_{DL}. In the former case the current value of the estimation is below the downlink interarrival time but, since no frames have been retrieved between events, the station has no information to estimate Δ_{DL}. In this case the current estimation is updated by doing $\Delta_{trig}(n+1) \leftarrow \beta\Delta_{trig}(n)$, where $\beta > 1$. In the latter case, the current estimation, $\Delta_{trig}(n)$, is above Δ_{DL}. Thus, after applying a memory to reduce spureous cases ($consec\_long\_burst$), the estimation is quickly decreased by doing $\Delta_{trig}(n+1) \leftarrow \frac{\Delta_{trig}(n)}{m}$. Note that if $m > 2$ the station knows that it is holding an estimation at least m times above the downlink interarrival time.

Finally, the boolean variables introduced in lines 19 and 27, $More\_Data\_update$ and No_Data_update, ensure that the estimation is updated always between consecutive events of the same zone (More Data zone or No Data zone). The reason for this is that a too short interval, Δ_t, between events can occur

[1]We extend the U-APSD definition of Service Period in the context of this chapter to be used also in the case of 802.11 power save mode. Service Period is defined as the period of time that starts when a station sends a trigger frame and ends when a frame with EOSP=1 in case of U-APSD or MD=0 in case of legacy power save mode is received from the AP.

[2]A more detailed definition of the More Data and No Data zones is provided in Section 3.3.

Algorithm 3.1: Routine executed in the station to update its current polling interval, $\Delta_{trig}(n)$.

```
– Variables definition
$\Delta_t \leftarrow$ Time between the current event and the previous one
$n\_fr\_rcvd \leftarrow$ Number of frames received between the current event and the previous one
$m \leftarrow$ Number of frames received during the Service Period (SP)
$\Delta_{trig} \leftarrow$ Polling interval used in the station
– Routine executed when a SP completes
if $m = 1$ then
    $n\_fr\_rcvd \leftarrow n\_fr\_rcvd + 1$
else if $m$, $n\_fr\_rcvd = 0$ and last trigger was signaling then
    $\Delta_{trig}(n+1) \leftarrow \beta\Delta_{trig}(n)$
    $n\_fr\_rcvd \leftarrow 0$
else if $m > 2$ then
    if $consec\_long\_burst > cons\_long\_burst\_MAX$ then
        $\Delta_{trig}(n+1) \leftarrow \frac{\Delta_{trig}(n)}{m}$
        $consec\_long\_burst \leftarrow 0$
    else
        $consec\_long\_burst \leftarrow consec\_long\_burst + 1$
else
    if $m > 1$ then
        if $More\_Data\_update$ is $TRUE$ then
            $\Delta_{trig}(n+1) \leftarrow \Delta_{trig}(n) - \gamma_{MD}(\Delta_{trig}(n) - \frac{\Delta_t}{n\_fr\_rcvd})$
            Reschedule next signaling trigger transmission
        else
            $More\_Data\_update \leftarrow TRUE$
        $No\_Data\_update \leftarrow FALSE$
        $n\_fr\_rcvd \leftarrow 0$
    else if $m = 0$ and last trigger was signaling then
        if $No\_Data\_update$ is $TRUE$ then
            $\Delta_{trig}(n+1) \leftarrow \Delta_{trig}(n) - \gamma_{ND}(\Delta_{trig}(n) - \frac{\Delta_t}{n\_fr\_rcvd})$
            Reschedule next signaling trigger transmission
        else
            $No\_Data\_update \leftarrow TRUE$
        $More\_Data\_update \leftarrow FALSE$
        $n\_fr\_rcvd \leftarrow 0$
$m \leftarrow 0$
```

when the value of the estimation moves from one zone to the other. As it will be seen in the next section, this could lead to a misleading estimation of the current error between $\Delta_{trig}(n)$ and Δ_{DL}.

We consider now the case where EDCA is the channel access function in use and various applications, with different delay requirements, run in the station at the same time through different Access Categories (AC). In this case, the trigger generation algorithm would only generate triggers at the estimated rate of the aplication sending frames more often, i.e. $min\{\Delta_{DL}[i]\}$. The former is possible because the algorithm can rely on the More Data bit and Max_SP_Length capabilities of the existent power saving mechanisms in order to retrieve frames from other downlink streams. To make the overall system adaptive, a simple procedure can be implemented in the station that by keeping track of the frames received from each AC decides which is the AC sending frames more often. The station would then run the algorithm over the selected AC that would also be used to generate signaling triggers.

The complete cycle of activity in a station can be summarized in the following terms. When a station is idle no trigger frames are generated. Once a Beacon frame is received anouncing the presence of buffered frames in the AP, a station starts generating trigger frames with an initial polling interval, $\Delta_{trig}(0)$, that is then dynamically adjusted by means of Algorithm 3.1. After sending a certain number of consecutive trigger frames receiving only empty frames from the AP a station considers that the application has finished and switches back to idle mode again.

Finally, we conclude this section discussing a simple heuristic that can be used to improve the performance of the algorithm when applications sending large packets are considered. Large packets in the application layer, e.g. like those generated by video codecs, are in many cases fragmented resulting in bursts of packets arriving at the AP which can mislead the algorithm into false More Data events. However, if the minimum MTU between the application generating packets and the AP is assumed to be known at the terminal, which with a reasonable probability can be considered to be the Ethernet MTU, the misleading effect of large packets can be reduced by remembering the size in the MAC layer of the packets downloaded during a service period, and considering as a single packet adjacent packets of size equal to the expected MTU[1].

3.2.1 Convergence to a multiple of the interarrival time: $\Delta_{trig} \rightarrow k\Delta_{DL}$

If a delay and jitter above Δ_{DL} can be afforded, further power saving and signaling load enhancements can be obtained by modifying Algorithm 3.1 to converge to a multiple of the downlink interarrival time, $\Delta_{trig} \rightarrow k\Delta_{DL}$, where k is an integer bigger than 1. The larger the value of k can be set, the larger the potential increase in power saving and reduction of signaling load.

In order to achieve this, Equation 3.9 has to be modified such that the equilibrium state is reached when the trigger interval used by a station, Δ_{trig}, equals the desired multiple of the interarrival time:

$$\Delta_{trig}(n+1) = \Delta_{trig}(n) - \gamma(\Delta_{trig}(n) - k\frac{\Delta_t(n)}{n\_fr\_rcvd(n)}) \tag{3.10}$$

In addition, the *No Data* and *More Data* events generation, which trigger an update of the current polling interval, also need to be adjusted accordingly. In this case, *No Data* events will be generated

[1]This heuristic would fail if packets of size equal to the MTU minus overheads are generated by the applications, but we consider this probability to be small.

when less than k frames are received within one service period while *More Data* events will be generated when more than k frames are received within one service period.

In the rest of the chapter we focus in the case of the polling interval converging to the downlink interarrival time, $k = 1$, since it is the one with the most stringent QoS requirements and thus, the most demanding for the proposed adaptive algorithm.

3.3 Algorithm analysis

In this section we use the model presented in Section 3.1 in order to gain a deeper understanding on the properties of the adaptive algorithm presented in the previous section. First, we study under which conditions the algorithm converges. Second, the convergence speed of the algorithm is analyzed providing a worst case bound on the required number of iterations. Finally, we discuss how the performance of the algorithm degrades when the assumptions of the model do not hold.

3.3.1 Proof of Convergence

For the purpose of this analysis, we define *convergence* as the process of making the estimation, $\Delta_{trig}(n)$, continuously approach at each iteration the actual downlink interarrival time, Δ_{DL}, while always guaranteeing $\Delta_{trig}(n) \geq \Delta_{DL}$. Some of the reasons for preferring to keep $\Delta_{trig}(n) \geq \Delta_{DL}$ have already been provided in Section 3.2. Additionally, further reasons will be given in this section. The same analysis presented here though, could be also applied if it would be preferred that the algorithm converges to Δ_{DL} guaranteeing $\Delta_{trig}(n) \leq \Delta_{DL}$.

Throughout the analysis we refer to the set of values of the estimation, $\Delta_{trig}(n)$, that are above Δ_{DL} as the *More data* zone, since only More Data events can be observed by the station in this zone. Similarly, the set of values of the estimation below Δ_{DL} are referred to as the *No Data* zone. The different zones of convergence are depicted in Figure 3.3.

To demonstrate the convergence of the algorithm we will prove that if the current value of the estimation, $\Delta_{trig}(n)$, is above the downlink interarrival time, Δ_{DL}, the next estimation update will result in a smaller value, $\Delta_{trig}(n+1)$, either above or equal to Δ_{DL}. It is important to note though that when the algorithm starts the station's initial polling interval, $\Delta_{trig}(0)$, could be below the downlink interarrival time. If the initial value is below the downlink interarrival time, appropriate mechanisms can be designed to ensure that we will move to the More Data zone. The proof of convergence of this section guarantees that once we enter into the More Data zone we will not leave it anymore and at each iteration the estimation will get closer to the actual Δ_{DL} value.

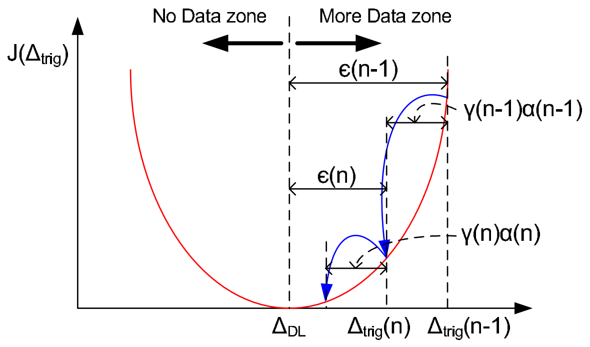


Figure 3.3: Evolution of $\epsilon(n)$ along the error function.

We start considering the more general case where the station reuses uplink data frames as triggers and the re-scheduling mechanism is used. Results regarding the no data triggers case can be obtained particularizing the presented analysis. The error incurred by a station at each estimation update is defined as $\epsilon(n) = \Delta_{trig}(n) - \Delta_{DL}$. Hence, if it holds that $0 \leq \frac{\epsilon(n+1)}{\epsilon(n)} < 1 \; \forall n$, the estimation will get closer to Δ_{DL} every time it is updated. The former can be proved by separately demonstrating:

1. $\frac{\epsilon(n+1)}{\epsilon(n)} < 1$. This condition ensures that the error ϵ reduces at each iteration.
2. $\frac{\epsilon(n+1)}{\epsilon(n)} \geq 0$. This condition ensures that the algorithm remains always in the same zone, either the More Data zone or the No Data zone.

We start proving that $\frac{\epsilon(n+1)}{\epsilon(n)} < 1 \; \forall n$. Figure 3.4 shows two consecutive events that update the estimation. In the figure, Δ_{DL} represents the interarrival time of the downlink stream and Δ_{trig} and Δ_{UL} represent the generation interval of signaling and data triggers, respectively. Note that $\delta = \lfloor \frac{\Delta_{UL}}{\Delta_{trig}} \rfloor$ signaling triggers are sent between two consecutive data triggers in the uplink and the time between a signaling trigger and an uplink data frame is $\Delta_{res} = \Delta_{UL} \bmod \Delta_{trig}$.

In Figure 3.4 the two signaling triggers marked with a circle represent More Data events that trigger an estimation update. In the first event, d_{01} represents the time between the trigger frame sent by the station and the arrival time of the last frame retrieved by this trigger from the AP. Note that since this is a More Data event and the station polls the AP every $\Delta_{trig}(n-1)$, $0 \leq d_{01} < \epsilon(n-1)$. An analog definition is used for d_{02} in the second event, the difference in this case is that $0 \leq d_{02} < \epsilon(n)$. In addition, Δ_t represents the time between the two consecutive events, Δ'_t represents the time between the first of the events and the last uplink data trigger sent before the second event, and k represents the number of triggers sent after Δ'_t until the second event occurred, with $1 \leq k \leq \delta$ [1].

[1]We assume in this case that the second More Data event is triggered by a signaling frame. The same analysis applies if this event is triggered by a data frame, replacing $k\Delta_{trig}$ by $\delta\Delta_{trig} + \Delta_{res}$.

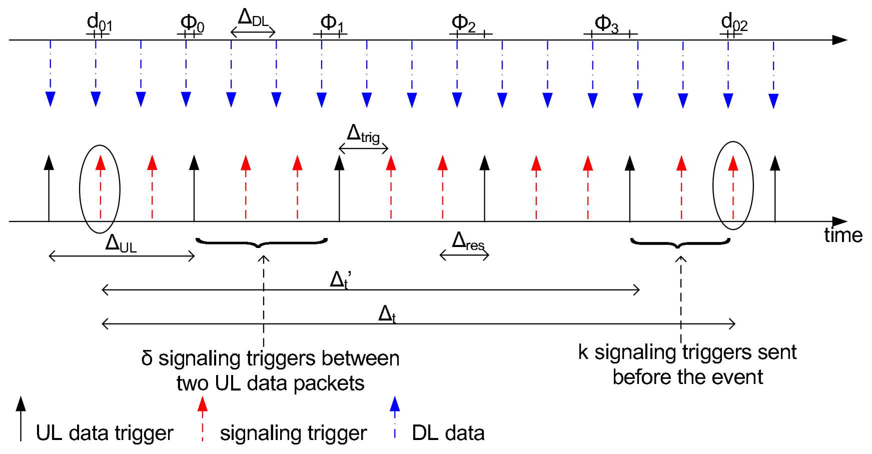


Figure 3.4: Timing between consecutive events.

Recalling now the definition of $\epsilon(n)$, Equation 3.9 and considering the current estimation in the More Data zone, it can be observed that $\frac{\epsilon(n+1)}{\epsilon(n)} < 1\ \forall n$ if and only if:

$$\Delta_{trig}(n+1) < \Delta_{trig}(n) \iff \Delta_{trig}(n) > \frac{\Delta_t(n)}{n\_fr\_rcvd(n)}$$

An upper bound on $\frac{\Delta_t(n)}{n\_fr\_rcvd(n)}$ expressed in terms of $\Delta_{trig}(n)$ is needed in order to prove the previous statement. Considering again Figure 3.4, $\frac{\Delta_t(n)}{n\_fr\_rcvd(n)}$ can be upper bounded in the following way:

$$\frac{\Delta_t}{n\_fr\_rcvd} = \frac{\Delta_t' + k\Delta_{trig}}{\lfloor \frac{d_{01}+\Delta_t'}{\Delta_{DL}} \rfloor + k + 1} \leq \frac{\Delta_t' + k\Delta_{trig}}{\lceil \frac{\Delta_t'}{\Delta_{DL}} \rceil + k} \tag{3.11}$$

Where the temporal index n has been ommited for clarity. Thus, $\Delta_{trig}(n) > \frac{\Delta_t(n)}{n\_fr\_rcvd(n)}$ can be proved by comparing $\Delta_{trig}(n)$ with the upper bound for $\frac{\Delta_t(n)}{n\_fr\_rcvd(n)}$ obtained in Equation 3.11:

$$\frac{\Delta_t' + k\Delta_{trig}}{\lceil \frac{\Delta_t'}{\Delta_{DL}} \rceil + k} < \Delta_{trig} \iff \frac{\Delta_t'}{\Delta_{trig}} < \lceil \frac{\Delta_t'}{\Delta_{DL}} \rceil \tag{3.12}$$

Where the previous inequality is always satisfied because we are considering the case where $\Delta_{trig}(n) > \Delta_{DL}$.

So far it has been proved that the algorithm always updates the estimation in the correct direction when a More Data event is observed. Next, we prove that the adaption step, γ, can ensure that the estimation, $\Delta_{trig}(n)$, does not bounce between the More Data and the No Data zones, i.e., $\frac{\epsilon(n+1)}{\epsilon(n)} \geq 0$.

Considering Equation 3.9, it can be seen that the *effective* step taken by the algorithm to update the current estimation is $\gamma(n)\alpha(n)$, with $\alpha(n) = \Delta_{trig}(n) - \frac{\Delta_t(n)}{n\_fr\_rcvd(n)}$. From Figure 3.3 and assuming a value of the estimation in the More Data zone, it can be observed that the updated value of the estimation,

$\Delta_{trig}(n+1)$, will remain in the More Data zone if and only if the step taken by the algorithm at this iteration is smaller than the current error, i.e., $\gamma(n)\alpha(n) \leq \epsilon(n)$.

In order to obtain an upper bound on $\gamma(n)$ that ensures $\Delta_{trig}(n+1) \geq \Delta_{DL}$, note first that based on the relations illustrated in Figure 3.4, $\frac{\Delta_t(n)}{n\_fr\_rcvd(n)}$ can be expressed as:

$$\frac{\Delta_t(n)}{n\_fr\_rcvd(n)} = \Delta_{DL} + \frac{d_{02} - d_{01}}{n\_fr\_rcvd(n)} \tag{3.13}$$

Recalling the definition of $\alpha(n)$ and the previous equation we can express $\alpha(n)$ as:

$$\alpha(n) = \Delta_{trig}(n) - \frac{\Delta_t(n)}{n\_fr\_rcvd(n)} = \epsilon(n) - \frac{d_{02} - d_{01}}{n\_fr\_rcvd(n)} \tag{3.14}$$

Considering now that $0 \leq d_{01} < \epsilon(n-1)$ and that, from Equation 3.9, $\epsilon(n-1)$ can be expressed as $\epsilon(n-1) = \epsilon(n) + \gamma(n-1)\alpha(n-1)$, the following upper bound on $\alpha(n)$ can be obtained:

$$\alpha(n) < \epsilon(n)\frac{n\_fr\_rcvd(n) + 1}{n\_fr\_rcvd(n)} + \frac{\gamma(n-1)\alpha(n-1)}{n\_fr\_rcvd(n)}$$

Thus, the previous upper bound on $\alpha(n)$ can be turned into a lower bound on $\epsilon(n)$ and recalling the condition to keep the next value of the estimation in the More Data zone, $\gamma(n)\alpha(n) < \epsilon(n)$, a conservative $\gamma(n)$ can be found, $\gamma_{CONS}(n)$, that keeps the estimation always in the desired zone of convergence:

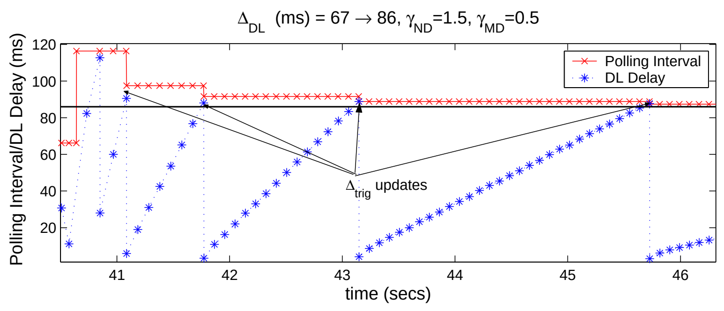
$$\gamma(n) < \frac{n\_fr\_rcvd(n) - \gamma(n-1)\frac{\alpha(n-1)}{\alpha(n)}}{n\_fr\_rcvd(n) + 1} = \gamma_{CONS}(n) \tag{3.15}$$

Continuing with the previous reasoning and going back to Equation 3.14 while recalling that $0 \leq d_{02} < \epsilon(n)$, a lower bound on $\alpha(n)$ can be obtained. Applying the same analysis to this bound, an aggressive bound on $\gamma(n)$ that forces the estimation to jump from one zone to the other can also be obtained, being:

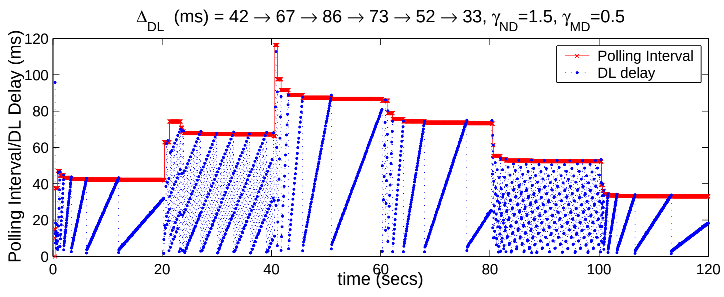
$$\gamma(n) > \frac{n\_fr\_rcvd(n)}{n\_fr\_rcvd(n) - 1} = \gamma_{AGGR}(n) \tag{3.16}$$

Although the previous bounds, $\gamma_{CONS}(n)$ and $\gamma_{AGGR}(n)$, have been derived under the assumption of having the estimation in the More Data zone, Equations 3.15 and 3.16 hold true also in the No Data zone.

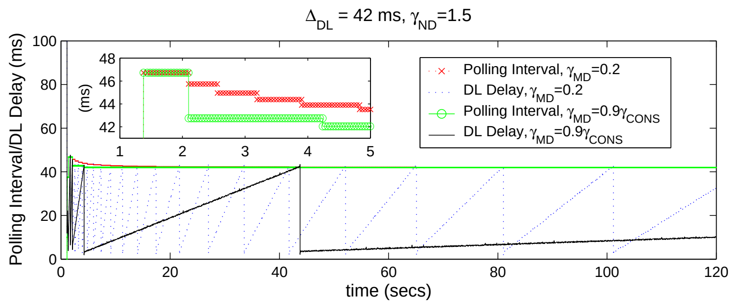
To illustrate how the algorithm updates the estimation while keeping always a value in the More Data zone, Figure 3.5(a) depicts the delay and polling interval experienced by a Wi-Fi station using U-APSD that retrieves a CBR stream from the AP. The values of γ in the More Data and No Data zone, γ_{MD} and γ_{ND}, are constant and chosen below and above the corresponding γ_{CONS} and γ_{AGGR} thresholds. From the figure, it is worth noting how the time between events increases as the error in the estimation reduces, as predicted by the model introduced in Section 3.1. This property of the algorithm has the



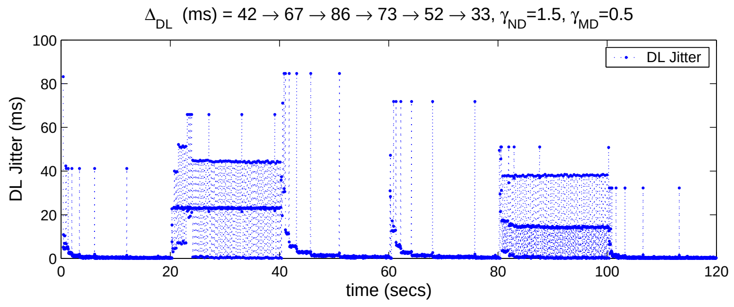
(a) Δ_{trig} adaptation to Δ_{DL}.



(b) Effect of Δ_{DL} variation on Delay and Δ_{trig} adaptation.



(c) Effect of γ on Δ_{trig} adaptation.



(d) Effect of Δ_{DL} variation on Jitter.

Figure 3.5: Δ_{trig} dynamics.

interesting consequence that, in the limit, the sequence of uplink triggers generated by the station and downlink frames arriving at the AP would synchronize, i.e., $\lim_{t\to\infty} d(k) = 0$. Note though, that if the algorithm would converge from the No Data zone, the behaviour in the limit would be different, i.e. $\lim_{t\to\infty} d(k) = \Delta_{DL}$. Hence this is another reason to make the algorithm converge from the More Data zone.

It has been proved so far that $\epsilon(n)$ reduces at each estimation update. In order to update the estimation though, the station has to keep observing new events. Hence, a new question arises that is whether the station will always observe enough events to make the estimation converge to Δ_{DL}. We prove in the following paragraphs that:

1. If no data triggers in the uplink are considered, the proposed algorithm always converges to Δ_{DL}.

2. If data triggers in the uplink are considered, the proposed algorithm can converge to a compact set of values around Δ_{DL}. In this case though, all the algorithm's requirements described in section 3.2 are also fulfilled.

To prove the previous statements, the case where the station generates uplink data triggers is considered. The results obtained are afterwards particularized to the case with no data triggers in the uplink.

The *converged* values of $\Delta_{trig}(n)$ are defined as those values that prevent the station from observing any further event. In order to find those values we define ϕ_i, $0 \leq \phi_i < \Delta_{DL}$, which represent the time difference between an uplink data trigger sent by a station and the last downlink arrival at the AP. See Figure 3.4 for a graphical representation.

As mentioned in Section 3.1, the compound sequence of uplink triggers and downlink frames in our model is periodic, with period $T = lcm(\Delta_{UL}, \Delta_{DL})$. T can hence be understood as composed of $N = \frac{T}{\Delta_{UL}}$ consecutive uplink subperiods, where each uplink superiod is formed by an uplink data trigger and $\delta = \lfloor \frac{\Delta_{UL}}{\Delta_{trig}} \rfloor$ consecutive signaling triggers and is shifted $\phi_i = (\phi_0 + i\Delta_{UL}) \bmod \Delta_{DL}$ with respect to the sequence of downlink data frames. Notice that ϕ_0 corresponds to the shift experienced by a reference uplink subperiod.

If the station holds a value of $\Delta_{trig}(n)$ such that no event is observed within T, $\Delta_{trig}(n)$ is a converged value of the estimation. Indeed, not experiencing any event in T means not experiencing any event in any of the N uplink subperiods that conform T.

It is proved in the Appendix at the end of this chapter that in the general case where Δ_{UL} is not multiple of Δ_{DL}, the station has to generate $\delta = \lfloor \frac{\Delta_{UL}}{\Delta_{DL}} \rfloor$ signaling triggers between uplink data triggers to avoid observing any event during T. Hence, a necessary condition for $\Delta_{trig}(n)$ being a converged polling interval is $\delta = \lfloor\frac{\Delta_{UL}}{\Delta_{trig}(n)}\rfloor = \lfloor\frac{\Delta_{UL}}{\Delta_{DL}}\rfloor$, or:

$$\frac{\Delta_{UL}}{\lfloor\frac{\Delta_{UL}}{\Delta_{DL}}\rfloor + 1} < \Delta_{trig}(n) < \frac{\Delta_{UL}}{\lfloor\frac{\Delta_{UL}}{\Delta_{DL}}\rfloor}$$

Considering that $\Delta_{trig}(n) > \Delta_{DL}$, we continue proving that a $\Delta_{trig_{max}}$ exists such that if $\Delta_{DL} < \Delta_{trig}(n) < \Delta_{trig_{max}}$ the station will never observe an event.

In order to find $\Delta_{trig_{max}}$, recall from Equation 3.1 that if no event is observed, the delay experienced by consecutive downlink frames can be expressed like $d_0 + k\epsilon(n)$, with $\epsilon(n) = \Delta_{trig}(n) - \Delta_{DL}$. Therefore, if $\delta = \lfloor\frac{\Delta_{UL}}{\Delta_{DL}}\rfloor$ signaling triggers are sent between two consecutive data triggers, the following condition must hold in order to avoid More Data events in any uplink subperiod:

$$\phi_i + \epsilon(n)\lfloor\frac{\Delta_{UL}}{\Delta_{DL}}\rfloor < \Delta_{DL} \iff \Delta_{trig}(n) \quad < \quad \Delta_{DL} + \frac{\Delta_{DL} - \phi_i}{\lfloor\frac{\Delta_{UL}}{\Delta_{DL}}\rfloor}$$

The previous expression implicitly assumes that the time between a signaling trigger and an uplink data trigger, Δ_{res}, is below Δ_{DL}. Otherwise, applying the same analysis, a tighter bound on $\Delta_{trig}(n)$ with the same structure would be found. The most restrictive condition on $\Delta_{trig}(n)$ is hence imposed by the uplink subperiod with the biggest ϕ_i. Recalling Equation 3.4, ϕ_{max} can be found to be $\phi_{max} = \Delta_{DL} - (gcd(\Delta_{UL}, \Delta_{DL}) - \phi_{min})$. Thus, $\Delta_{trig_{max}}$ can be written as:

$$\Delta_{trig_{max}} = min\{\frac{\Delta_{UL}}{\lfloor\frac{\Delta_{UL}}{\Delta_{DL}}\rfloor}, \Delta_{DL} + \frac{\Delta_{DL} - \phi_{max}}{\lfloor\frac{\Delta_{UL}}{\Delta_{DL}}\rfloor}\}$$

Applying the same analysis when $\Delta_{trig}(n) < \Delta_{DL}$, it can be found that no event ever occurs if $\Delta_{trig_{min}} < \Delta_{trig}(n) < \Delta_{DL}$, where:

$$\Delta_{trig_{min}} = max\{\frac{\Delta_{UL}}{\lfloor\frac{\Delta_{UL}}{\Delta_{DL}}\rfloor + 1}, \Delta_{DL} - \frac{\phi_{min}}{\lfloor\frac{\Delta_{UL}}{\Delta_{DL}}\rfloor}\}$$

Note that in the general case, Δ_{UL} not multiple of Δ_{DL}, the amount of uplink data triggers that will download a data packet from the AP, i.e., those uplink data packets that are effectively reused as triggers, will be always the same and correspond to the number of uplink subperiods where $\lceil\frac{\Delta_{UL}}{\Delta_{DL}}\rceil$ downlink frames arrive at the AP. This number is found in the Appendix at the end of this chapter.

In the particular case where Δ_{UL} is a multiple of Δ_{DL}, the same analysis can be applied to find a new converged value of Δ_{trig} located above Δ_{DL}, $\Delta'_{trig_{max}}$, such that only $\delta = \lfloor\frac{\Delta_{UL}}{\Delta_{DL}}\rfloor - 1$ signaling triggers are generated by a station between uplink data triggers. This is again another argument to prefer the convergence of the algorithm from the More Data zone.

Note that regardless of the final converged value estimated in the station, the original requirements of the algorithm are always being fulfilled by the way the events have been defined. First, if a More

Data event never occurs, the delay is effectively bounded to the downlink interarrival time. Second, if a No Data event never occurs, no useless signaling is generated.

The converged values of $\Delta_{trig}(n)$ when no uplink data frames are reused as triggers can be obtained from the expressions of $\Delta_{trig_{min}}$ and $\Delta_{trig_{max}}$ by observing that:

$$\lim_{\Delta_{UL}\to\infty} \Delta_{trig_{min}} = \lim_{\Delta_{UL}\to\infty} \Delta_{trig_{max}} = \Delta_{DL}$$

Therefore, without uplink data triggers, the only converged value of $\Delta_{trig}(n)$ that refrains the station from observing any event is $\Delta_{trig}(n) = \Delta_{DL}$.

Finally, to illustrate the analysis of convergence introduced in this section, Figures 3.5(b) and 3.5(d) depict the delay and jitter experienced by a downlink stream whose interarrival time is changed in a step-like fashion. The Wi-Fi station is configured as in the experiment for Figure 3.5(a) and generates data triggers in the uplink every 90ms during the time intervals (20,40) and (80,100) seconds.

As predicted in the analysis, it can be observed by the constantly decreasing slope of the delay curves that, when no uplink data triggers are considered, the algorithm constantly approaches Δ_{DL}. In the time intervals where the station sends uplink data triggers though, the predicted periodic situation in the channel is observed. Note that the different jitter levels observed in Figure 3.5(d) correspond to the different values predicted in the model introduced in Section 3.1.

3.3.2 Speed of Convergence

In the previous section it has been proved that the proposed algorithm always converges to a value that fulfills our design objectives. In this section we study the speed of the algorithm to converge to such a value.

Our goal is to find an upper bound on the number of updates needed in order to reduce an initial error in the estimation $\epsilon(0)$ to $\epsilon(n) < \alpha\Delta_{DL}$, where α is a variable defined for the purpose of this analysis, $0 < \alpha < 1$.

Assuming that our algorithm operates in the More Data zone, as recommended, and recalling Equation 3.13, the error experienced in the next estimation update can be expressed and upper bounded in the following way:

$$\epsilon(n+1) \quad = \quad \epsilon(n)(1-\gamma) + \gamma\frac{d_{02} - d_{01}}{n\_fr\_rcvd(n)} < \epsilon(n)(1 - \gamma\Psi(n))$$

Where $\Psi(n) = \frac{n\_fr\_rcvd(n)-1}{n\_fr\_rcvd(n)}$ and $0 \leq d_{02} < \epsilon(n)$. Note that $\Psi(n)$ is a monotonically growing function of $n\_fr\_rcdv(n)$ and since the minimum number of frames that a station can retrieve from the AP between two consecutive More Data events is 2 then $\frac{1}{2} < \Psi(n) < 1$.

Based on that, a lower bound on the speed with which the error decreases can be obtained as:

$$\frac{\epsilon(n+k)}{\epsilon(n)} < \prod_{j=0}^{k-1}(1-\gamma\Psi(n+j)) < (1-\frac{1}{2}\gamma)^k$$

Where a constant adaptation step, γ has been assumed. If γ would be adaptive, its minimum value should be considered in order to obtain the previous bound.

Note that the adaptation step, γ, can be effectively used to control the speed of convergence. Nevertheless, an upper bound, γ_{CONS}, was found in the previous section in order to guarantee the convergence keeping the value of the estimation always in the same zone. A trade-off between speed of convergence and stability controlled by the adaptation step is an intrinsic characteristic of the steepest descent algorithm.

If the algorithm converges faster than $(1-\frac{1}{2}\gamma)^k$, an upper bound on the number of updates needed to reduce the error of the estimation from $\epsilon(0)$ to $\alpha\Delta_{DL}$, can be found as:

$$K_{max} = -\log(\frac{\epsilon(0)}{\alpha\Delta_{DL}})\frac{1}{\log(1-\frac{1}{2}\gamma)}$$

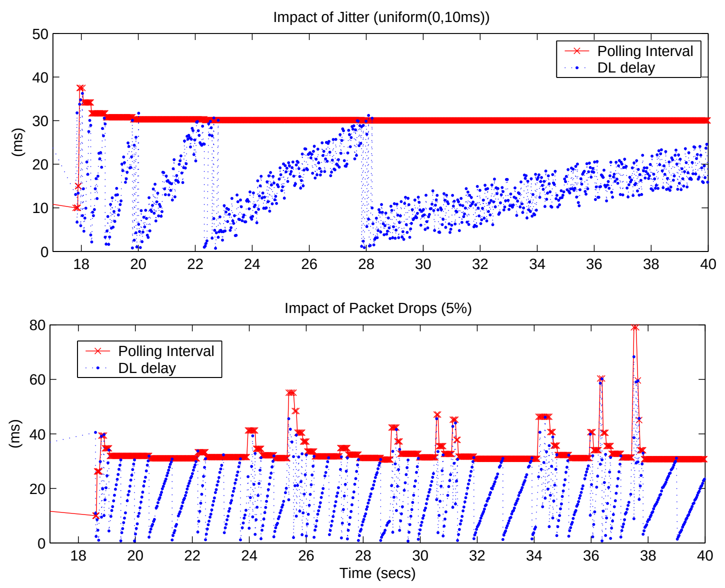
The former upper bound holds true regardless of whether the station generates uplink data triggers or not. The difference will be on the time needed to reach the critical number of estimation updates, K_{max}. The reason is that the time between consecutive events is different depending on whether uplink data triggers are considered or not. In any case though, the time between events increases when the error in the estimation, $\epsilon(n)$, decreases.

The benefits of updating the estimation in a way proportional to $\frac{1}{\epsilon(n)}$ are indeed twofold. On the one hand, the algorithm will react fast when there are sudden changes in the downlink interarrival time, because the error, $\epsilon(n)$, will increase. On the other hand, if the downlink interarrival time remains stable, the algorithm minimizes the required computational load by relaxing the frequency of updates as $\epsilon(n)$ decreases.

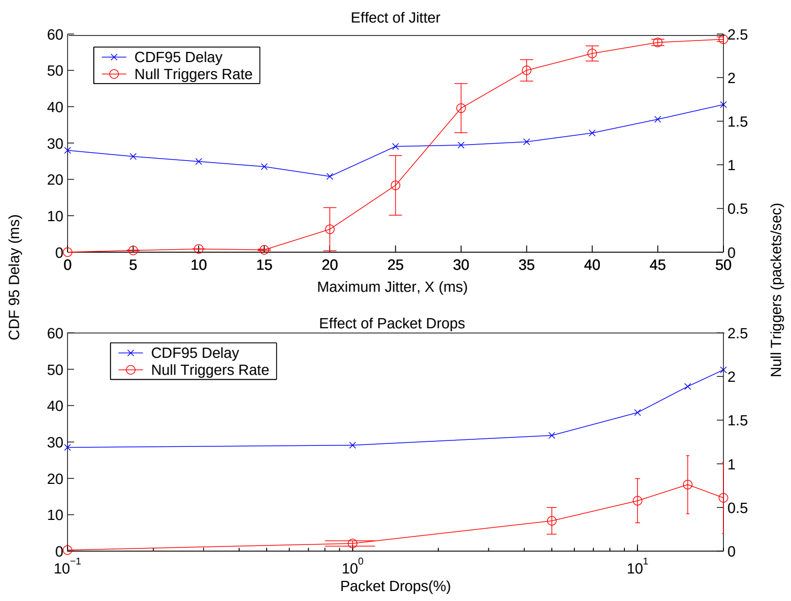
Finally, to illustrate how the adaptation step can be used to tune the speed of convergence, in Figure 3.5(c) we show how a Wi-Fi station using U-APSD estimates an interarrival time of 42ms using two different values of γ_{MD}, $\gamma_{MD} = 0.2$ and $\gamma_{MD} = 0.9\gamma_{CONS}$. A noticeable faster estimation is observed when $\gamma_{MD} = 0.9\gamma_{CONS}$, while keeping the estimation always in the More Data zone.

3.3.3 Relaxing the Assumptions of the Model

In order to derive the convergence properties of our adaptive algorithm we have assumed so far that the downlink and uplink data streams were perfectly periodic. However, this strong assumption only holds



(a) Dynamics of the algorithm in presence of jitter or drops.



(b) Performance with varying jitter and drops.

Figure 3.6: Behavior of the algorithm when relaxing the assumptions of the model.

in reality when congestion in the wired and wireless parts of the network is kept very low. In this section we take a simulative approach in order to analyze the degradation experienced by the algorithm when the previous assumptions are not valid.

We consider the same scenario used for Figure 3.5(a), but having a downlink stream with an inter-arrival time of 30ms and introducing behind the AP a virtual node which delays every incoming packet by a random value uniformly distributed between 0 and Jms, and drops every incoming packet with probability P. Thus, we want to study the robustness of the adaptive algorithm to increasing values of J and P.

Figure 3.6(a) depicts the dynamics of the algorithm in the presence of jitter (upper graph) and packet drops (lower graph). The first remarkable point is that as observed in both graphs the essential behavior predicted in Section 3.1, a saw-tooth pattern for delay, is maintained in the presence of jitter and packet drops. The actual dynamics though slightly differ in either case. In both cases, jitter or drops, if a trigger is sent and no frame is present in the AP the algorithm will experience a No Data event. On the one hand, if the missing frame was delayed due to jitter, the next time the station triggers the AP it will probably retrieve two frames, thus experiencing a More Data event. Since the algorithm is designed to only update the estimation between consecutive events of the same type (see Algorithm 3.1), No Data events and More Data events cancel out in the case of jitter resulting in a very stable estimation. On the other hand, if the No Data event was due to a packet drop the algorithm will not experience a More Data event in the next trigger which results in sporadic increases of the estimation above the target polling interval.

To better quantify the effects of jitter and packet drops on the algorithm performance we increase J from $0ms \rightarrow 50ms$ and P from $0.1\% \rightarrow 20\%$. This range for jitter and drops covers the margins recommended by ITU-T in (63). Figure 3.6(b) illustrates for this experiment the worst case downlink delay and the amount of *null* triggers (triggers sent by the station which find no data in the AP). The results show how the algorithm can maintain a bounded delay for the downlink stream for a large range of jitter and drops, $J < 40ms$ and $P < 10\%$. However, as the jitter and packet drops increase, degradation unavoidably appears in terms of null triggers which could result in an increased level of congestion in the network.

3.4 Performance evaluation

In this section we present a performance evaluation of our proposed adaptive algorithm. We divide this performance evaluation in two subsections. First, subsection 3.4.1 discusses the behavior of our adaptive algorithm when used with different applications in a typical Wi-Fi deployment. Second, subsection 3.4.2 analyzes the scalability and peformance of our adaptive algorithm as congestion in the Wi-Fi network increases.

The analysis has been performed via simulations. We extended the 802.11 libraries provided by OPNET (64) to include the power saving extensions defined in 802.11 and 802.11e and our proposed adaptive trigger generation algorithm.

We consider an infrastructure mode Wi-Fi network, with an AP beacon interval of 100ms. The physical layer chosen in our simulations is 802.11b with all stations transmitting at 11Mbps.

Next, we present a definition of a *traffic mix* that we consider representative of todays typical Wi-Fi deployments. Throughout this section we will consider stations using one/various of the applications that conform this traffic mix. Sometimes we will use the concept of a *cluster* of stations, which we define as a group of four stations where each station runs one of the applications defined in our traffic mix. The applications used in our traffic mix together with the EDCA Access Category used to transmit traffic of this application are described in Table 3.1.

	Description	**AC**
Voice	G.711 Voice codec with silence suppression. Data rate: 64kbps. Frame length: 20ms. Talk spurt exponential with mean 0.35s and silence spurt exponential with mean 0.65s.	AC_VO
Video	Streaming video download. MPEG-4 real traces of the movie 'Star Trek: First Contact' obtained from (65). Target rate: 64kbps. Frame generation interval: 40ms.	AC_VI
Web	Web traffic. Page interarrival time exponentially distributed with mean 60s. Page size 10KB plus 1 to 5 images of a size uniformly distributed between 10KB and 100KB.	AC_BE
FTP	FTP download. File size of 1 MB. Inter-request time exponentially distributed with mean 60s.	AC_BK

Table 3.1: Applications Description.

For our experiments we consider that stations in the Wi-Fi network communicate through the AP with stations in a wired domain. The wired domain is modeled as an Ethernet segment behind the AP and a virtual node which represents the backbone of the wired network and introduces jitter and drops. Based on (63) we consider that the Voice and Video traffic traversing the backbone network suffer a maximum delay variation of $5ms$ and a 0.1% packet drop probability. Additionally Web and

FTP applications communicate with a server using TCP New Reno and experiencing an RTT of 20ms (typical RTT in a domestic internet path (86)).

As previously mentioned the channel access function considered in our simulations is EDCA. We assume a fixed configuration of the EDCA QoS parameters based on the 802.11e standard recommendation (32). The parameters used are detailed in Table 3.2.

EDCA	**AIFS**	**CWmin**	**CWmax**	**TXOP length**
AC_VO	2	31	63	3.264 ms
AC_VI	2	63	127	6.016 ms
AC_BE	3	127	1023	0
AC_BK	7	127	1023	0

Table 3.2: EDCA configuration for the different ACs.

Throughout this section our adaptive algorithm will be configured in the following way. The initial polling interval, $\Delta_{trig}(0)$, is set to 10ms, $consecutive\_long\_burst$ is 2, β is set to 1.5 and the algorithm switches off after sending three consecutive triggers without finding data in the AP.

In addition we will consider that all the trigger generation algorithms studied in this section use U-APSD, configured with *Max_SP_Length* set to *All*, instead of Legacy 802.11 Power Save Mode, due to its better support for QoS.

3.4.1 Dynamics of the algorithm with realistic applications

In order to study how our adaptive algorithm behaves with different applications we consider the following scenario. A set of five clusters, which results in a total of 20 stations in the network, generate traffic according to our defined traffic mix. This traffic is considered to be background load in the Wi-Fi network. In addition we consider a tagged station configured to run the specific application that we want to study. In this section we will consider this station running the following applications: i) VoIP with silence suppression, ii) Video Streaming, iii) TCP download and iv) a mix of different applications.

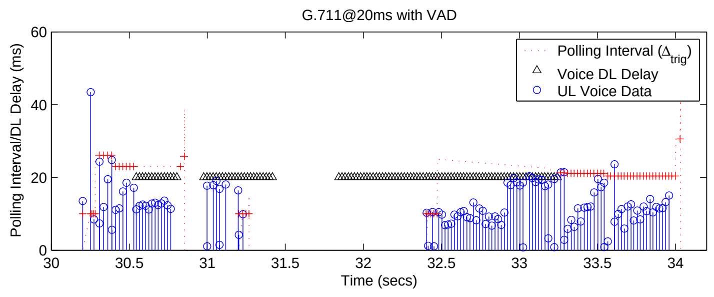
3.4.1.1 VoIP with VAD

A G.711 voice codec with voice activity detection (VAD) has been considered in our experiments.

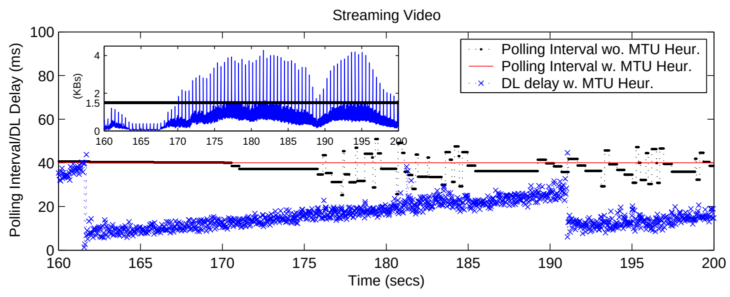
The instantaneous behaviour of the adaptive algorithm is illustrated in Figure 3.7(a). The dynamics of the algorithm can be summarized in the following terms. First, when there is a burst of voice packets in the downlink, e.g. around 30.2 or 33.5 seconds, the station applies the adaptive algorithm to adjust its polling interval that converges to 20ms. Second, after sending 3 consecutive triggers without response, the algorithm infers that there is a silence gap and cancels the generation of triggers, as observed at 34 seconds. Switching off the algorithm though, can turn into an increased delay for the first voice samples when the activity restarts, as seen at 30 seconds in the trace. Finally, when the application generates data frames in the uplink, the signaling triggers are deferred by means of the re-scheduling mechanism.

3.4.1.2 Video Streaming

The behaviour of the adaptive algorithm in case of Video Streaming is depicted in Figure 3.7(b). The MPEG-4 codec generates frames at a nominal interarrival time of 40ms. These frames though, may need to be fragmented (MTU of Ethernet) and as result the AP can receive bursts of frames every 40ms.



(a) Adaptive algorithm behavior with VoIP and VAD.



(b) Adaptive algorithm behavior with MPEG 4 Video Streaming.

Figure 3.7: Dynamics of the adaptive algorithm with a VoIP codec with silence suppression (upper graph) and with a Video codec (lower graph).

Figure 3.7(b) illustrates the instantaneous polling interval used in the station with and without the MTU heuristic defined in Section 3.2, the MAC delay experienced by the video frames retrieved from the AP and in a subgraph, the instantaneous evolution of the frame sizes generated by the MPEG-4 codec. It can be seen that before 170 seconds, when the size of the video frames is most of the time below the fragmentation threshold, the algorithm maintains a very stable value of the estimation regardless of whether the MTU heuristic is used or not. From 170 seconds on though, the codec generates a burst of frames above the fragmentation threshold, making the frame interarrival time at the AP deviate from 40ms. In this case, if the MTU heuristic is not applied the estimated polling interval shakes around 40ms. If the MTU heuristic is applied though, the estimation remains pefectly stable at 40ms.

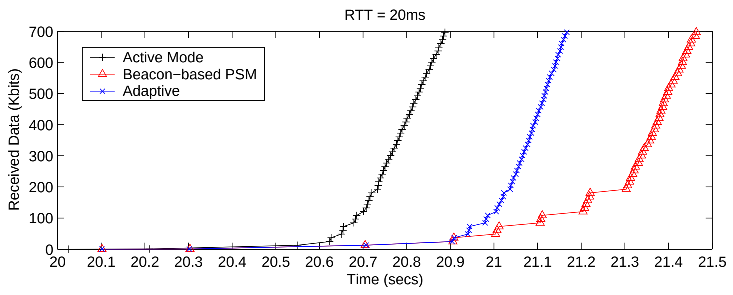
3.4.1.3 TCP application

In order to illustrate the effect of the adaptive algorithm on applications running over TCP, we consider in this section that our station under study performs a TCP download of a 500KB file.

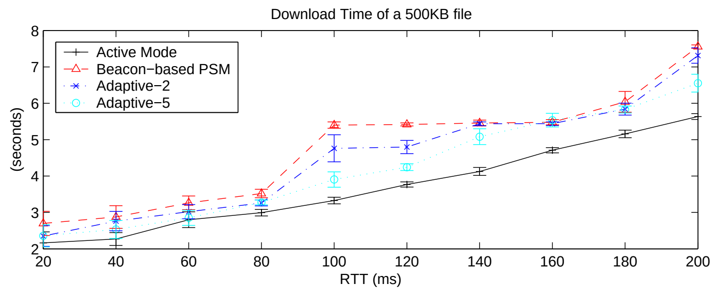
In (39) an interesting interaction between TCP and the Beacon-based[1] power saving protocols used in Wi-Fi was introduced. This interaction is as follows. The first TCP packets sent by the TCP connection in slow start are buffered in the AP and only retrieved by the station after a Beacon frame. While retrieving these frames, the station generates TCP ACKs in the Uplink and immediately goes back to the sleep state. After an RTT the next window of TCP packets sent by the server will reach the AP and be buffered there until the next Beacon frame. This process will repeat until the time required by the station to retrieve the whole window of packets stored in the AP after a Beacon gets above the RTT between the AP and the server; from that moment on the station will remain awake until the TCP connection closes. Notice though that the station may never enter this 'active-mode' like behavior if its TCP advertised window is smaller than the required critical value.

This effect can be altered in a beneficial way when our adaptive algorithm is used. Figure 3.8(a) depicts the instantaneous traces of a TCP download between a Wi-Fi station and a server experiencing a 20ms RTT with the AP in the case of the adaptive algorithm, a traditional Beacon-based power save mode and a station in active mode. The previous interaction is clearly observed in the case of the Beacon-based power save mode. However, in the case of the adaptive algorithm, the length of the initial RTTs can be highly reduced because the additional triggers generated by the algorithm after a TCP burst can capture the new burst of TCP packets arriving from the server before the next Beacon frame. Hence, the actual improvements obtained in the case of the adaptive algorithm will depend on the value of the RTT experienced by the connection, the number of additional triggers generated by the algorithm before switching off and the value of β in Algorithm 3.1. We do not study in this work how to optimally tune these two parameters. A detailed study on the interactions between TCP and Wi-Fi power saving protocols is presented in Chapter 6.

[1]We consider a Beacon-based power saving protocol as a protocol where the station only triggers the AP for data after receiving a Beacon frame with the TIM bit set to 1.



(a) TCP download comparison.



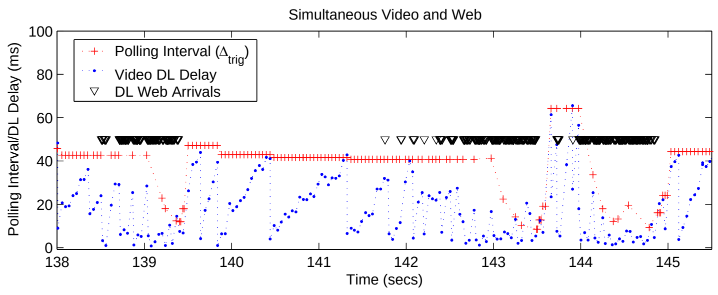
(b) Effect of the RTT on the TCP download time.

Figure 3.8: Dynamics of the adaptive algorithm with TCP traffic.

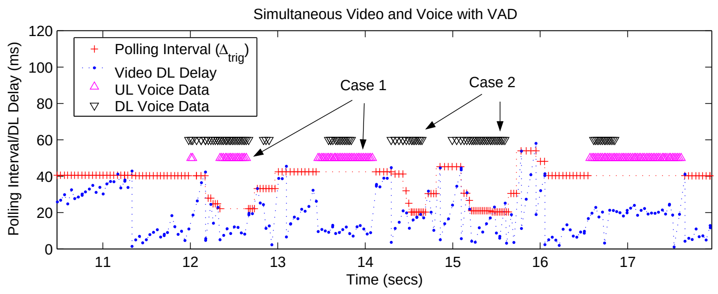
To better illustrate the effects of the adaptive algorithm on the lifetime of TCP connections, in Figure 3.8(b) the download time of a 500KB file in our reference scenario is depicted while increasing the value of the RTT between the AP and the TCP server. As in (39) the TCP advertised window in the Wi-Fi station is configured to be 240 Kbits. Four different cases are considered, a station in active mode, a Beacon-based power save algorithm, the adaptive algorithm generating two triggers before switching off (*Adaptive-2*) and the adaptive algorithm generating 5 triggers before switching off (*Adaptive-5*). The results corroborate that the adaptive algorithm can reduce the download time with respect to the traditional Beacon-based algorithm and that this reduction increases with the number of triggers generated before switching off.

3.4.1.4 Simultaneous Applications

Until now we have always assumed that our Wi-Fi station under study was running only one application at a time. This is often not the case in realistic scenarios where a terminal generates traffic from more than one application at once. In this section we study how the adaptive algorithm reacts to traffic generated by multiple applications at the same time.



(a) Simultaneous Video and Web.



(b) Simultaneous Video and Voice with VAD.

Figure 3.9: Dynamics of the adaptive algorithm with multiple simultaneous applications.

Figure 3.9(a) illustrates the behavior of the algorithm in a terminal running a Video Streaming application, through AC_VI, while performing Web browsing, through AC_BE. The Figure depicts the polling interval estimation kept by the algorithm, the delay experienced by the Video frames and the presence of Web data packets buffered in the AP (black markers). The dynamics of the algorithm are as follows. When no Web traffic is present in the AP the algorithm converges to the interarrival time of the Video codec, i.e. 40ms. However, when a burst of Web traffic arrives at the AP, the algorithm suddenly sees an increased amount of data and reduces the estimation in order to quickly retrieve the extra data. When the burst of Web data packets is retrieved, the polling intervals returns to 40ms.

Figure 3.9(b) illustrates the case where a terminal simultaneously generates traffic from a Video Streaming application, through AC_VI, and a Voice codec with silence suppression, through AC_VO. The behavior of the adaptive algorithm can be understood in the following terms. When the Voice codec is in silence in both uplink and downlink directions, as observed in the trace before 12 seconds, the polling interval converges to the video codec interarrival time, i.e. 40ms. When Voice packets are generated in the uplink, as observed in the trace in the areas labeled as *Case 1*, the rescheduling mechanism avoids the generation of signaling triggers, and the uplink Voice packets download both

Voice and Video packets buffered in the AP.[1] Finally, when downlink Voice packets are buffered in the AP but no Voice data packets are generated in uplink, as observed in the trace in the areas labeled as *Case 2*, the polling interval estimation converges to the most stringent delay requirement, in this case Voice with a 20ms interarrival time.

As shown in the previous experiments the algorithm performs as desired when simultaneous applications are considered. Signaling load is generated according to the application showing the most stringent delay requirements, and the algorithm is fast enough to quickly adapt to bursty applications, e.g. Web and Voice with silence suppression.

3.4.2 Performance under congestion

In this section we study how the adaptive algorithm behaves when the congestion in the Wi-Fi network increases. For this purpose we consider a set of N clusters in the Wi-Fi, one cluster being composed of four stations and each station running a different application from our defined traffic mix, and we increase N until congestion appears.

In addition we compare the adaptive algorithm with two other algorithms existent in the state of the art. Unlike our adaptive proposal though, these algorithms require knowledge about the application characteristics. We will assume that these algorithms are perfectly configured according to the application characteristics in order to set an upper bound on the expected performance of the adaptive algorithm. Next, we describe the mentioned algorithms that will be hereafter referred to as the *Ideal Static* and the *Ideal Static+Signaling Reduction(SigRed)* algorithms.

The *Ideal Static* algorithm configures the operation of U-APSD depending on the traffic generated by the stations. When the Wi-Fi terminal runs a Voice or Video application, U-APSD is statically configured to periodically generate trigger frames with an interval adapted to the nominal application interarrival time[2]. However if the terminals use applications running over AC_BE or AC_BK, like Web or FTP in our experiment, the Ideal Static algorithm configures U-APSD to behave like a Beacon-based power save mode.

The *Ideal Static+SigRed* algorithm is an improvement over the previous algorithm inspired on the BSD protocol proposed in (39) that reduces the amount of introduced signaling load when the application is in a silence period. The idea is the following. When the terminal runs a Voice or Video application a nominal polling interval is defined that matches the nominal application characteristics. However, if the terminal sends a signaling frame and observes that no data was buffered in the AP

[1]Notice that we configure all ACs as trigger and delivery enabled in U-APSD.

[2]If Uplink data triggers are sent signaling triggers are recheduled.

the algorithm infers that the application is in a silence period and increases the polling interval like $T_{int}(n+1) = T_{int}(n)(1+p)$, where $p > 0$. Whenever traffic is again detected in the downlink direction the trigger interval is returned to its nominal value. The parameter p can be used to trade-off the amount of signaling load introduced during silence periods and the extra delay experienced by the downlink stream if wrong silence periods are inferred. For our evaluation we consider $p = 0.5$ and in order to keep a bounded delay $T_{int_{MAX}} = 100ms$. Regarding applications running over AC_BE or AC_BK we consider that the Ideal Static+SigRed algorithm behaves in the same way than the Ideal Static one.

Next, we present the performance of the *Adaptive*, *Ideal Static* and *Ideal Static+SigRed* algorithms in terms of QoS, power consumption and introduced signaling load. In addition we finalize this subsection providing a deeper insight on how congestion in the Wi-Fi network affects the dynamics of the adaptive algorithm.

The simulations presented in this section have a length of 300 seconds with a warm-up phase of 30 seconds. The graphs showing average values are plotted with the corresponding 95% confidence intervals[1]. The graphs regarding delay show the 99% percentile of the delay computed considering together the samples obtained from all simulation runs.

3.4.2.1 QoS Performance

In this section we analyze the Throughput and Delay metrics. Since the distributed power saving mechanisms degrade the performance mainly in the downlink direction (AP $\rightarrow$ STA), we focus on the downlink results which is the performance bottleneck of the system.

Figure 3.10(a) shows for each AC the average MAC throughput experienced in the downlink direction. The results show that the Adaptive (solid line) and the Ideal Static+SigRed (dash-dot line) algorithms outperform the Ideal Static algorithm (dotted line) in all ACs except AC_VO, which still does not reach saturation due to the small size of the VoIP packets compared to the AC_VO TXOP length and the amount of buffering available in the AP. The main reason for the improved performance observed in the case of the Adaptive and Ideal Static+SigRed algorithms is their ability to reduce signaling load when the application is in a silence period. Notice that in the case of Voice, detecting silence periods in the downlink results in a reduction of the amount of high priority load (AC_VO) introduced in the channel, which benefits all other ACs. In addition, a slightly better performance under congestion is observed in the case of the Adaptive algorithm with respect to the Ideal Static+SigRed algorithm. The reason is the

[1]Note that the confidence intervals are present in all graphs although they might not be visible in some cases due to its small size

particular configuration of both algorithms which results in the Adaptive algorithm introducing slightly less signaling load during Voice silence periods.

In Figure 3.10(b) the worst case (99% percentile) delay is depicted for the different algorithms under study and each AC. The delay results closely resemble the throughput ones, with the Adaptive and Ideal Static+SigRed algorithms clearly outperforming the Ideal Static one, the only difference being that now differences are observed also for AC_VO. When the network is not yet fully congested (<15 stations), the Adaptive algorithm bounds the downlink delay to the interarrival time in the case of AC_VI but not in the case of AC_VO, where a higher value is observed. This is the price to pay in order to save signaling load because when the trigger generation stops the next voice samples can only be recovered with the Beacon frame, experiencing hence a higher delay. The same effect is observed in the case of the Ideal Static+SigRed algorithm for AC_VO, due to the silence periods of the codec, and AC_VI, because of spureous increments in the trigger interval due to jitter in the Video stream. Notice that considering a higher value for p in the Ideal Static+SigRed algorithm would result in higher worst case delays when the network is not congested.

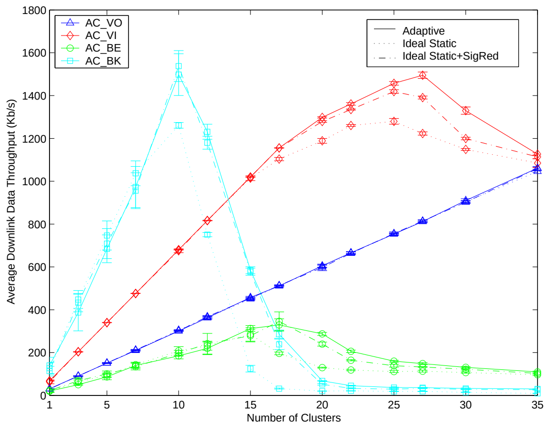
Finally, we conclude that in terms of QoS our proposed Adaptive algorithm can provide a performance equivalent to that of optimally configured dynamic approaches existent in the state of the art (Ideal Static+SigRed) and a much superior performance than static approaches (Ideal Static), with the advantage of not requiring a priori knowledge about the applications characteristics.

3.4.2.2 Power Saving Efficiency

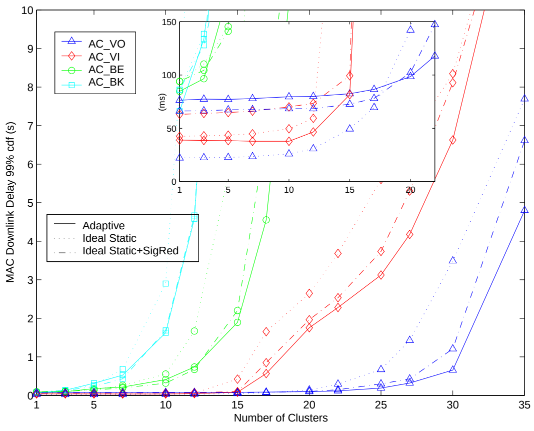
The main objective of any power saving mechanism is to reduce the power consumption of a station. In this section, we analyze the power consumed by the stations in order to study whether the benefits exhibited by the adaptive algorithm have a price in terms of power consumption.

The power consumption model used for the evaluation has been derived based on current Wi-Fi cards/chipsets available in the market and consists of four states: Sleep, Listen, Reception and Transmission. In order to obtain the power consumed by the stations, we first compute the percentage of time spent during an active session in each state by the stations per AC. Then, we translate this generic metric into *mA* weighting the current consumed by a station in each state. This information has been obtained from the product datasheet of a common PCMCIA Wi-Fi card (66). Thus, the metric represents the average instantaneous current consumption (in mA) experienced by the stations in each AC. The current consumption values used are shown in Table 3.3[1]

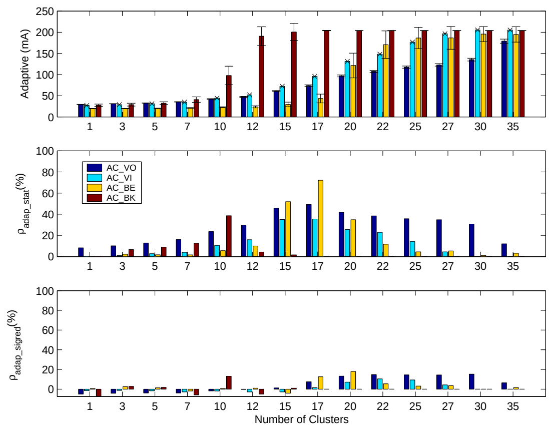
[1]For the sleep mode we used the value of a previous model of a Cisco PCMCIA card (Cisco Aironet 350) since no information was available for the current one



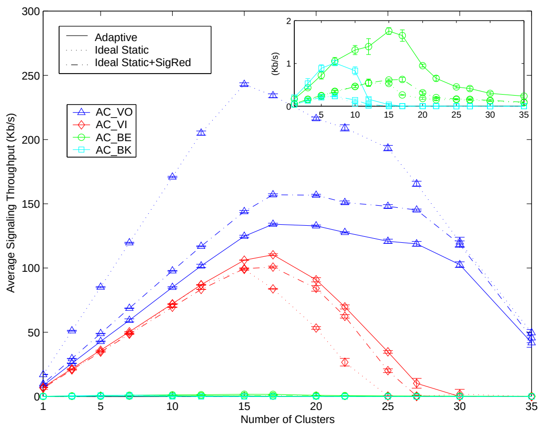
(a) MAC DL throughput when increasing the number of stations.



(b) MAC DL delay when increasing the number of stations.



(c) Current consumption when increasing the number of stations.



(d) Signaling overhead when increasing the number of stations.

Figure 3.10: Overall Network Performance.

Cisco Aironet^{TM}	Sleep	Listen	Rx	Tx
Current (mA)	15	203	327	539

Table 3.3: Current consumption levels of a popular PCMCIA card.

The upper part of Figure 3.10(c) illustrates the average current consumption levels for each AC for the adaptive algorithm. It can be observed how the average current consumption increases with the congestion in the network. These results though, confirm that when the network is not congested, a noticeable current consumption reduction can be achieved during an active session if a distributed power saving mechanism is used. Looking for instance at the case of VoIP, less than 50mA are consumed on average when the network is not congested, in front of almost 200mA when the network is highly congested and the stations spend almost all the time awake.

Regarding the current consumption differences between the different algorithms, the middle and lower part of Figure 3.10(c) illustrate the relative current consumption reduction of the adaptive algorithm with respect to the other ones ($\rho_{Adaptive\_AlgX} = \frac{AlgX(mA)-Adaptive(mA)}{AlgX(mA)}$). We can see that with respect to the Ideal Static algorithm (middle figure) a significant power consumption reduction is achieved by the adaptive algorithm in all ACs. With respect to the Ideal Static+SigRed algorithm (lower figure) though, a slight power consumption reduction is observed only when the network is congested. The main reason for the reduced current consumption is again on the smaller amount of signaling load introduced by the adaptive algorithm, which results in less congestion in the network.

3.4.2.3 Signaling Overhead

The main factor determinining the scalability of the different algorithms under study is the amount of signaling load introduced. The challenge is to minimize the amount of signaling load while at the same time provide the required QoS to the applications. In this section we validate the previous statements related to the amount of signaling introduced by each algorithm by showing in Figure 3.10(d) the total signaling both in uplink and downlink introduced by each AC.

Regardless of the algorithm considered, the first notable effect is a reduction in the amount of signaling introduced in all ACs when the congestion in the channel increases. This effect is intrinsic to the U-APSD protocol and is due to the following two facts: i) the AP delivers all buffered data for a station when receiving a trigger frame, and ii) the station does not generate new trigger frames if there is an ongoing service period. Thus, when the network gets congested, the time that the AP needs to deliver a frame in a service period becomes longer, which in turn increases the chances that a new frame addressed to the station arrives at the AP within an ongoing service period. This translates into an increase of the stations' listen time and in a reduction of the number of triggers generated.

Focusing on AC_VO, we can see the effective reduction in signaling load achieved by the Adaptive and Ideal Static+SigRed algorithms. Indeed, stopping the trigger generation when there is no activity in the downlink, saves both the uplink signaling generated by the station and the downlink signaling generated by the AP in order to end the service period. In addition, a slightly higher amount of signaling is introduced by the Ideal Static+SigRed algorithm which could be reduced, at the price of accepting higher delays, by increasing the parameter p.

Regarding the AC_VI application, when the network is not saturated a very similar amount of signaling is introduced by all the algorithms, which saturates before in the case of the Ideal Static algorithm due the higher level of congestion in the network in this case.

Finally, looking at the AC_BE and AC_BK results, we can see that a very small amount of signaling is introduced as compared to the other ACs. The reason is that TCP traffic arrives in bursts that can be retrieved by the stations with a single trigger thanks to the configured *Max_SP_length* value. The subgraph in the graph shows that a slightly higher amount of signaling is introduced in the case of the adaptive algorithm. The reason is that once the burst of packets is downloaded, the adaptive algorithm generates some additional triggers prior to switching off which as previously seen in subsection 3.4.1 can help to improve the performance of TCP.

3.4.2.4 Understanding the effect of congestion

We finish this section providing an analysis of how congestion in the Wi-Fi network affects the dynamics of the adaptive algorithm.

Figure 3.11 depicts the CDF of the trigger interval estimation for the stations running Video considering three different levels of congestion, i.e. 5, 15 and 25 clusters (i.e. 5x4,15x4 and 25x4 competing stations). As congestion increases, e.g. 15 clusters, the estimation becomes more unstable but is still kept around the desired nominal interarrival time (40ms in the case of Video). The reason is that congestion in the Wi-Fi network translates into packet drops or extra delays which distort the interarrival time of the downlink stream perceived by the algorithm.

These results show that the estimation kept by the algorithm becomes less accurate as congestion in the Wi-Fi network increases but it is still kept in 80% of the cases within a $\pm 10\%$ range even for a significantly high congestion level as 25x4 competing stations. Note from Figure 3.10(a) that Video traffic has already started to experience congestion at 25x4 competing stations. Hence we conclude that the proposed algorithm is able to cope with moderate to high levels of congestion in the Wi-Fi network.

3.5 Conclusions

Designers of mobile computing devices incorporating the Wi-Fi technology face new challenges with respect to the QoS and power saving requirements to be met in order to satisfy users' expectations. IEEE

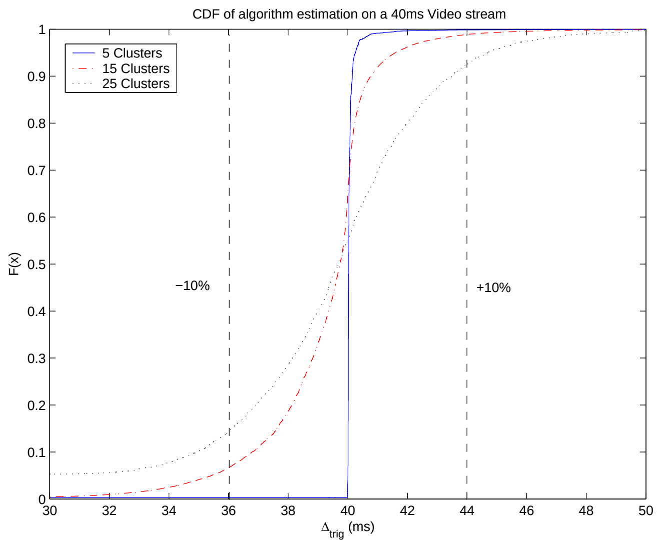


Figure 3.11: Effect of congestion on the Adaptive Algorithm.

802.11 and 802.11e have defined protocols to address these challenges but the actual algorithms required to achieve the desired performance are left undefined to allow for vendor differentiation. The focus of our work in this chapter has been the study of the dependencies between the configurable parameters of the distributed power saving mechanisms and its impact over the resulting QoS performance.

Our contributions in this chapter have been the following. First, we have derived an analytical model of the dependency of the delay and jitter of real-time traffic with the trigger interval used by the power saving protocol. Second, we have designed an adaptive algorithm, that based on the steepest descent method, determines the appropriate trigger interval for requesting frames to the AP according to the instantaneous load offered by the application. Our proposed algorithm uses only information available at the MAC layer. Third, we have analyzed the convergence properties of our adaptive algorithm, providing guidelines to set its configurable parameters and a worst case bound on the number of updates required to converge. Finally, we have compared, by means of simulations, the performance of our approach with approaches existent in the state of the art which require knowledge of the applications' characteristics.

Appendix 3.A Convergence conditions with uplink triggers

Assuming the definitions introduced in Section 3.3, K_i frames arrive at the AP during uplink subperiod i, where $K_i = \lfloor \frac{\Delta_{UL}+\phi_i}{\Delta_{DL}} \rfloor$. Hence:

$$K_i = \lfloor \frac{\Delta_{UL}}{\Delta_{DL}} \rfloor + \lfloor \frac{\Delta_{UL} \bmod \Delta_{DL} + \phi_i}{\Delta_{DL}} \rfloor$$

Thus:

$$K_i = \begin{cases} \lfloor \frac{\Delta_{UL}}{\Delta_{DL}} \rfloor & \text{if } \phi_i < \Delta_{DL} - \Delta_{UL} \bmod \Delta_{DL} \\ \lceil \frac{\Delta_{UL}}{\Delta_{DL}} \rceil & \text{if } \phi_i \geq \Delta_{DL} - \Delta_{UL} \bmod \Delta_{DL} \end{cases}$$

Assuming the general case where Δ_{UL} is not multiple of Δ_{DL} ($\Delta_{UL} \neq n\Delta_{DL},\ n \in \mathbb{Z}$), and recalling $0 \leq \phi_i < \Delta_{DL}$, the channel situation periodic $T = lcm(\Delta_{UL}, \Delta_{DL}) = M\Delta_{DL} = N\Delta_{UL}$, and T composed by N consecutive uplink subperiods, during $l > 0$ and $j > 0$ uplink subperiods $\lfloor \frac{\Delta_{UL}}{\Delta_{DL}} \rfloor$ and $\lceil \frac{\Delta_{UL}}{\Delta_{DL}} \rceil$ downlink frames are received at the AP respectively, with:

$$\begin{cases} l\lfloor \frac{\Delta_{UL}}{\Delta_{DL}} \rfloor + j\lceil \frac{\Delta_{UL}}{\Delta_{DL}} \rceil = M \\ l + j = N \end{cases}$$

Therefore:

$$\begin{aligned} l &= N(\lfloor \tfrac{\Delta_{UL}}{\Delta_{DL}} \rfloor + 1) - M \\ j &= M - N\lfloor \tfrac{\Delta_{UL}}{\Delta_{DL}} \rfloor \end{aligned}$$

Thus, within T there are both uplink subperiods receiving $\lfloor \frac{\Delta_{UL}}{\Delta_{DL}} \rfloor$ and $\lceil \frac{\Delta_{UL}}{\Delta_{DL}} \rceil$ data frames in the downlink. Therefore the station must generate $\delta = \lfloor \frac{\Delta_{UL}}{\Delta_{DL}} \rfloor$ signaling triggers between two consecutive data triggers in order not to generate any event in any uplink subperiod. Thus, the reuse of uplink data triggers in this case is $\frac{j}{N}$.

In the case that $\Delta_{UL} = n\Delta_{DL},\ n \in \mathbb{Z}$, the same number of data frames arrive at the AP in all uplink subperiods, $\frac{\Delta_{UL}}{\Delta_{DL}}$, and a converged value of Δ_{trig} can be obtained generating $\delta = \frac{\Delta_{UL}}{\Delta_{DL}}$ or $\delta = \frac{\Delta_{UL}}{\Delta_{DL}} - 1$ signaling triggers between uplink data triggers. Thus, the reuse of uplink data trigers in this case will be 0 or 1 respectively.

4

Leveraging 802.11n Frame Aggregation to enhance QoS and Power Consumption in Wi-Fi networks

As described in Chapter 2 the 802.11n technology has defined very efficient frame aggregation schemes that reduce the overhead existent in legacy Wi-Fi networks. It is easy to realize that such frame aggregation techniques can also be utilised in order to increase energy efficiency. For instance, within the context of a Voice call, a Wi-Fi device may decide to aggregate Voice packets in pairs hence reducing in half the number of required channel access attempts and doubling its sleep intervals. There is clear price to be paid when using aggregation though, and is an increase in delay which might degrade the QoS of real-time applications. Therefore, our goal in this chapter is to study how to properly leverage the aggregation capabilities of 802.11n in order to improve QoS and energy efficiency in Wi-Fi.

Current 802.11n products in the market use frame aggregation mostly for non time sensitive applications, but it is not clear if frame aggregation can be benefitial with real-time applications that have tight delay requirements. A common approach used both in the state of the art and in the industry is what is known as Zero Delay Frame Aggregation (ZFA) (87), which consists of aggregating all frames present in the transmission buffer when a channel access is obtained, without introducing any extra delay. ZFA though mostly applies to data traffic which, due to TCP congestion control, naturally fills up transmission buffers. In this chapter we will study the question of how to use frame aggregation in order to improve the performance of real-time applications.

Notice that the adaptive algorithm presented in Chapter 3, already provided some hooks that could be used to tune the amount of aggregation used by a Wi-Fi device running a real-time application. For instance as described in the previous chapter, our adaptive algorithm can be easily modified to converge to a multiple k of the application's interarrival time. Thus, by setting $k = 2$ one could implement the policy described in the previous paragraph where a Wi-Fi device transmits and receives aggregates of two frames during a Voice call. In particular, the work presented in this chapter, studies how to optimally select the aggregation interval used in a Wi-Fi station according to the state of the network. The contents of this chapter have been submitted to (27) and its main contributions are as follows:

- We analyze the effect of 802.11n MAC aggregation on the performance of the distributed Wi-Fi power saving protocols, i.e. 802.11 PSM and U-APSD. Our study reveals that U-APSD, currently used when battery operated devices require real-time applications like Voice, is poorly suited to benefit from the 802.11n aggregation mechanisms. Instead, 802.11 PSM significantly benefits from aggregation and can outperform U-APSD when congestion in the network increases.

- We propose a novel algorithm *Congestion Aware - Delayed Frame Aggregation (CA-DFA)* that, using only information available at layer two, adapts the amount of aggregation used by a Wi-Fi device according to the level of congestion in the network. CA-DFA significantly outperforms alternative solutions in the state of the art in terms of QoS, energy saving and network capacity, and can be easily implemented in current 802.11n devices.

This chapter is organized as follows. Section 4.1 describes the results of a simulative study that evaluates the combined performance of the 802.11n aggregation mechanisms and the current Wi-Fi QoS and power saving protocols. Section 4.2 presents the design and evaluation of our CA-DFA algorithm, and Section 4.3 evaluates its performance. Finally, Section 4.4 summarizes and concludes this chapter.

4.1 Effect of 802.11n frame aggregation on Wi-Fi QoS and power saving protocols

In this section we present the results of a simulative study that evaluates the effect of the 802.11n MAC aggregation mechanisms on the current Wi-Fi QoS and power saving protocols. Our focus is in protocols currently deployed in the market and thus, we consider in our evaluation EDCA for QoS, 802.11 PSM and U-APSD for power saving, and 802.11n A-MPDU as MAC aggregation technique, please refer to Chapter 2 for a description of these protocols. We start this section by describing our simulation framework in Section 4.1.1, and then discuss the results obtained in our study in Section 4.1.2.

4.1.1 Simulation framework

Our evaluation is based on packet level simulations using OPNET (64). In particular, we extended an 802.11n OPNET model contributed by Intel (88), to include the 802.11 PSM and U-APSD power saving protocols. This model was used for internal evaluation of competing proposals in the IEEE 802.11 TGn group.

We consider the *HotSpot* scenario defined by the TGn group in (89), where stations are stationary and randomly placed within a 30 meter radius from the AP. In addition, we consider that stations implement 2x2 MIMO, which is becoming increasingly available even for mobile devices (85). In our simulations though, stations do not transmit using a fixed data rate but instead adapt their data rate according to the varying radio conditions. To model the radio channel we use the *TGn Channel Model E* in the 5Ghz band which was also defined by the TGn group, and has been proven to faithfully model realistic channel conditions (90).

In order to evaluate the combined performance of the Wi-Fi QoS and power saving protocols and the frame aggregation mechanisms defined in 802.11n, we define a basic *cluster* of stations, and increase the number of clusters present in our HotSpot scenario until the network starts to saturate. Our basic cluster is comprised of four types of stations which generate a traffic mix representative of a typical HotSpot. The applications used by each type of stations together with their Wi-Fi Access Category (AC) are depicted in Table 4.1.

	Description	**AC**
Voice	Bidirectional Voice calls with G.711 (64Kbps) at 20ms. Talk spurt and silence spurt exponential with mean 0.35 seconds and 0.65 seconds	AC_VO
Video	Downlink VBR stream at 25fps with an average rate of 1Mbps and a peak rate of 5Mbps obtained from (65).	AC_VI
Web	Inter-page request time exponentially distributed of mean 15 seconds. RTT of 20ms between the AP and the Web server (typical RTT in a domestic internet path (86)). Size and number of objects in a Web page are modelled according to (91).	AC_BE
FTP	FTP download of a 20MB file. RTT of 20ms between the AP and the file server. TCP New Reno used as transport protocol.	AC_BK

Table 4.1: Applications Description.

We repeat our experiments with four different configurations with the aim of illustrating the effect of each individual protocol on the achieved performance. These configurations are:

- *802.11 PSM*, where all stations operate using 802.11 PSM and no aggregation is used.
- *802.11 PSM+ZFA*, where all the stations operate using 802.11 PSM and perform A-MPDU aggregation using a Zero Delay Frame Aggregation (ZFA) approach.
- *U-APSD*, where all the stations use U-APSD and no aggregation is used.
- *U-APSD+ZFA*, where all the stations use U-APSD and perform MAC aggregation using the ZFA scheme.

Thus, our goal is to understand the combined effect of these protocols in terms of performance of the different applications, power consumption of the Wi-Fi devices and total capacity of the network.

We configure the AP to generate a Beacon frame every 100ms. In addition, in 802.11 PSM stations generate PS-Poll frames after the Beacon using AC_BE, as recommended in the 802.11 standard (33). Regarding U-APSD, Voice and Video stations proactively trigger the AP every 20ms or 40ms respectively (an uplink data frame or a QoS Null frame can be used as triggers). Instead, Web and FTP U-APSD stations generate a trigger frame upon receiving a Beacon indicating that there is data buffered for them. The interested reader is referred to chapter 3 or to our early work in (47) for more details on this U-APSD implementation.

The QoS settings used in our evaluation that define the relative prioritization among applications and the aggregation capabilities of each device, are described in Table 3.2[1]. These settings are obtained based on the recommendations defined in the 802.11 standard (33).

	AIFS	**CWmin**	**CWmax**	**TXOP**	**Max.Agg.**
AC_VO	2	7	15	1.5 ms	64
AC_VI	2	15	31	3 ms	64
AC_BE	3	31	1023	0 ms	64
AC_BK	7	31	1023	0 ms	64

Table 4.2: EDCA configuration for the different ACs.

Regarding power consumption, we use a well known model based on a popular Wi-Fi card/chipset (24), which consists of four basic states: Sleep, Listen, Reception and Transmission. We obtain the

[1]Notice in Table 4.2 that a station can aggregate up to *Max.Agg.* frames as long as the resulting A-MPDU fits within the configured TXOP limit. In addition, AC_BE and AC_BK are allowed to transmit only one aggregated frame when accessing the channel, which can span up to 3 ms.

power consumed by the stations' Wi-Fi interface by computing the amount of time spent during an active session in each state and then applying the power consumed in each state per unit of time. The power consumption values used are shown in Table 4.3.

Broadcom 4311^{TM}	**Sleep**	**Listen**	**Rx**	**Tx**
Power (mW)	20	390	1500	2000

Table 4.3: Power consumption levels used in our study.

Finally, each point in the following graphs has been obtained considering at least 5 simulation runs of 150 seconds with a transient period of 30 seconds. Average values are plotted with the correspondent confidence intervals at 95%. Cumulative distribution functions are obtained considering together the values obtained across all simulation runs.

4.1.2 Performance evaluation

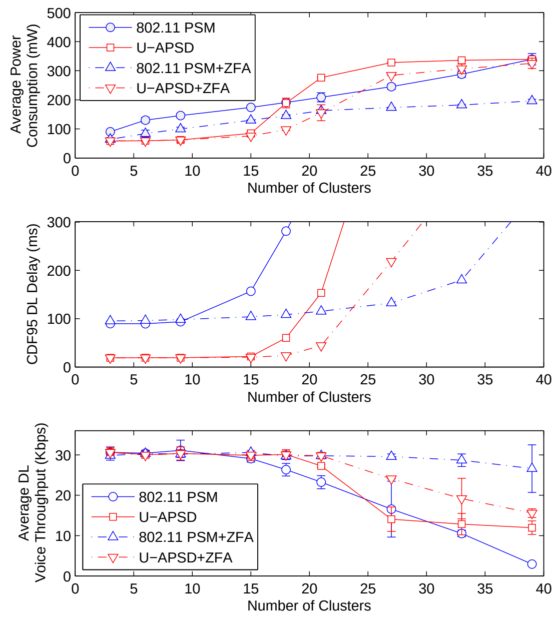
In this section we describe the results obtained in our study. For readibility reasons we divide the analysis in two parts depending on the QoS requirements of the applications.

4.1.2.1 QoS sensitive applications

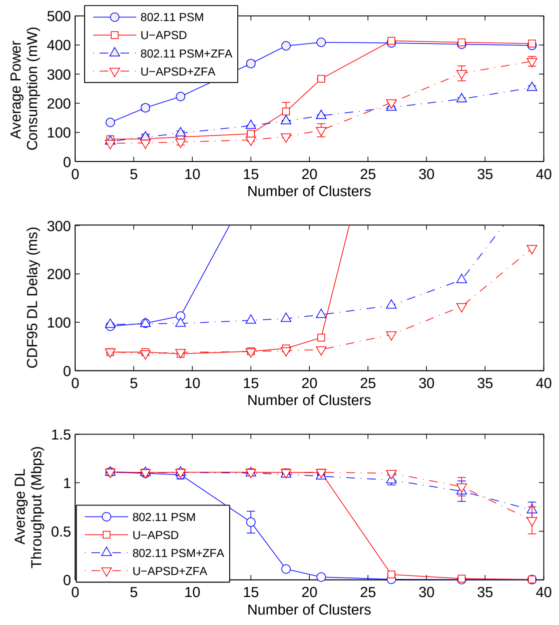
Figures 4.1(a) and 4.1(b) depict the performance of Voice and Video applications when the number of station clusters in the network increases. The figures are divided in three subfigures: i) the upper subfigure depicting average power consumption, ii) the middle subfigure depicting the worst case (cdf95) downlink delay[1], and iii) the lower subfigure depicting the average throughput obtained by a Voice or Video station. In addition, each subfigure depicts the performance obtained for all configurations under study: i) *802.11 PSM*, ii) *802.11 PSM+ZFA*, iii) *U-APSD* and iv) *U-APSD+ZFA*.

We start discussing the performance of the *802.11 PSM* and *U-APSD* configurations which is similar for the Voice and Video applications. As expected, *U-APSD* outperforms *802.11 PSM* in terms of QoS, energy efficiency and network capacity. The main reasons for the better performance of *U-APSD* (as already pointed out in (47)) are: i) shorter polling interval which allows reducing delay for QoS sensitive applications, ii) smaller signaling overhead as compared to *802.11 PSM*, where stations generate a PS-Poll for each packet buffered in the AP, and iii) smaller collision probability with respect to *802.11 PSM*, where all stations try to access the medium immediately after the Beacon frame.

[1]Notice that when a power saving protocol is used the downlink link suffers a higher delay than the uplink one.



(a) Voice Performance.



(b) Video Performance.

Figure 4.1: QoS sensitive Applications. Performance of Voice and Video stations when increasing the number of station clusters in the network, in terms of Power Consumption (upper graph), CDF95 Downlink Delay (middle graph) and Average Downlink Throughput after the de-jitter buffer (lower graph).

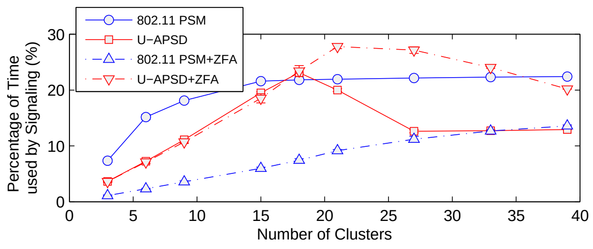
Let us now discuss the *802.11 PSM+ZFA* and *U-APSD+ZFA* configurations where we add 802.11n MAC aggregation capabilities to the previous power saving protocols. As expected, aggregation increases network capacity both for 802.11 PSM and U-APSD. However, the network capacity gain is significantly higher in the case of 802.11 PSM, as observed specially in the case of Voice. The main reason for this is that 802.11 PSM intrinsically creates *aggregation opportunities* by buffering packets in the AP and only transmitting them every 100ms (with the Beacon frame), whereas stations using U-APSD (in an attempt to reduce delay) trigger the AP at shorter intervals thus creating less aggregation opportunities. The novel result in Figure 4.1 is that when congestion increases, the aggressive behavior of U-APSD turns out to be counter-productive even in terms of QoS.

Interestingly, although 802.11 PSM creates more aggregation opportunities, theoretically it should also introduce a higher signaling overhead than U-APSD (transmission of a PS-Poll per buffered packet) and experience a higher collision probability (synchronization of stations transmission after the Beacon). Therefore, a deeper analysis is needed to understand why the increased number of aggregation opportunities in 802.11 PSM overcomes the previous disadvantages and allows *802.11 PSM* to outperform *U-APSD*. For that purpose, the signaling and channel access overheads introduced in the channel by the different protocols under study is analyzed, see Figures 4.2(a) and 4.2(b).

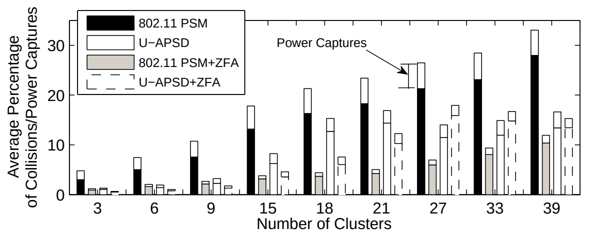
Considering first Figure 4.2(a) it can be observed that while signaling[1] drastically reduces for 802.11 PSM when aggregation is used, it does not decrease for U-APSD. The reason why signaling reduces for *802.11 PSM+ZFA* is that the AP is often able to transmit all the buffered data for a station aggregated in a single A-MPDU upon reception of a PS-Poll. Hence, stations send only one PS-Poll per Beacon interval. However, aggregation does not reduce signaling in the case of U-APSD+ZFA, where, given the more frequent polling intervals, a trigger frame typically retrieves a single frame from the AP. Indeed, aggregation even increases the amount of signaling in U-APSD+ZFA when the network is congested compared to the U-APSD case. This is an indirect consequence stemming from the fact that U-APSD stations reduce the signaling they introduce when congestion increases. Thus, as congestion starts earlier in the case of U-APSD than in the case of U-APSD+ZFA, the signaling overhead gets reduced earlier. Notice that this effect was already observed when evaluating the different algorithms presented in Chapter 3.

Finally, Figure 4.2(b) depicts the channel access overhead as the percentage of collisions in the channel for each of the different protocols under study (for the sake of clarity confidence intervals are not included in the graph). It is interesting to see that while the percentage of collisions is clearly

[1]We consider as signaling, PS-Poll frames in 802.11 PSM, QoS Null frames in U-APSD, and RTS-CTS handshakes in both protocols.



(a) Signaling Overhead. Percentage of channel time spent on signaling defined as PS-Polls, QoS-Nulls and RTS-CTS handshakes.



(b) Channel Access Overhead. The total height of a bar depicts the percentage of collisions. The white top portion of each bar represents the amount of collisions that resulted in a power capture.

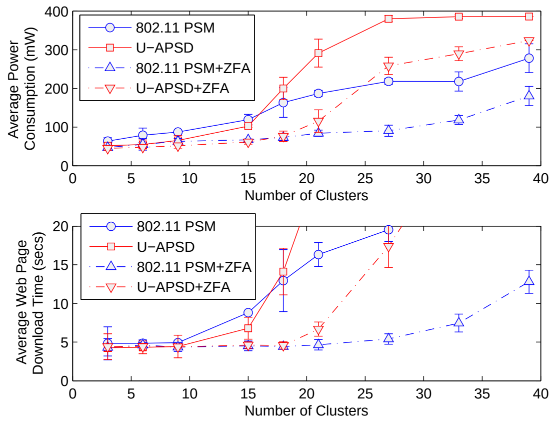
Figure 4.2: Overhead incurred by the power saving protocols under study. The upper graph depicts signaling overhead, and the lower one channel access overhead (collisions).

higher in *802.11 PSM*, it drastically reduces when aggregation is used because the number of frames contending for the channel after each Beacon is much lower. In addition, the white top portion of each bar in Figure 4.2(b) depicts the percentage of collisions that resulted in a *power capture* event in our experiment. A power capture event is defined as a collision event where a receiver successfully decodes one of the colliding packets. This effect is very common in Wi-Fi networks and has been thoroughly studied in (92). Notice that while power captures affect the fairness of the CSMA/CA protocol used in Wi-Fi, they can in some cases increase overall network capacity, because upon such an event one of the colliding stations does not need to retransmit and continues to use reduced contention parameters. Thus, the significant number of power capture events observed in our simulations partially mitigates the increased collision probability introduced in 802.11 PSM, contributing to the explanation of its enhanced performance.

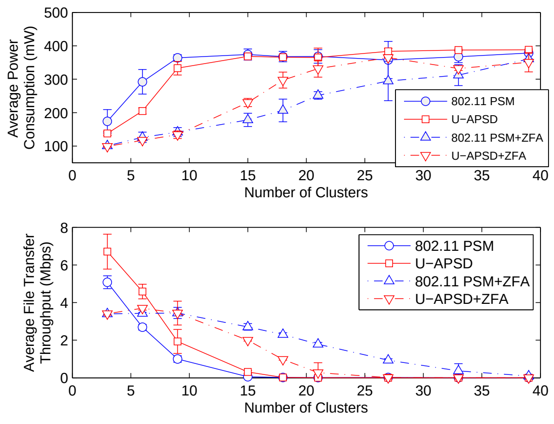
4.1.2.2 Non-QoS sensitive applications

We discuss in this section the performance of the non-QoS sensitive applications, i.e. Web browsing and FTP downloads. Figures 4.3(a) and 4.3(b) depict the performance of the Web and FTP stations, respectively, in terms of power consumption (upper graph) and average Web page download time or average file transfer throughput (lower graph).

The effect of the protocols under study on the non-QoS sensitive applications is similar to the one introduced on the QoS sensitive applications. The reason is that an improvement in the channel usage efficiency results in an enhacement in the performance not only of QoS sensitive applications but also of non-QoS sensitive applications since more bandwidth is available for data transmissions. Thus, when no aggregation is used and the network is not heavily congested, the *U-APSD* configuration slightly outperforms the *802.11 PSM* configuration. However, like in the case of QoS sensitive applications, this trend is reversed when aggregation is used, where the higher aggregation efficiency in *802.11 PSM* leaves more bandwidth available for non-QoS sensitive applications. In addition, another factor contributing to the performance of 802.11 PSM, is that in this case all stations (including QoS sensitive ones) send PS-Polls after the Beacon frame using the AC_BE settings. However, in the U-APSD case, stations request their buffered data transmitting QoS-Nulls according to the priority of the application they are intended for. Hence, penalizing lower priority applications.



(a) Web Performance.



(b) FTP Performance.

Figure 4.3: Non-QoS sensitive Applications. Performance of Web and FTP stations when increasing the number of Clusters in the network, in terms of Power Consumption (upper graph) and Average Web Page Delay/Average FTP Connection Throughput (lower graph).

Besides the expected general performance trends, there is an interesting effect in the case of the FTP application which we describe in detail next. As observed in Figure 4.3(b), aggregation significantly improves performance when the network is congested. However, when the network is not congested (5 clusters) the throughput experienced by a FTP file transfer is significantly higher without aggregation (802.11 PSM) than with aggregation (802.11 PSM+ZFA). This effect is similar to the *performance inversion* effect originally pointed out in (39) and can be explained in the following way. If when new TCP packets arrive at the AP, a station has already retrieved all the packets that were buffered in the AP, then the station is in sleep mode and the connection throughput degrades because these new TCP packets have to wait until the next Beacon to be transmitted. On the other hand, if the new TCP packets arrive before the previously buffered TCP packets have been transmitted, the station remains awake and the transfer continues as in the active mode case. The latter case is much more likely to happen when no aggregation is used because the TCP packets buffered in the AP take longer to be transmitted.

4.2 CA-DFA: Congestion Aware-Delayed Frame Aggregation

In this section we design and evaluate an algorithm that dynamically adjusts the amount of frame aggregation used by 802.11n stations according to the level of congestion in the network. Hereafter we refer to this algorithm as the *Congestion Aware-Delayed Frame Aggregation* (CA-DFA) algorithm. Our goal is to leverage the 802.11n aggregation mechanisms in order improve QoS, power consumption and network capacity.

4.2.1 Discussion

The intuition behind the design of our CA-DFA algorithm is based on the insights obtained in the performance evaluation study presented in Section 4.1. As we have seen, the benefits of aggregation clearly depend on the amount of congestion in the network. When congestion is low, aggregation may unnecessarily introduce delay that harms the QoS of time sensitive applications like Voice or Video. However, when congestion in the network increases aggregation makes a more efficient use of wireless resources and increases network capacity.

The design of CA-DFA relates to the aggregation mechanisms proposed in (87) where the authors introduced two mechanisms to increase the efficiency of VoIP in Wi-Fi networks: i) a *Zero-delay Frame Aggregation* (ZFA) mechanism and ii) a CW_{min} adaptation algorithm. As previously explained, the ZFA mechanism aggregates multiple VoIP packets addressed to a certain station in the same MAC frame, whenever more than one VoIP packet is ready for transmission in the MAC queue of either the AP or the stations, and does not introduce any extra delay. This is the same scheme used by the *802.11 PSM+ZFA* and *U-APSD+ZFA* configurations that we studied in Section 4.1. In addition, the CW_{min} adaptation algorithm allows to increase aggregation opportunities in the stations that otherwise would mostly occur only in the downlink direction (AP). This is achieved by allowing the AP to use a smaller CW_{min} than the stations, such that a symmetric amount of bandwidth is allocated in uplink and downlink.

The major difference between our CA-DFA algorithm and (87) is that CA-DFA dynamically introduces delay in order to increase the number of aggregation opportunities instead of adapting contention settings. The idea of delaying access to the channel in order to increase aggregation in 802.11n was first introduced in (94), where the authors identified that traffic flowing through high priority Access Categories (ACs) in EDCA achieves a low aggregation efficiency because it accesses the channel with small delays. Notice, that in the previous section we have identified that a similar phenomena arises when 802.11n frame aggregation is combined with the U-APSD protocol, which triggers the AP too often. In order to resolve the previous conflict the authors in (94) introduced a *Delayed Channel Access* (DCA) algorithm where stations intentionally delay access to the channel in order to create more efficient aggregates. Like in CA-DFA, the extra delay introduced by DCA is proportional to the level of congestion in the network. Further work in (95) and (96) identified several problems that arise when DCA is applied to TCP flows and proposed solutions to improve performance in this case. Although both DCA and CA-DFA increase efficiency by introducing delay in the MAC layer, their focus differs in that DCA is optimized for stations that are in Active Mode and CA-DFA will be optimized for stations that are in power saving. For instance, in DCA all stations and the Access Point (AP) mesure their channel access delay and adjust their level of aggregation accordingly. However, an AP can not implement an algorithm like DCA when the stations are in power saving, because in this case the AP can only transmit after receiving a trigger frame from the station, which is the reason why we will design CA-DFA to control the trigger interval used in the stations.

4.2.2 CA-DFA design

CA-DFA follows the design guidelines outlined in Chapter 3. First, we want the algorithm to operate at layer two (in the Wi-Fi driver). This allows to reuse the presented algorithm in any type of device with a Wi-Fi interface and to easily apply it to different applications. Second, in order to guarantee an easy deployment, the designed algorithm should not require any modification to current 802.11 standards. Finally, the proposed algorithm should be able to run in a distributed way in each station in the Wi-Fi network.

CA-DFA runs in a Wi-Fi station and is composed of two independent modules:

i. *Congestion Estimation*: A module that estimates the level of congestion in the network.

ii. *Dynamic Aggregation*: A module that adjusts the amount of MAC aggregation according to the current level of congestion.

Since reducing power consumption is one of our key design objectives, the algorithm implementation presented in this chapter is optimized for stations in power save mode. However, the presented principles can also be applied when stations do not run a power saving protocol. Next, we present a detailed description of the previous modules.

4.2.2.1 Measuring congestion

There are several metrics in Wi-Fi that correlate to the amount of congestion in the network and could be used for our purposes. For instance: number of retransmissions, access delay (time since a packet is first in the transmission queue until it is successfully transmitted) and channel utilization.

The previous metrics though can only sense the amount of congestion experienced locally by each station, which is a severe limitation in Wi-Fi infrastructure networks, where the bottleneck is usually located in the AP (87). Thus, depending on the selected metric, stations could sense a non-congested medium whilst the AP would experience a severe backlog in its transmission queues, degrading the QoS of Voice or Video applications. In addition, some of these metrics, e.g. channel utilization, would require a station to continuously listen to the medium and are therefore, not applicable to stations in power save mode.

There is however a generic solution to the problem of estimating the congestion experienced by the AP when stations employ a power saving protocol. For instance, in the U-APSD case the AP can only send data to a station after receiving a trigger frame from the station, i.e. when a *service period* starts. Typically, upon receiving a trigger frame the AP extracts the corresponding data frame from the power save buffer and inserts it in its transmission buffer. Therefore, the time required by the AP to complete this transmission, and signal the end of the service period, is correlated with the amount of congestion that it experiences. Therefore, a U-APSD station can sense the amount of congestion experienced by the AP simply by *measuring the duration of its service periods*.

In general, in order to account for the possibility of congestion occurring both in uplink and downlink, CA-DFA uses two simultaneous metrics to estimate congestion:

- Average Service Period Duration (avg_SP_{dur}): used to estimate downlink congestion. This metric is updated every time the station initiates a service period using an Exponentially Weighted Moving Average (EWMA) filter, i.e. $avg\_SP_{dur}(n+1) = (1-\alpha)avg\_SP_{dur}(n)+\alpha last\_avg\_SP_{dur}$.
- Average Access Delay (avg_access_delay): used to estimate uplink congestion. This metric is also updated using an EWMA estimate every time a station completes a transmission in uplink. The same α value is used in the two EWMA filters.

As a result, CA-DFA is able to react upon both uplink and downlink congestion by considering as its most reliable congestion estimate the maximum between the previous two congestion variables, i.e. $cong\_est = \max\{avg\_SP_{dur}, avg\_access\_delay\}$.

4.2.2.2 Adjusting the level of aggregation

Building on the congestion measurements introduced in the previous section, CA-DFA dynamically adjusts the amount of MAC aggregation based on the following heuristic:

- If the estimated congestion is below a certain threshold, the amount of MAC aggregation is reduced in order to benefit from reduced delays.
- If the estimated congestion is above a certain threshold, the amount of MAC aggregation is increased in order to reduce the level of congestion in the network.

The detailed operation of CA-DFA is illustrated in Algorithm 4.1 and described next.

In order to control the level of aggregation, CA-DFA uses the concept of a *Service Interval* (SI) that is the minimum interval between two transmission attempts in the medium. Thus, at times $t_j = t_0+jSI$, a station inspects its transmission buffer, aggregates all buffered MPDUs in one or more A-MPDUs, and inititates a transmission attempt. Any data that arrives at the station between transmission attempts is buffered awaiting for potential aggregations until the next transmission attempt. In addition, if at t_j there is no buffered data, a station in power save mode still generates a signaling trigger to poll the AP for its buffered data. Notice that the service interval, SI, controls the trade-off between delay and aggregation.

Real-time applications are the main target of CA-DFA, therefore a sensible approach is to adjust SI at multiples of SI_{min}, i.e. $SI = L \times SI_{min}$, where SI_{min} is the minimum interval used between consecutive transmission attempts (e.g. codec packet generation interval in the case of VoIP), $1 \leq L \leq \lfloor \frac{D}{SI_{min}} \rfloor$ is an integer value that defines the service interval currently being used, and D is a delay bound for a given application or Access Category. Depending on each particular implementation, the values of SI_{min} and D could be configured according to the applications expected to run on each Access

Algorithm 4.1: CA-DFA Algorithm.

```
– Variables definition
$D \leftarrow$ Maximum delay bound tolerated by a certain AC.
$SI_{min} \leftarrow$ Minimum interval between transmissions for a certain AC.
$T_{mon} \leftarrow$ Time period between consecutive evaluations of the medium.
$count\_limit \leftarrow$ Number of consecutive $T_{mon}$ periods between updates.
$\beta, \gamma \leftarrow$ Constants used to derive the Threshold values.
$LP\_aware \leftarrow$ Make CA-DFA aware of low priority applications
– Inititalization routine
$L \leftarrow 1, L_{min} \leftarrow 1, L_{max} \leftarrow \lfloor \frac{D}{SI_{min}} \rfloor$
$SI \leftarrow L \times SI_{min}$
$THR_{up} = \beta SI$
$THR_{down} = \gamma \max\{L_{min}, (L-1)\} SI_{min}$
$up\_count \leftarrow 0$
$down\_count \leftarrow 0$
$t_{init} \leftarrow current\_time$
$L_{up}, L_{down}, first\_turn \leftarrow false$
$t\_SI_{start}, t\_outer_{start} \leftarrow t_{now}$
– Routine executed at $t_{mon}(k) = t_{init} + kT_{mon}$
$cong\_est \leftarrow \max\{avg\_SP_{dur}, avg\_access\_delay\}$
$update \leftarrow false$
if $(cong\_est > THR_{up})$ or $(LP\_aware$ and $LP\_starve)$ then
    $up\_count \leftarrow up\_count + 1$
    $down\_count \leftarrow \max\{0, down\_count - 1\}$
    if $up\_count \geq count\_limit$ then
        $L \leftarrow \min\{L_{max}, L + 1\}$
        $update \leftarrow true$
else if $(cong\_est < THR_{down})$ and $!(LP\_aware$ and $LP\_starve)$ then
    $down\_count \leftarrow down\_count + 1$
    $up\_count \leftarrow \max\{0, up\_count - 1\}$
    if $down\_count \geq count\_limit$ then
        $L \leftarrow \max\{L_{min}, L - 1\}$
        $update \leftarrow true$
else
    $up\_count \leftarrow \max\{0, up\_count - 1\}$
    $down\_count \leftarrow \max\{0, down\_count - 1\}$
if $update$ is $TRUE$ then
    $up\_count \leftarrow 0$
    $down\_count \leftarrow 0$
    $SI \leftarrow L \times SI_{min}$
    $THR_{up} = \beta SI$
    $THR_{down} = \gamma \max\{L_{min}, (L-1)\} SI_{min}$
    $outer\_control\_alg()$
– Routine executed when receiving a Beacon
$AP\_delay[AC_i] \leftarrow BSS\_AC\_Access\_Delay\_Elem$
if $\max\{AP\_delay[AC_{BE}], AP\_delay[AC_{BK}]\} > THR_{LP}$ then
    $LP\_starve \leftarrow true$
else
    $LP\_starve \leftarrow false$
```

Category, or could even be dynamically set up by the higher layers. In addition, when a station uses U-APSD, SI_{min} can be configured to be equal to the default polling interval, so that when $L = 1$ no further delays are introduced on top of U-APSD.

CA-DFA maintains two thresholds, THR_{up} and THR_{down}, that are defined as proportions, $0 < \gamma < \beta < 1$ of the current service interval (lines 11, 12, 40 and 41). A station periodically executes Algorithm 4.1 and checks its congestion estimate against these thresholds in order to decide if SI should increase or decrease. Specifically, when congestion is above $THR_{up} = \beta SI$ (lines 21-26), SI is increased in order to mitigate congestion, and when congestion is below $THR_{down} = \gamma \max\{L_{min}, (L-1)\} SI_{min}$ (lines 27-32), SI is decreased[1] in order to benefit from reduced delays. CA-DFA always updates the current service interval in discrete steps equal to SI_{min} (lines 25, 31 and 39).

CA-DFA can be understood as a distributed control loop where each station tries to maintain its congestion estimate, i.e. $cong\_est = \max\{avg\_SP_{dur}, avg\_access\_delay\}$, between THR_{up} and THR_{down}, by means of adjusting its individual service interval (the control output). A consequence of the previous dynamics is that, if the AP is the bottleneck, then $cong\_est = avg\_SP_{dur}$, and CA-DFA provides a soft bound on the worst case average power consumption of a Wi-Fi station regardless of the level of congestion. The reason is that the average time a U-APSD station stays awake can be expressed as $\frac{avg\_SP_{dur}}{SI}$. Thus, given that CA-DFA adjusts SI in order to maintain $avg\_SP_{dur} \leq THR_{up}$, the average duty cycle of a CA-DFA station can be bounded by:

$$\frac{avg\_SP_{dur}}{SI} \leq \frac{THR_{up}}{SI} = \beta \tag{4.1}$$

Notice though that setting β too small reduces the hysteresis margin, $THR_{up} - THR_{down}$, and increases the frequency at which CA-DFA updates the service interval, which may increase delay due to spurious service interval updates.

Two parameters are defined in CA-DFA that aid a station to filter out spurious congestion events in order to minimize the number of SI updates. These parameters are the monitoring interval T_{mon}, at which a station executes Algorithm 4.1 in order to update SI, and the $count_{limit}$ parameter, which is a minimum number of monitoring intervals where the congestion variables must be above THR_{up} or below THR_{down} before the station updates its service interval (lines 24 and 30). In the next section we will study how to appropriately set these parameters.

[1]Notice that THR_{down} is defined upon the service interval that will be used if the current one is decreased. The intention is to be confident that the congestion experienced with a reduced service interval will be acceptable.

We have so far described how CA-DFA adjusts its service interval when it detects congestion in the channel. Next we present two optimizations to the basic behavior of CA-DFA.

4.2.2.3 CA-DFA Optimizations

Low Priority Aware CA-DFA

Our first optimization relates to the fact that the EDCA protocol provides a better shielding from congestion to high priority applications than to low priority ones. Therefore, in a multi-service network, low priority applications may starve before high priority ones start to experience any significant congestion that would trigger CA-DFA to increase the used service interval. Our goal thus, is to improve the performance experienced by low priority applications when CA-DFA is used while keeping an acceptable QoS for high priority ones.

If high priority applications do not need to operate at their minimum possible delay, the basic CA-DFA behavior described in the previous section can be slightly modified in order to benefit low priority applications. Hereafter, we refer to this CA-DFA variation as *Low Priority Aware* CA-DFA. The idea behind this optimization is to let a CA-DFA station running a real-time application increase its service interval not only when it directly experiences congestion, but also when low priority applications in the network begin to starve. This functionality is implemented in Algorithm 4.1 simply checking the flag LP_starve when deciding whether to increase or decrease the current service interval (lines 21 and 27). The obvious question then is how can a station in power save mode running a real-time application learn about the congestion experienced by low priority applications that may be running in other stations.

In order to solve this problem we decide to make use of the *BSS_AC_Access_Delay_Element* defined in 802.11k (71), which is embedded by the AP in every Beacon frame, and indicates the access delay experienced by the AP in each Access Category. Therefore, in order to discover whether low priority traffic is starving, a CA-DFA station periodically wakes up for the Beacon, learns about the congestion in the network experienced in low priority ACs, and compares this value to a pre-configured congestion threshold, THR_{LP} (lines 44-48). Note that CA-DFA can dynamically enable or disable its *Low Priority Aware* configuration using the flag LP_aware (lines 21 and 27).

Improving Service Interval Stability

Our second CA-DFA optimization relates to the goal of stabilizing oscillations in the service interval selected by Algorithm 4.1. For this purpose, we propose an outer control algorithm executed every time CA-DFA updates the used service interval (line 42 in Algorithm 4.1).

As it will be shown in the next section, in a network with downlink congestion CA-DFA stations may get synchronized. The reason is that stations experience a correlated congestion because their packets go through the same queue in the AP. Thus, upon congestion the queues in the AP become backlogged, all stations observe high service period durations, and increase their service interval. However, a synchronous increase in the service interval of many stations can suddenly push down congestion in the network, hence triggering a synchronous decrease in the service interval of many stations which again results in an increase of congestion. This synchronization introduces a harmful oscillatory behavior in the stations service interval, which may result in undesired delays. The goal of the outer control algorithm is to detect and avoid these oscillations.

Our outer control algorithm is described in Algorithm 4.2 and is executed every time the service interval is updated in Algorithm 4.1. Its basic principle is to maintain a histogram of the service intervals being used by a station, which is computed considering the amount of time that each service interval is used (lines 10 to 12 in Algorithm 4.2).

Every time the used service interval is updated, Algorithm 4.2 evaluates whether an *oscillation* in the used service intervals has occurred, where an oscillation is defined as an *increase-decrease-increase* cycle (line 15), or viceversa (line 17). Thus, when an oscillation is detected the algorithm evaluates whether too aggressive service intervals are being used that should be pruned. For this purpose, the average service interval experienced during the detected oscillation is computed (line 22), and if service intervals smaller than the average one occur with a specific probability, these smaller service intervals are pruned from the allowed set of service intervals (lines 20 to 32).

The intuition behind this algorithm is that by avoiding the use of aggressive (small) service intervals, transient congestion (which forces a station to react using high service intervals) will be avoided and service intervals will stabilize. As a conservative measure though, depending on the number of service intervals present in the oscillation, we only allow to prune at maximum one or two of the service intervals in the oscillation (line 25). Specifically, we prune small service intervals in an oscillation only if these occur with a probability $p_{min} < p < p_{max}$ (line 27), with the intention that a small interval occuring very seldomly is not pruned and a small interval that is the predominant interval used within the oscillation is also not pruned. The performance of this algorithm will be carefully evaluated in the next section.

Given that load in a network changes dynamically, a mechanism is needed to revert the action of this algorithm and allow a CA-DFA station to benefit again from small service intervals. For this purpose CA-DFA makes use of another Information Element defined in 802.11k, the *BSS Load Element* (71). This element is transmitted by the AP within the Beacon frame and advertises the current level of

Algorithm 4.2: Outer Control Algorithm.

```
– Variables definition
$i_{max} \leftarrow L_{max} - L_{min} + 1$ number of allowed service intervals
$SI\_hist_i, i = 1...i_{max} \leftarrow SI$ histogram
$L \leftarrow L_0$ such that current $SI = L_0 SI_{min}$
$L_{up}, L_{down} \leftarrow$ track if $SI = LSI_{min}$ is increasing or decreasing
$L_{last} \leftarrow L_1$ such that last $SI = L_1 SI_{min}$
$update\_now \leftarrow false$
– $outer\_control\_alg()$
//Computing the $SI$ histogram
$t\_SI_{end} \leftarrow t_{now}$
$SI\_hist_L = SI\_hist_L + (t\_SI_{end} - t\_SI_{start})$
$t\_SI_{start} \leftarrow t_{now}$
//Detect oscillations
if $L > L_{last}$ then
    detect_oscillation($L_{down}$,$L_{up}$)
else if $L < L_{last}$ then
    detect_oscillation($L_{up}$,$L_{down}$)
//Pruning too aggressive Service Intervals
if $update\_now$ then
    $t\_outer_{end} \leftarrow t_{now}$
    $\forall i,\ SI\_hist_i \leftarrow \frac{SI\_hist_i}{t\_outer_{end} - t\_outer_{start}}$
    $SI_{avg} \leftarrow \sum_{i=1}^{i_{max}} SI\_hist_i \times i \times SI_{min}$
    $i_{init} \leftarrow \min_i\{SI\_hist_i > 0\}$
    $num\_present \leftarrow |SI\_hist_i > 0|$
    $max\_pruning \leftarrow i_{init} + \max\{\lfloor \frac{num\_present-1}{2} - 1 \rfloor, 0\}$
    $p \leftarrow \sum_{l=i_{init}}^{i_{init}+max\_pruning} SI\_hist_l | l \times SI_{min} < SI_{avg}$
    if $p_{min} < p < p_{max}$ then
        $L_{min} \leftarrow l + 1$
        $saved\_util \leftarrow last\_beacon\_util$
    $t\_outer_{start} \leftarrow t_{now}$
    $first\_turn \leftarrow false$
    $\forall i,\ SI\_hist_i \leftarrow 0$
$L_{last} \leftarrow L$
– detect_oscillation($L_1$,$L_2$)
if !$L_1$ then
    $L_2 \leftarrow true$
else if $L_1$ then
    $L_1 \leftarrow false, L_2 \leftarrow true$
    if !$first\_turn$ then
        $first\_turn \leftarrow true$
    else
        $update\_now \leftarrow true$
– Routine executed when receiving a Beacon
$last\_beac\_util \leftarrow BSS\_load\_Elem$
if $L_{min} > 1$ then
    $saved\_util \leftarrow \max\{last\_beac\_util, saved\_util\}$
    if $last\_beac\_util < \rho \times saved\_util$ then
        $L_{min} \leftarrow 1$ after random delay
```

utilization in the network. Thus, when Algorithm 4.2 restricts the set of allowed service intervals, it also records the current utilization advertised by the AP (line 29). Thereafter, if when receiving a Beacon frame a station observes an utilization level significantly below the recorded one, i.e. $\rho \times saved\_util$, with $0 < \rho < 1$ in line 47, the set of allowed service intervals is restablished in order to reconverge to a suitable service interval[1].

Finally, we would like to notice that the two proposed CA-DFA optimizations rely on functionality which is currently being mandated by the Wi-Fi Alliance in the Voice over Wi-Fi Enterprise certification (93). We therefore believe that it is reasonable to assume the wide availability of this functionality in the near future.

4.2.2.4 Complexity of CA-DFA

Since the main goal of CA-DFA is to reduce power consumption, together with providing a good QoS, it is important to asses that the computational complexity required to implement CA-DFA, which can be entirely implemented in software within a Wi-Fi driver, does not negatively impact power consumption. Next, we analyze the complexity incurred by the different modules that compose CA-DFA:

- First, as explained in Section 4.2.2.1, CA-DFA manages to measure congestion in a way that is completely seamless to the operation of the Wi-Fi power saving protocols. In addition, CA-DFA updates its EWMA estimates when a service period or a channel access completes. We consider that maintaining an EWMA estimate has a minimal impact in terms of complexity.

- Second, the basic version of CA-DFA described in Algorithm 4.1 is only executed every T_{mon} seconds which, as it will be discussed in the next section, is typically a multiple of the Beacon interval. Notice that current Wi-Fi chipsets already perform several operations when receiving a Beacon frame. In addition, Algorithm 4.1 only involves comparisons between the current congestion estimates and THR_{up} and THR_{down}. We therefore claim that the computational impact of this algorithm is minimal.

- Finally, the outer control algorithm described in Algorithm 4.2 is in the worst case executed every $T_{mon} \times \min\{down\_count, up\_count\}$ seconds. The $detect\_oscillation()$ procedure therein has as well a minimal complexity since it only involves comparing certain state variables. In case an oscillation is detected, which can occur in the worst case only every:

$$2 \times T_{mon} \times \min\{down\_count, up\_count\} + T_{mon} \times \max\{down\_count, up\_count\}$$

[1]To avoid that different stations get synchronized, CA-DFA randomizes within 10 seconds the reuse of the pruned service intervals.

seconds, since it requires at least an $increase \rightarrow decrease \rightarrow increase$ cycle (or viceversa), CA-DFA performs a set of computations that linearly depend on the maximum number of allowed service intervals i_{max}. Notice that in a practical setting one wants to keep the delays in a Wi-Fi network under $\sim 200ms$ or less which, if a typical minimal service interval of $SI_{min} = 20ms$ is considered, results in a reduced i_{max} of $\frac{200}{20} = 10$.

For the reasons previously stated, we claim that the extra computational complexity required to implement CA-DFA has a negligible impact on power consumption.

4.2.3 CA-DFA dynamics and parameter setting

In this section we illustrate the dynamics of CA-DFA. Given the complexity to accurately model the considered protocols in an analytical way, we use the following simulation scenario that will ease the task of identifying the properties of CA-DFA and tuning its parameters. We consider a Hotspot scenario like the one of Section 4.1, where all stations run a bi-directional CBR application of 256Kbps over AC_VI with a packet interarrival time of $20ms$ and use a fixed transmission rate of 39Mbps (16 QAM 3/4). The rest of simulation parameters are identical to the ones described in Section 4.1.

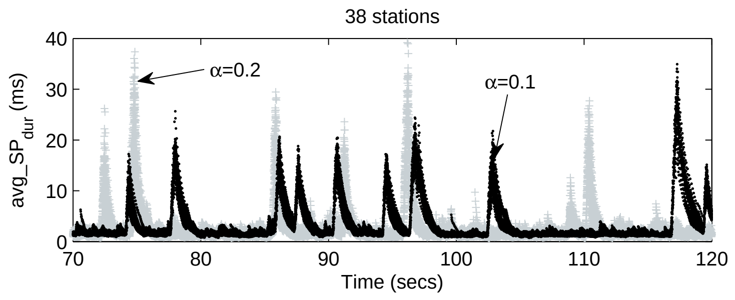
In our evaluation the maximum allowed delay bound and the minimum service interval in CA-DFA are set to $D = 100ms$ and $SI_{min} = 20ms$ respectively. In addition we configure $\gamma = \frac{\beta}{2}$, which eases the task of configuring CA-DFA because the hysteresis margin becomes dependent only on the parameter β. Specifically:

$$\Delta_{THR} = \begin{cases} \beta \times \frac{SI_{min}}{2} \times (L+1) & L > 1 \\ \beta \times \frac{SI_{min}}{2} & L = 1 \end{cases} \tag{4.2}$$

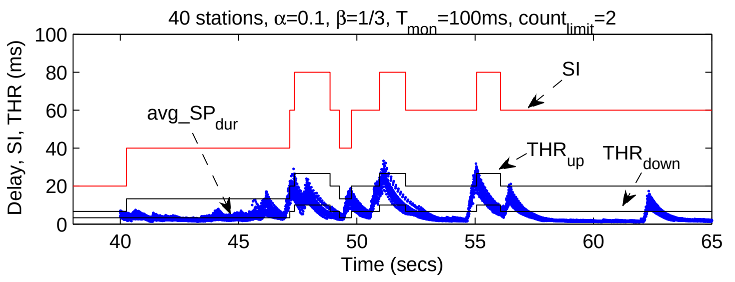
Therefore, increasing β increases both the hysteresis margin (reducing the number of service interval updates) and the tolerated power consumption, as explained in the previous section. The effect of the CA-DFA parameters β, T_{mon} and $count_{limit}$ will be studied in this section.

In order to illustrate how CA-DFA tracks the congestion in the network, Figure 4.4(a) depicts the most restrictive congestion metric, avg_SP_{dur}, experienced by *all* the stations in a scenario containing 38 stations. The congestion metric is plotted for two different values of α in the EWMA filter, $\alpha = 0.1$ (black dots) and $\alpha = 0.2$ (grey crosses). As clearly observed in the figure, when the AP is backlogged, all stations experience a sudden increase in their congestion metric for both values of α.

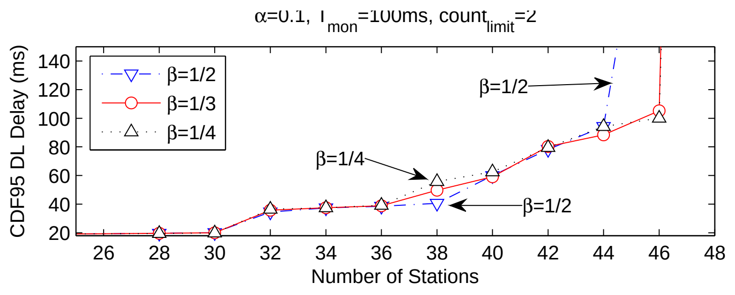
Figure 4.4(b) illustrates the basic dynamics of CA-DFA, where a sample station tracks its congestion metric in a scenario with 40 stations. It can be observed how the service interval, SI, and congestion thresholds, THR_{up} and THR_{down} are updated depending on the measured congestion according to



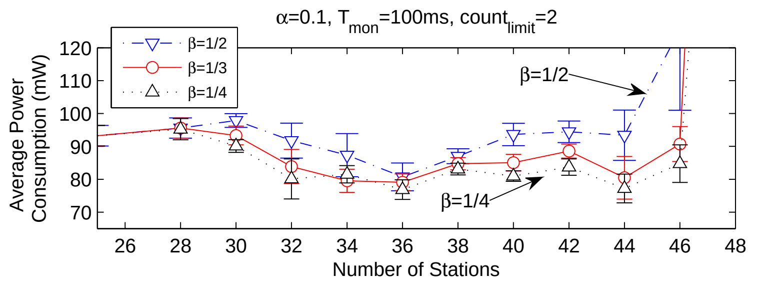
(a) CA-DFA stations experiencing correlated congestion. Congestion spikes are clearly identified with $\alpha = 0.1$ and $\alpha = 0.2$.



(b) Dynamics of SI,THR_{up} and THR_{down} as congestion varies.



(c) Effect of β on worst case (cdf95) delay.



(d) Effect of β on average power consumption.

Figure 4.4: CA-DFA Dynamics-1.

Algorithm 4.1. Notice how initially CA-DFA reacts in order to maintain the experienced congestion between THR_{up} and THR_{down}. However, after 57 seconds the experienced congestion falls below THR_{down} but the service interval remains unchanged. The reason is the operation of the outer control algorithm which upon the oscillation occurring between 47 secs and 52 secs, prunes service intervals of 20ms and 40ms. The efficiency of this mechanism will be analysed in detail later in this section.

In order to study the effect of β, where $THR_{up} = \beta \times SI$, in the performance of CA-DFA, Figures 4.4(c) and 4.4(d) depict respectively the 95% of the delay cumulative distribution function (cdf95) and average power consumption experienced by the stations in our reference scenario, as the number of stations in the network increases. Three different values of $\beta = \{1/2, 1/3, 1/4\}$, are considered in our evaluation. Looking first at the effect of β on delay in Figure 4.4(c), we can see how, regardless of β, delay tends to increase in a step-wise fashion due to the fact that CA-DFA selects bigger service intervals as congestion increases. The effect of β can be understood looking at the delay curves with $\beta = 1/2$ and $\beta = 1/4$. When congestion starts to appear but is not yet irrecoverable, e.g. at 38 stations, a bigger β may result in smaller delays because in that case CA-DFA tolerates a higher amount of congestion before selecting higher service intervals. However, when congestion increases close to network capacity, e.g. after 44 stations, using a big β results in a smaller network capacity (defined as the number of stations which result in delays below the configured delay bound) than using a smaller β. The reason is that when using a bigger β CA-DFA requires longer times to react to congestion, which leads to persistent congestion states when the network operates close to capacity. Looking at power consumption in Figure 4.4(d), we can see as expected that even when the network is not congested, a smaller β results in a smaller average power consumption. In addition, notice how by means of increasing the service interval, CA-DFA manages to effectively upper bound the average power consumption experienced by the stations until the network overloads.

In Figure 4.5(a) we repeat the previous experiment but considering the effect of the parameters T_{mon} and $count_{limit}$. Specifically, we consider 6 different configurations, $T_{mon} = \{100, 200, 400ms\}$ with $count_{limit} = 2$ (solid lines), and $T_{mon} = \{100, 200, 400ms\}$ with $count_{limit} = uniform(2, 5)$ (dash-dotted lines). Notice that we study a randomized $count_{limit}$ because this unsynchronizes CA-DFA stations, and might thus be benefitial. Figure 4.5(a), depicts the worst case delay (cdf95) results for this experiment. We can see in this figure how the best performing configuration is the one with the minimum T_{mon} and $count_{limit}$, i.e. $T_{mon} = 100ms$ and $count_{limit} = 2$. The reason is that increased values of T_{mon} or $count_{limit}$ result in longer times for CA-DFA to react to congestion, and as we have previously seen this reaction time turns out to be critical when the network operates close to capacity.

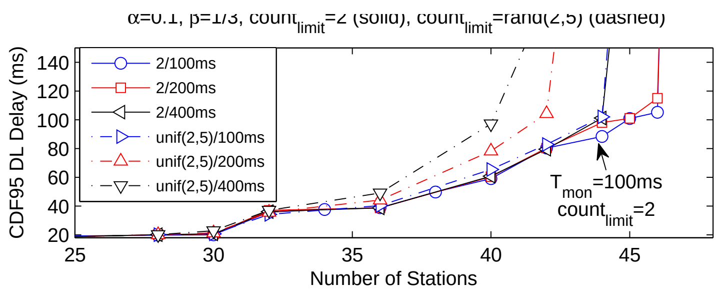
For the sake of space, we do not plot the power consumption results for this case which basically follow the same dynamics like the delay ones.

In order to study how far CA-DFA operates from an *optimal* dynamic service interval algorithm, Figure 4.5(b) depicts the worst case delay (cdf95) performance of CA-DFA as compared to a set of fixed service interval configurations, where the service interval is varied from 20ms to 100ms in steps of 20ms. Figure 4.5(b) clearly depicts the trade-offs of a fixed service interval approach, i.e. delay versus network capacity, that CA-DFA strives to resolve. Indeed, we can see that as congestion increases the worst case delay obtained with CA-DFA (red solid line) closely follows the minimum among all the fixed delay configurations. Notice though that CA-DFA tends to switch to a higher service interval slightly before the correspondent fixed configuration becomes completely congested, e.g. at 42 stations. The reason is that power consumption in those cases is above the configured threshold. As we have seen before, delay could be reduced using a bigger value for β at the price though of a smaller network capacity and a higher power consumption.

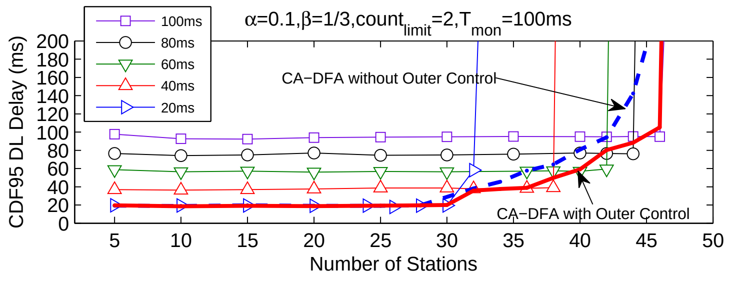
Figure 4.5(b) also depicts the performance of CA-DFA without the outer control algorithm described in Algorithm 4.2 (blue dashed line). We can clearly see how the performance of CA-DFA degrades when the outer control algorithm is not used. As explained in the previous section, the reason for this degradation is the service interval oscillations caused by the synchronization between the different CA-DFA stations. These oscillations and the correspondent stabilization obtained by the outer control algorithm are illustrated in Figure 4.5(c) for a sample station when we have 44 stations in the network. The values of p_{min} and p_{max} in Algorithm 4.2 were empirically set, in this experiment and in the rest of the chapter, to 0.05 and 0.6 respectively, which showed a good performance in our applications of interest.

Finally, Figure 4.5(d) depicts the service interval and the per packet delay experienced by a sample station in a scenario where we dynamically vary the load in the network. Specifically, 46 stations enter the network between 0 and 40 secs, and thus the sample station increases its service interval up to 100ms. However, ten stations leave the network at both 60 secs and 110 secs. This fact can be recognized by a CA-DFA station looking at the channel utilization advertised by the AP in the Beacon frame ($\rho = 0.8$ was used in this experiment). Thus, every time the channel load reduces, CA-DFA resets the operation of the outer control algorithm and the station reconverges to a service interval appropriate to the current level of congestion in the network.

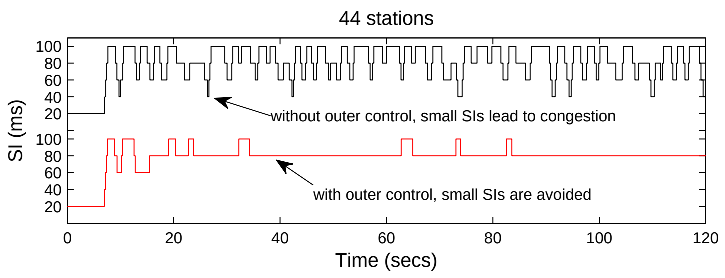
After having analysed the basic dynamics of CA-DFA, we evaluate in the next section the performance of CA-DFA to be expected in a realistic scenario.



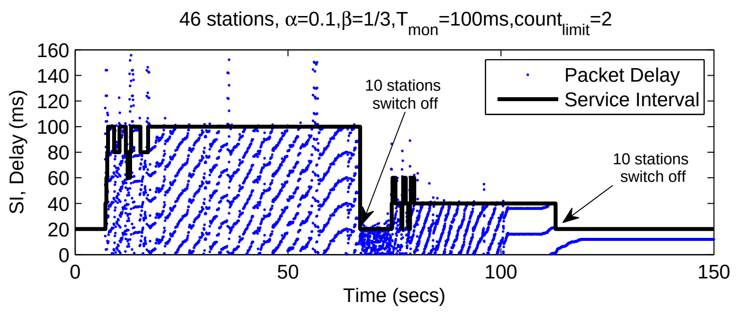
(a) Effect of T_{mon} and $count_{limit}$ on worst case (cdf95) delay.



(b) Comparison between CA-DFA and a fixed SI approach.



(c) Effect of the outer control algorithm on the SI oscillations.



(d) CA-DFA SI adaptation with a varying network load. The used SI upper bounds the experienced delay.

Figure 4.5: CA-DFA Dynamics-2.

4.3 CA-DFA performance evaluation

In this section we present the results of a simulative study that evaluates the performance of our CA-DFA algorithm compared to the one of the current Wi-Fi QoS and power saving protocols.

4.3.1 Simulation framework

Exactly the same simulation framework described in Section 4.1.1 is used to evaluate the performance of CA-DFA. In addition, being CA-DFA a distributed scheme that relies on each individual station autonomously reacting to congestion, a key aspect to asses its performance in practice is to evaluate the benefits obtained when only part of the stations in the network implement CA-DFA. For that purpose, we evaluate the following protocol configurations:

- *802.11 PSM+ZFA*, where all stations implement 802.11 PSM and a zero delay frame aggregation scheme is used.
- *U-APSD+ZFA*, where all stations implement U-APSD and a zero delay frame aggregation scheme is used.
- *CA-DFA-1/3*, where one third of the Voice and Video stations implement CA-DFA, and two thirds of the Voice and Video stations implement U-APSD+ZFA.
- *CA-DFA-2/3*, where two thirds of the Voice and Video stations implement CA-DFA, and one third of the Voice and Video stations implement U-APSD+ZFA.
- *CA-DFA-all*, where all Voice and Video stations implement CA-DFA.
- *Low Priority Aware CA-DFA-all*, where all Voice and Video stations implement the *Low Priority Aware* variant of CA-DFA.

Notice that in the previous configurations, CA-DFA applies to Voice and Video stations and Web and FTP stations implement U-APSD+ZFA. In addition, no changes are required in the Access Point. According to the study presented in the previous section, we configure CA-DFA using the following parameters, $\alpha = 0.1$, $\beta = \frac{1}{3}$, $\gamma = \frac{\beta}{2}$, $T_{mon} = 100ms$, $count_{limit} = 2$, $\rho = 0.8$, $p_{min} = 0.05$ and $p_{max} = 0.6$. Finally, we set $THR_{LP} = 1.5ms$ in the *Low Priority Aware CA-DFA-all* configuration.

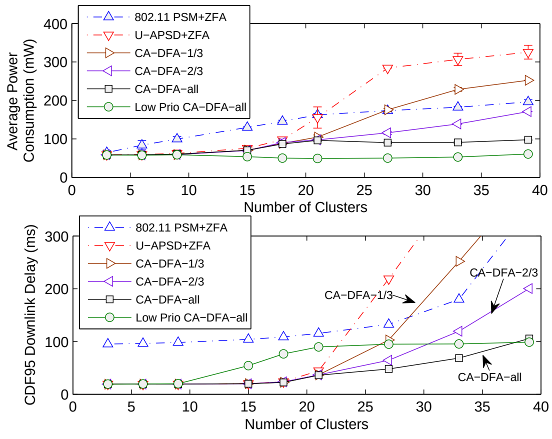
4.3.2 Performance evaluation

Next, we describe the results obtained in our study, which are again divided in: i) performance of QoS sensitive applications, i.e. Voice and Video, and ii) performance of non-QoS sensitive applications, i.e. Web and FTP download.

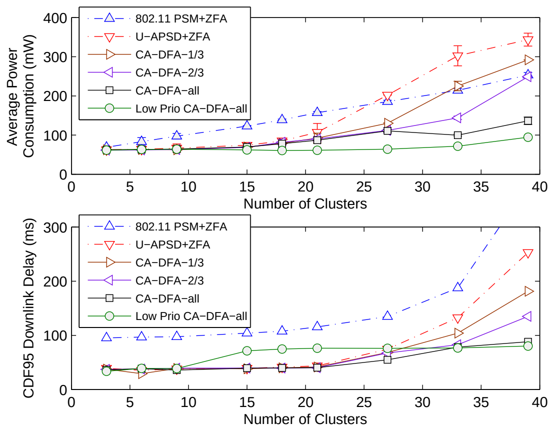
4.3.2.1 Performance of QoS sensitive applications

Figures 4.6(a) and 4.6(b) illustrate the performance of the different protocols under study in the case of the Voice and Video applications. For simplicity, we omit the throughput results in this case, and only plot power consumption (upper graph) and worst case delay (lower graph).

The protocols under study affect the performance of Voice and Video in a similar way. We can see in the figure that as the number of Voice and Video stations in the network implementing CA-DFA increases, i.e. *CA-DFA-1/3*, *CA-DFA-2/3* and *CA-DFA-all* configurations, the performance of *all* Voice and Video stations, not only those implementing CA-DFA, significantly improves under congestion, both in terms of power consumption and delay, compared to the performance obtained when all stations implement U-APSD+ZFA. In addition, the performance gains achieved with CA-DFA exhibit a linear behavior as the number of stations implementing CA-DFA increases, resulting in the *CA-DFA-all* configuration clearly outperforming both the *U-APSD+ZFA* and the *802.11 PSM+ZFA* configurations as the network gets congested.



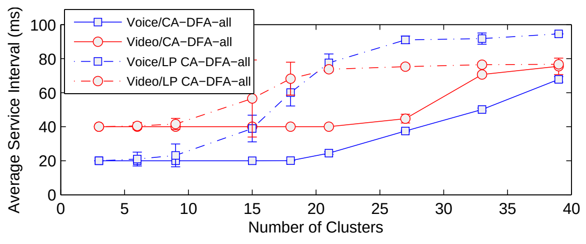
(a) Voice Performance.



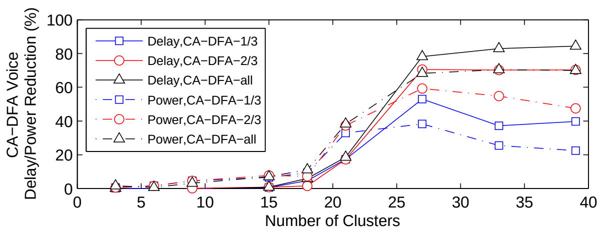
(b) Video Performance.

Figure 4.6: QoS sensitive Applications. Effect of CA-DFA on average power consumption (upper graph) and worst case delay (lower graph) for Voice and Video applications.

The performance gain obtained by the different partial deployments of CA-DFA, as compared to the basic *U-APSD+ZFA* case, is illustrated in Figure 4.7(b) in the case of Voice showing a maximum reduction close to 80% in both power consumption and worst case delay. In addition, Figure 4.7(a) depicts the average service interval used by Voice and Video stations in the *CA-DFA-all* configuration (solid lines), where it can clearly be seen how, as congestion increases in the network, Voice and Video stations increase their service interval in order to operate with more efficient aggregations. Interestingly, we can see that Voice stations still do not operate at their maximum allowed service interval (100ms), which means that there would still be room for more Voice stations in the network.



(a) Average SI for Voice and Video stations.



(b) CA-DFA gain over U-APSD for Voice.

Figure 4.7: CA-DFA dynamics. Variation of the used SI as congestion increases, and performance gains with different deployment scenarios.

Next, we turn our attention to the performance achieved by the *Low Priority Aware CA-DFA-all* configuration. We can see in the lower part of Figures 4.6(a) and 4.6(b), how this configuration introduces higher delays in the Voice and Video applications because it reacts not only to the congestion experienced by the high priority applications, but also to the congestion experienced by the low priority applications, hence resulting in higher service intervals being used. The service intervals used by the Voice and Video stations in this configuration are also depicted in Figure 4.7(a) (dashed lines). Note that a positive side effect of the friendliness to non-QoS sensitive applications is that by using higher service intervals and efficient aggregations the sleep periods of Voice and Video stations increase resulting in a significantly reduced power consumption, as depicted in the upper part of Figures 4.6(a) and 4.6(b).

4.3.2.2 Performance of non-QoS sensitive applications

Figures 4.8(a) and 4.8(b) depict the performance of the Web and FTP applications, in terms of average power consumption (upper graph) and Web page download time or FTP throughput (lower graph).

Looking at the figures we can see how the *CA-DFA-1/3*, *CA-DFA-2/3* and *CA-DFA-all* configurations progressively improve the performance of the low priority applications as compared to the reference *U-APSD+ZFA* configuration. However, none of these configurations outperform the *802.11 PSM+ZFA* configuration in terms of Web and FTP applications. The reason is the fact that the *802.11 PSM+ZFA* configuration intrinsically gives a better treatment to the low priority applications, in spite of the Voice and Video applications, because all applications operate with an effective service interval equal to the Beacon interval, i.e. 100ms, and contend for their data after a Beacon frame sending PS-Polls that use the same priority, i.e. AC_BE. On the other hand, when using U-APSD+ZFA or CA-DFA, different applications request their buffered data in the Access Point by means of QoS-Nulls which are transmitted using the same Access Category than their intended application.

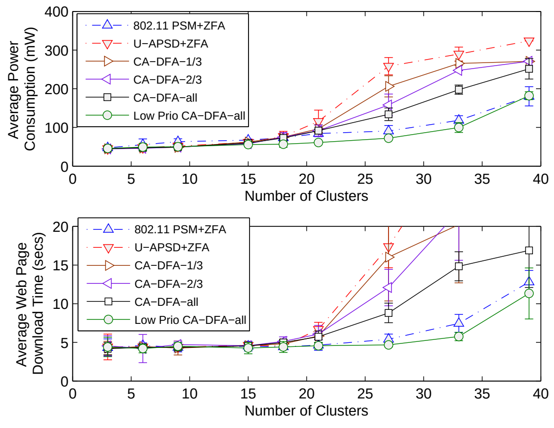
Like in the case of the QoS sensitive applications, the *Low Priority Aware CA-DFA-all* configuration exhibits a distinctive behavior. By increasing the service interval of Voice and Video application as soon as low priority applications start to experience congestion, CA-DFA benefits low priority applications and outperforms the *802.11 PSM+ZFA* configuration for both Web and FTP[1].

4.4 Conclusions

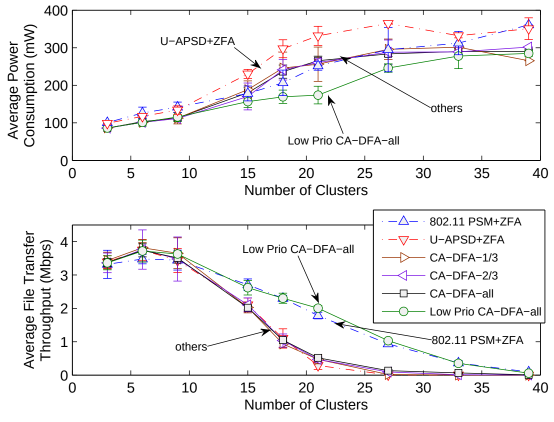
In this chapter we have presented a study of how the MAC aggregation features defined in IEEE 802.11n can be used to complement and enhance the existing Wi-Fi QoS and power saving protocols. Specifically, we have analyzed the effect of 802.11n MAC aggregation on the performance of 802.11 PSM and U-APSD and have proposed the *Congestion Aware-Delayed Frame Aggregation (CA-DFA)* algorithm that, using only information available at layer two, adapts the amount of aggregation used by a Wi-Fi device according to the level of congestion in the network.

The main conclusions that can be drawn from our results are: i) U-APSD, currently used when battery operated devices require real-time applications like Voice, is poorly suited to benefit from the 802.11n aggregation mechanisms, ii) 802.11 PSM significantly benefits from aggregation and can outperform U-APSD when congestion in the network increases, iii) An algorithm like CA-DFA that adapts

[1]Notice that the throughput performance experienced by FTP does not significantly improve with respect to the *802.11 PSM+ZFA* configuration because this comes mostly determined by the slow EDCA settings used for AC_BK together with the TCP congestion control algorithms.



(a) Web Performance.



(b) FTP Performance.

Figure 4.8: Non-QoS sensitive Applications. Effect of CA-DFA on average power consumption (upper graph) and performance (lower graph) of the Web and FTP applications in a multi-service Wi-Fi network.

the aggregation used by a Wi-Fi device according to the level of congestion in the network can significantly outperform other approaches in the state of the art in terms of power saving and network capacity, even with a gradual market penetration.

5

On Centralized Schedulers for 802.11e WLANs: Distribution vs Grouping of Resources Allocation

In Chapters 3 and 4 we have addressed the challenge of providing an energy efficient operation for real-time traffic when considering the distributed QoS and power saving protocols defined in Wi-Fi, i.e. EDCA regarding QoS and 802.11 PSM and U-APSD regarding power saving. These distributed protocols are currently widely deployed in the market, and products implementing them are certified by the Wi-Fi Alliance under the WMM certification program (35).

Unlike distributed protocols though, the centralized QoS and power saving protocols defined in 802.11, i.e. HCCA and S-APSD, have not been widely deployed in the market, due to a variety of reasons. For instance the HCCA protocol defined in 802.11e (32) failed to offer QoS guarantees when multiple Wi-Fi networks overlap, which is a situation commonly found in practice. This drawback though is currently being addressed in the 802.11aa standard (117). Another reason that hindered the deployment of these centralized protocols, was the fact that the distributed protocols were simply good enough for Data traffic, typically Web browsing, which represented most of the traffic being carried over Wi-Fi networks. However, this assumption might not hold true if Wi-Fi networks start to be commonly used to transmit High-Definition Video traffic, as indicated by the increase of Video capable devices being Wi-Fi certified (105). In addition, some studies have pointed out that some Wi-Fi products in the market do not implement the EDCA protocol in an accurate way (107). These innacuracies pose serious difficulties if advanced QoS functionalities like Admission Control have to be implemented over EDCA.

Centralized protocols like HCCA, described in Chapter 2, eliminate the need for contention to access the channel, and thus offer the possibility of increasing the efficiency of the MAC, which is essential for applications like Video. In addition, coupling HCCA with S-APSD, where the stations wake up according to a schedule delivered by the Access Point, in order to transmit and receive data without contention and quickly return to sleep, may significantly improve energy efficiency of mobile computing devices. Therefore, the focus of our work in this chapter is the analysis and evaluation of a novel energy efficient resource allocation algorithm for the centralized QoS and power saving capabilities of 802.11e. In particular, our contributions in this chapter have been published in (28) and are as follows:

- We present the design of *DRA*, a novel HCCA/S-APSD scheduler running in the Access Point, that with a very low complexity spreads in time the service periods of the stations running S-APSD, hence maximizing QoS and energy efficiency.
- A thorough performance evaluation that depicts the advantages of our DRA scheduler in front of a scheduler that groups in time the service periods of the stations running in S-APSD, as commonly done in the state of the art. Our performance evaluation considers the QoS and energy trade-off of both HCCA and EDCA traffic.

The rest of the chapter is organized as follows. Section 5.1 summarizes previous research work in resource allocation algorithms for HCCA and S-APSD. In Section 5.2 we describe and model analytically our proposed solution for *Distributing* in time as much as possible Resource Allocations (DRA). Section 5.3 presents a *Grouping* algorithm that will be used for comparison reasons (GRA). Following that, in Section 5.4 we evaluate the performance of our proposed DRA scheduler as compared to the GRA one in terms of QoS and power consumption for both HCCA and EDCA. Finally Section 5.5 summarizes our findings and concludes this chapter.

5.1 Related work on centralized scheduling

To the best of our knowledge most of the approaches to HCCA scheduling presented in the literature share the basic idea of polling an admitted flow at regular intervals in order to fulfill the delay and bandwidth requirements defined in a Traffic Specification message (TSPEC) when setting up a flow (see Chapter 2 for a detailed description on HCCA). Hereafter we will refer to the time interval used by the Access Point (AP) to poll a station for a flow as the Service Interval (SI) of this flow. The most common approach in the literature to schedule HCCA resource allocations is what we defined as *Grouping* approach which was inspired by the example scheduler described in the 802.11e standard (32). The idea behind this scheduler is to define a basic service interval common to all flows of all stations. This service interval is selected in order to fulfill the most stringent delay requirement among the different flows. Thus, at intervals defined by the basic service interval, the AP starts to sequentially poll all associated flows. The main problem of this approach is that a common service interval for all flows in general can not be guaranteed to match the packet generation rate of each individual flow. In this case bandwidth may be wasted by unnecessary polling of stations. (108) and (109) are relevant examples of proposed implementations of this approach.

In contrast to the *Grouping* solution, another approach called *Earliest Due Date* (EDD), which was first proposed in (110), does not try to avoid collisions between the polling times of different flows but instead uses Earliest Deadline First (EDF) to resolve such collisions. A deadline, related with the service interval and the delay bound, is associated to each flow and is used to decide which flow to poll first in case there is more than one flow pending to be polled at the same point in time. Among these flows the EDD scheduler selects the one with the closest deadline. This solution removes the need to impose a common basic service interval as compared to the grouping approach.

In our previous work (106), we proposed an algorithm that determines the Service Start Time (SST) of each flow in order to distribute in time as uniformly as possible the allocation of resources for the different flows. We refer to this approach as the *Distribution* approach. Like in the EDD scheduler, no requirement needs to be imposed on the service intervals demanded by the different flows. In addition, this scheme minimizes the chances of having more than one HCCA flow pending to be polled at the same point of time. Our *Distribution* approach is related to the concept of *Spectrum Load Smoothing* (SLS) proposed in (115). SLS allows cognitive radios to support QoS guarantees in a distributed way by coordinating their transmissions across a certain *smoothing period*, which relates to the QoS requirements of each station. Notice though, that spreading in time transmissions to/from a station implementing S-APSD increases its power consumption, because the station has to stay awake until its correspondent transmissions complete. For this reason, we opt to distribute resource allocations by shifting their starting times, while trying to complete them as soon as possible once the corresponding station is awake. If stations would tolerate their transmissions to be smoothed over a certain period, our distribution approach and SLS could be applied together to further smooth in time resource allocations. In this chapter though, we have not considered this scenario.

As it can be observed, the *Grouping* and *Distribution* approaches pursue opposite objectives. The *Grouping* approach *reserves* the channel for HCCA only once every basic service interval but potentially for a long time since all HCCA flows need to be polled. Instead, in the *Distribution* approach the number of HCCA allocations might be larger but their *individual* duration is in general shorter. The EDD aproach can not be classified in any of the two groups since the time when each flow makes the

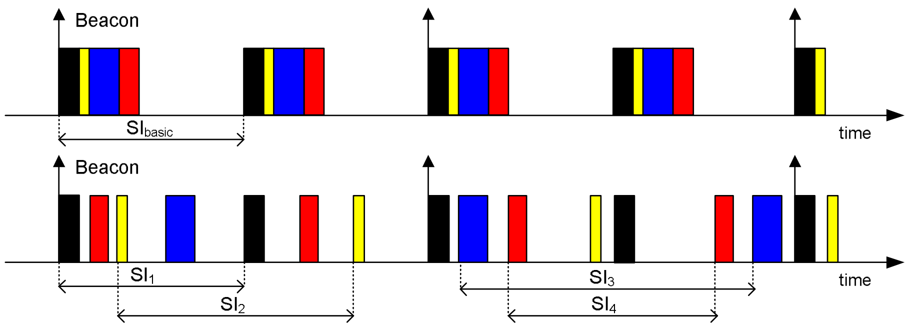


Figure 5.1: Example of HCCA allocations with a *Grouping* (upper graph) and a *Distribution* (lower graph) approach.

initial request together with the service intervals used determines whether HCCA allocations are rather grouped or distributed. Figure 5.1 provides an example of HCCA allocations distribution according to the *Grouping* and *Distribution* approaches.

We argue that distributing, rather than grouping, the allocation of resources in a wireless medium is a more efficient approach in several aspects. The reason is that establishing hard bounds on the transmission time required to accomodate the QoS of a flow in a Wi-Fi network is a difficult task, because variations in the wireless channel may require to increase the original resources allocated to individual flows by means of using a more robust Modulation and Coding Scheme (MCS) or retransmitting packets. In order to better illustrate the innerent variability in the wireless medium, Table 5.1 reports the average and maximum service times required to service every 40ms a sample video flow[1] of average and peak rates 256kbps and 2Mbps under different transmission parameters.

PHY Rate	Average Allocation	Peak Allocation
54Mbps	0.22ms	1.68ms
24Mbps	0.47ms	3.6ms
6Mbps	1.83ms	14.1ms

Table 5.1: Required video allocation time for various PHY conditions.

Observing the potential range of variation in the resources required to accomodate a single flow, providing deterministic guarantees by dimensioning the system under the worst case situation, i.e. peak rate and slowest MCS, may result into extremly limited system capacities. Instead, we claim that variations of the wireless channel can be better accomodated using a distributed approach, where the scheduling

[1]Real video traces obtained from (65) are used.

time of different flows are separated as much as possible and hence the probability of a planned allocation having to wait for the completion of a previous allocation is reduced. The previous fact should be indeed benefitial for both QoS and power consumption.

Based on the previous arguments we focus our work in this chapter in the design and evaluation of an enhancement of our *Distribution* algorithm proposed in (106) which significantly reduces its computational load and in the evaluation of the performance improvements to be expected as compared to a *Grouping* scheduler.

5.2 Distributed Resource Allocation (DRA)

The problem of scheduling a set of periodic flows, or tasks, has been thoroughly researched in the literature, specially in the field of real time operating systems. In this chapter we propose a generic *Distributed Resource Allocation (DRA)* scheduling algorithm that maximizes the minimum distance between the serving times of different flows. The DRA algorithm is based on our previous proposal presented in (106) which required the exploration of the least common multiple (LCM) of the different flows' service intervals and enhances it by: i) considering the serving time of each flow in the scheduling decision and ii) achieving a pseudo-polynomial complexity by removing the need of exploring the LCM of the flows' service intervals.

5.2.1 Defining a way to distribute resource allocations

As discussed in the previous section our goal can intuitively be stated as defining an algorithm that separates as much as possible consecutive resource allocations in the channel. In this section we show that our desired *separation* or *distribution* can be achieved by means of an algorithm that maximizes the minimum distance between allocated flows.

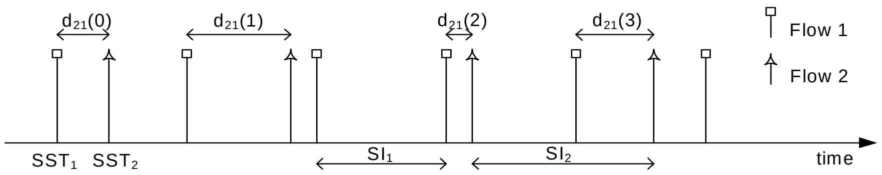


Figure 5.2: Distances between resource allocations.

To start off, a better understanding on the concept of distance between periodic flows is needed. Figure 5.2 depicts the occurences in the channel of two periodic flows with periods, or service intervals in HCCA terminology, SI_1 and SI_2, which have starting times SST_1 and SST_2. Figure 5.2 also depicts the distance between consecutive allocations of the two flows $d_{2,1}(k)$. Indeed, one can express the distance between an occurrence of flow 2 and the previous occurrence of flow 1 as:

$$d_{2,1}(k) = (d_{2,1}(0) + kSI_2) \bmod SI_1 \tag{5.1}$$

Where $d_{2,1}(0)$ is a reference distance between two resource allocations, $d_{2,1}(0) = SST_2 - SST_1$ in Figure 5.2.

Later in this section, it will be formally shown that the previous sequence, $d_{2,1}(k)$, is indeed periodic and that its elements can be expressed as $d_{2,1}(j) = d_{min_{2,1}} + j \cdot gcd(SI_1, SI_2)\ j \geq 0$, where $d_{min_{2,1}} = d_{2,1}(0) \bmod gcd(SI_1, SI_2)$, and gcd stands for greatest common divisor. Thus, an objective way of increasing the separation between the service times of flows 1 and 2 is to increase $d_{min_{2,1}}$ and/or $gcd(SI_1, SI_2)$.

Maximizing $gcd(SI_1, SI_2)$ requires modifying the service intervals used to poll each flow, and is intuitively achieved by setting both service intervals as similar as possible, for instance making them all multiple of a certain value SI_{basic}, being this value as high as possible. However, manipulating the service intervals assigned to each flow in order to maximize the separation between consecutive allocations in the channel can have counter effects. It may turn into wasted bandwidth if the polling interval does not efficiently match the application generation rate, i.e. in the case of voice or video codecs, or it may result in increased delays if adjusting to a multiple of SI_{basic} results in a polling interval above the desired delay bound. Therefore, in this chapter we consider that the service interval used to poll each flow will be determined based on a set of different requirements (QoS, bandwidth efficiency, power consumption,etc), and given the selected service intervals we attempt to separate the allocation of flows as much as possible.

If we consider the service intervals SI_1 and SI_2 as given, the only way to increase the separation between any two allocations of flows 1 and 2 in the previous example is to maximize the minimum distance, $d_{min_{2,1}}$. Looking at the definition of $d_{min_{2,1}}$ we can see that although $d_{min_{2,1}}$ is bounded by $gcd(SI_1, SI_2)$ it can be maximized by properly setting $d_{2,1}(0)$ which is defined by the starting time of each flow, SST_i. Note that SST_i can be determined by the scheduler and conveyed to the flows in HCCA through the Schedule Element (33).

The previous example is too simplistic in the sense that we consider only two flows in the channel. Consider now the case that N periodic flows are already scheduled and a new flow requests entry to the system that has therefore to be granted a certain starting time SST_i. Clearly, in a general setting it is not possible to select a SST_i for the new flow that maximizes the distances between the polling times of the new flow and all of the already scheduled flows at the same time. In this case the distances between the new flow and each of the already scheduled flows could be weighted to decide on which distances are more important to maximize. However a compelling argument from Real Time Scheduling Theory (RTST) drives us towards again trying to maximize the minimum of those distances. The problem of scheduling task sets in RTST can be stated in very similar terms to the problem of scheduling flows in HCCA. It is known then from the feasibility tests devised in RTST that if a deadline is associated to each flow and Earliest Deadline First (EDF)[1], which has been proven optimal under many RTST settings, is used as scheduling discipline, then the *critical instant* (112) occurs when all flows try to access the shared resource at the same time. The critical instant is defined as the release time of a flow for which the response time, and hence the probability of a deadline violation, is maximized. Maximizing minimum distances is therefore an intuitive way of moving away from this worst-case situation trying hence to maximize the number of admitted flows if an admission control for HCCA would be defined similar to the feasibility tests employed in RTST.

Next, we present our DRA algorithm that allocates a start time for a new flow requesting access to the system such that the minimum distance between the service periods of this new flow and the service periods of the already scheduled flows is maximized. Although defined in the context of HCCA, the presented DRA algorithm can be used in any context where scheduling of periodic flows is required.

5.2.2 The two flows case

To facilitate the understanding of the analysis of our proposal let us first consider a system consisting in two flows. The first flow has already been scheduled and is periodically served according to its Service Start Time (SST) SST_1, Service Interval SI_1 and serving time $TXOP_1$[2]. A second flow requests at a certain point of time, which is considered to be $t = 0$, to be scheduled in the system with service interval SI_2 and serving time $TXOP_2$. The goal of our algorithm is to find the service start time for the requesting flow, SST_2, that maximizes the minimum distance between the serving times of flows 1 and 2. In the rest of the chapter we refer to the regular instants where a flow is scheduled to be served as the *release* instants of this flow. Figure 5.3 represents flows 1 and 2 and $next\_rel\_time(1)$ is a variable defined starting from $t = 0$ that contains the next release time of flow 1.

Let us define the *left* distances between flow 2 and flow 1, $d_{l_{2,1}}(k)$ where $k \in \mathbb{N}$, as the time differences between the $k-th$ release time of flow 2 and the last release time of flow 1. Similarly, the *right* distances, $d_{r_{2,1}}(k)$, are defined as the time differences between a release time of flow 2 and the next release time of flow 1. Additionally let us define the *left* and *right effective* distances, $d_{l\_eff_{2,1}}(k)$ and

[1]Formally, preemptive EDF has to be considered although maximizing the minimum distance is also a common-sense approach in the case of non preemptive EDF (113).

[2]*TXOP* stands for Transmission Opportunity and is the transmission time granted to a polled flow in HCCA.

SI_i	Service interval used for flow i
$TXOP_i$	Transmission Opportunity used for flow i
$d_{l_{i,j}}(k)$	Sequence of left distances of flows i and j
$d_{r_{i,j}}(k)$	Sequence of right distances of flows i and j
$d_{l\_eff_{i,j}}(k)$	$d_{l_{i,j}}(k) - TXOP_j$
$d_{r\_eff_{i,j}}(k)$	$d_{r_{i,j}}(k) - TXOP_i$
$d_{l\_min_{i,j}}$	Minimum left distance of flows i and j
$d_{r\_min_{i,j}}$	Minimum right distance of flows i and j
$d_{l\_eff\_min_{i,j}}$	$d_{l\_min_{i,j}} - TXOP_j$
$d_{r\_eff\_min_{i,j}}$	$d_{r\_min_{i,j}} - TXOP_i$
$\phi_{i,j}$	Reference distance between flows i and j

Table 5.2: Variables used in the analysis.

$d_{r\_eff_{2,1}}(k)$, as the difference between the end of the serving time of one flow and the release time of the other flow. All these distances are depicted in Figure 5.3. In addition, Table 5.2 contains a summary of the main variables used throughout this section.

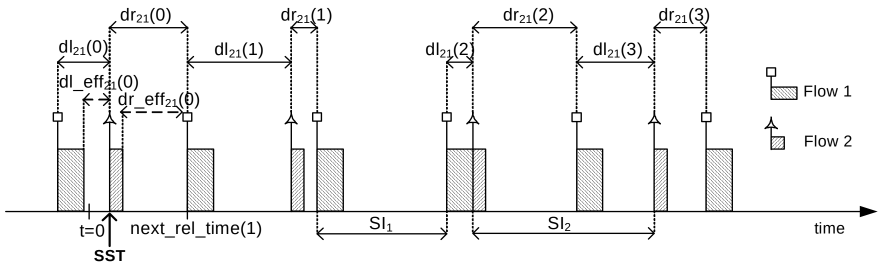


Figure 5.3: Distances between release times.

From our definitions it is immediate to derive that $d_{l\_eff_{2,1}}(k) = d_{l_{2,1}}(k) - TXOP_1$ and $d_{r\_eff_{2,1}}(k) = d_{r_{2,1}}(k) - TXOP_2$. Therefore, in order to maximize the minimum effective distance between flows 1 and 2, we can focus our analysis in maximizing the minimum *left* and *right* distances.

Our goal is thus to find the SST of flow 2 that maximizes the minimum distance between the release times of flows 1 and 2. Notice for this purpose that $d_{l_{2,1}}(k)$ and $d_{r_{2,1}}(k)$ can be expressed as:

$$\begin{aligned} d_{l_{2,1}}(k) &= (d_{l_{2,1}}(0) + kSI_2) \bmod SI_1 \\ d_{r_{2,1}}(k) &= SI_1 - d_{l_{2,1}}(k) \end{aligned} \tag{5.2}$$

Where $d_{l_{2,1}}(0)$ is the initial *left* distance taken as reference.

Then, the first step towards our goal is to express the minimum value of $d_{l_{2,1}}(k)$ and $d_{r_{2,1}}(k)$, which we want to maximize, as a function of the SST to be assigned to flow 2. For that purpose notice that

$d_{l_{2,1}}(k)$ can be expressed as:

$$d_{l_{2,1}}(k) = d_{l_{2,1}}(0) + kSI_2 - mSI_1 \tag{5.3}$$

Where $k, m \in \mathbb{Z}$. If we now separate the right hand size of the previous equation into terms that can be divided by $d = gcd(SI_1, SI_2)$, where gcd stands for greatest common divisor, we obtain:

$$d_{l_{2,1}}(k) = d_{l_{2,1}}(0) \bmod d + pd + kSI_2 - mSI_1 \tag{5.4}$$

$$d_{l_{2,1}}(k) - d_{l_{2,1}}(0) \bmod d = jd \tag{5.5}$$

Where $k, m, p, j \in \mathbb{Z}$. Thus, considering that in Equation 5.2, $d_{l_{2,1}}(k) \leq SI_1$:

$$d_{l_{2,1}}(k) = d_{l_{2,1}}(0) \mod d + jd,\ 0 \leq j < N = \frac{SI_1}{d} \tag{5.6}$$

Therefore, the minimum value of $d_{l_{2,1}}(k)$ is $d_{l\_min_{2,1}} = d_{l_{2,1}}(0) \bmod gcd(SI_1, SI_2)$. In addition, $d_{r\_min_{2,1}}$ can be obtained in the following way. Notice that $d_{r\_min_{2,1}} = SI_1 - d_{l\_max_{2,1}}$, thus if there are only $N = \frac{SI_1}{d}$ different values for $d_{l_{2,1}}(k)$, then $d_{l\_max_{2,1}} = SI_1 - (gcd(SI_1, SI_2) - d_{l\_min_{2,1}})$, and $d_{r\_min_{2,1}} = gcd(SI_1, SI_2) - d_{l\_min_{2,1}}$.

A simple transformation can now be used to express the minimum *left* and *right* distances as a function of the SST to be assigned to flow 2. If this SST equals $next\_rel\_time(1)$ the minimum *left* distance will be zero, hence:

$$d_{l\_min_{2,1}} = (SST - \phi_{2,1}) \bmod gcd(SI_1, SI_2) \tag{5.7}$$

Where $\phi_{2,1}$ is an initial shift defined as $\phi_{2,1} = next\_rel\_time(1) \bmod gcd(SI_1, SI_2)$.

Having found the expression of the minimum value for the *left* and *right* distances it is immediate to obtain an expression for the minimum *left* and *right effective* distances since $d_{l\_eff\_min_{2,1}} = d_{l\_min_{2,1}} - TXOP_1$ and $d_{r\_eff\_min_{2,1}} = d_{r\_min_{2,1}} - TXOP_2$.

Figure 5.4 illustrates the minimum values of $d_{l\_eff_{2,1}}(k)$ and $d_{r\_eff_{2,1}}(k)$ as a function of the selected SST for given values of $TXOP_1$ and $TXOP_2$ ($TXOP_1 > TXOP_2$ in the figure). The minimum *left* and *right* effective distances achieve maximum values of $gcd(SI_1, SI_2) - TXOP_1$ and $gcd(SI_1, SI_2) - TXOP_2$ respectively, but these maximum values do not occur at the same time.

Considering again Figure 5.4, the overall minimum distance, i.e., $\min\{d_{l\_eff\_min_{2,1}}, d_{r\_eff\_min_{2,1}}\}$, is a periodic set of triangular shapes of unitary slope. This minimum distance function has period $T = gcd(SI_1, SI_2)$ and presents discontinuities at points $\phi_{2,1} + k \cdot gcd(SI_1, SI_2)$, $k \in \mathbb{N}$, which we refer hereafter as *critical points*. Thus, the values of SST that maximize the minimum distances between any two release times of flow 1 and flow 2 are:

$$SST_{OPT} = \phi_{2,1} + k \cdot \frac{gcd(SI_1, SI_2)}{2} + \frac{TXOP_1 - TXOP_2}{2}$$

Where k is an odd integer, $k = \{..., -1, 1, 3, 5, 7, ...\}$.

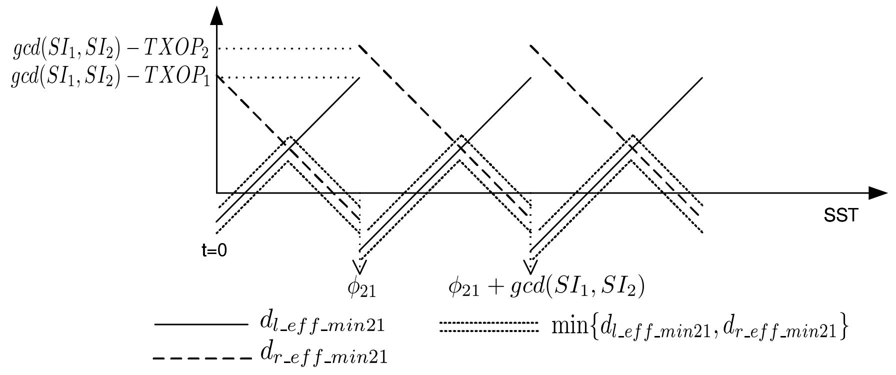

$gcd(SI_1, SI_2) - TXOP_2$
$gcd(SI_1, SI_2) - TXOP_1$
SST
t=0
ϕ_{21}
$\phi_{21} + gcd(SI_1, SI_2)$
$d_{l\_eff\_min21}$
$d_{r\_eff\_min21}$
$\min\{d_{l\_eff\_min21}, d_{r\_eff\_min21}\}$

Figure 5.4: Minimum *left* and *right* effective distances.

5.2.3 The N flows case

Based on the 2 flows case analysis, we extend our results now to the N flows case which consists in N periodic flows already scheduled in a system with service intervals SI_i and serving times $TXOP_i$, where $i = 1...N$, and a new flow arriving at time $t = 0$ which requires to be scheduled with service interval SI_{N+1} and serving time $TXOP_{N+1}$. As in the previous case, our goal is to find the initial release time for the new flow, SST_{N+1}, that maximizes the minimum effective distance between this flow and the already scheduled flows.

Defining as before the *left* and *right* distances of the new flow, $N+1$, with each of the already scheduled flows, i where $i = 1...N$, we can divide the N-flows problem into N different 2-flow problems:

$$\begin{aligned} d_{l\_min_{N+1,i}} &= (SST - \phi_{N+1,i}) \bmod gcd(SI_i, SI_{N+1}) \\ d_{r\_min_{N+1,i}} &= gcd(SI_i, SI_{N+1}) - d_{l\_min_{N+1,i}} \end{aligned}$$

With $\phi_{N+1,i} = next\_rel\_time(i) \bmod gcd(SI_i, SI_{N+1})$.

Similarly, the *left* and *right* effective minimum distances can be defined as $d_{l\_eff\_min_{N+1,i}} = d_{l\_min_{N+1,i}} - TXOP_i$ and $d_{r\_eff\_min_{N+1,i}} = d_{r\_min_{N+1,i}} - TXOP_{N+1}$. Figure 5.5 represents the minimum *left* and *right* effective distances in a system where two flows were already scheduled and a new flow has requested access. In the figure $d_{l\_eff\_min_{N+1,i}}$ and $d_{r\_eff\_min_{N+1,i}}$ are not depicted separately but instead we directly plot $\min\{d_{l\_eff\_min_{N+1,i}}, d_{r\_eff\_min_{N+1,i}}\}$ for each scheduled flow. Notice in the figure that $TXOP_1 > TXOP_3 = TXOP_2$.

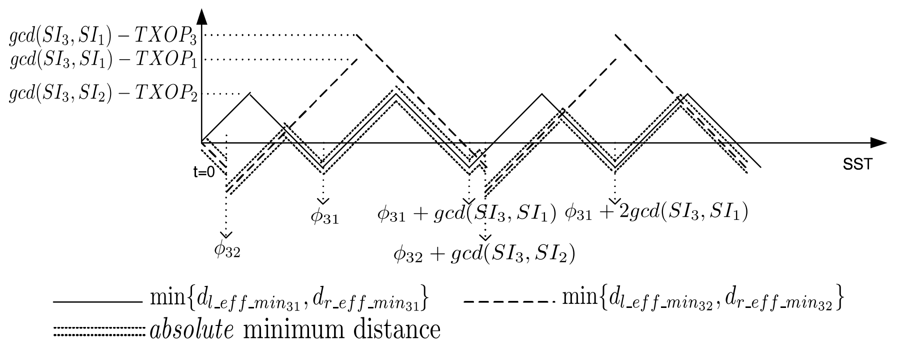

$gcd(SI_3, SI_1) - TXOP_3$
$gcd(SI_3, SI_1) - TXOP_1$
$gcd(SI_3, SI_2) - TXOP_2$
t=0
SST
ϕ_{31}
$\phi_{31} + gcd(SI_3, SI_1)$
$\phi_{31} + 2gcd(SI_3, SI_1)$
ϕ_{32}
$\phi_{32} + gcd(SI_3, SI_2)$
$\min\{d_{l\_eff\_min_{31}}, d_{r\_eff\_min_{31}}\}$
$\min\{d_{l\_eff\_min_{32}}, d_{r\_eff\_min_{32}}\}$
absolute minimum distance

Figure 5.5: Minimum *left* and *right* distances.

Notice that in case of having N flows, our goal is to find SST_{N+1} that maximizes $\min\{d_{l\_eff\_min_{N+1,i}}, d_{r\_eff\_min_{N+1,i}}\}$, which we refer to as the *absolute* minimum distance.

Thus, realizing that the absolute minimum distance is also a periodic set of triangular shapes of unitary slope, a simple algorithm that finds the SST_{N+1} that maximizes this distance can be implemented in the following way:

1. Compute $gcd(SI_i, SI_{N+1})$ for all the N already scheduled flows, $0 < i \leq N$.

2. Compute the period of the absolute minimum distance:

$$T' = lcm(gcd(SI_i, SI_{N+1}), ..., gcd(SI_N, SI_{N+1}))$$

3. For each already scheduled flow generate all critical points, $\phi_{N+1,i} + k \cdot gcd(SI_i, SI_{N+1})$, contained in T'.

4. Define a sorted list L containing all critical points.

5. Define a function F that operating on list L obtains the SST that maximizes the minimum effective distance.

In order to find the optimum SST, the function F needs to go through all the critical points contained in L and obtain the maximum of the triangular shape between each two consecutive points. Hence, the function F needs to examine in each critical point the value of $\min\{d_{l\_eff\_min_{N+1,i}}, d_{r\_eff\_min_{N+1,i}}\}$ for all N already scheduled flows and find out the initial value of the triangular shape. Our implementation of function F has a worst-case time complexity bounded by $O(M(N+1))$, where M is the

Algorithm 5.1: Distributed Resource Allocation Algorithm (DRA).

```
– Variables definition  $SI_i \leftarrow$ Service interval of flow $i$ (obtained from the TSPEC).
$next\_rel\_time(i) \leftarrow$ Next release time of flow $i$.
$list\_critical\_points \leftarrow$ List that contains the critical points of the absolute minimum distance function.
– Routine for scheduling a new flow  for $i = 1$ to $N$ do
    $partial\_gcd(i) \leftarrow gcd(SI_{N+1}, SI_i)$
    $\phi_{N+1,i} \leftarrow next\_rel\_time(i) \bmod gcd(SI_{N+1}, SI_i)$
$T' \leftarrow lcm(partial\_gcd(1), ..., partial\_gcd(N))$
$\phi_{min} = min\{\phi_{N+1,1}, ..., \phi_{N+1,N}\}$
$j \leftarrow 0$
for $i = 1$ to $N$ do
    $k \leftarrow 0$
    while $\phi_{N+1,i} + k \cdot partial\_gcd(i) \leq \phi_{min} + T'$ do
        $list\_critical\_points(j) \leftarrow \phi_{N+1,i} + k \cdot partial\_gcd(i)$
        $k \leftarrow k + 1, j \leftarrow j + 1$
$L \leftarrow$ Quicksort $list\_critical\_points$ in increasing order
$SST_{OPT} \leftarrow F(L)$.
```

maximum size of list L and N is the number of already scheduled flows. For the sake of space and clarity of the explanation our implementation of function F is provided in the Appendix at the end of this chapter.

A pseudo-code implementation of our algorithm to distribute resource allocations is shown in Algorithm 5.1. In the rest of the chapter we refer to this algorithm as the *DRA* algorithm.

It is interesting to discuss how the DRA algorithm behaves in an overloaded situation, since as the number of flows increases, or flows with high $TXOP$s are considered, overlapping between flows may be unavoidable. This situation though does not impose any constraint on the behavior of DRA. If overlapping occurs the effective minimum distance will become negative, as illustrated in Figure 5.5, and DRA will simply select the starting service time that results in the minimum amount of overlapping. However, in order to limit the amount of overlapping such that no harm is done on the QoS requirements of a specific flow, an admission control module complementing DRA would be needed.

Finally, to illustrate the distribution achieved under a max-min distance criteria Figure 5.6 depicts two example allocations. The upper part of the Figure shows the resulting distribution when scheduling three flows with a period of 40ms, where flow i enters the system before flow j if $i < j$. The lower part of the Figure shows the resulting allocation when scheduling, again in the specified order, three flows of periods 40ms, 60ms and 80ms. It is worth noticing that in a general setting when each flow has a different period the resulting distribution does depend on the order of arrival of the flows. Indeed, it is interesting to notice that under HCCA it is possible for the AP to reschedule the allocation times of previously scheduled flows when a new flow wants to enter the system by sending a new Schedule

Element to each existent flow in the system, hence paying a price in increased signaling and complexity. In this chapter though we do not explore the problem of finding, given a set of N flows to be scheduled, the scheduling order that results in the maximum separation under DRA.

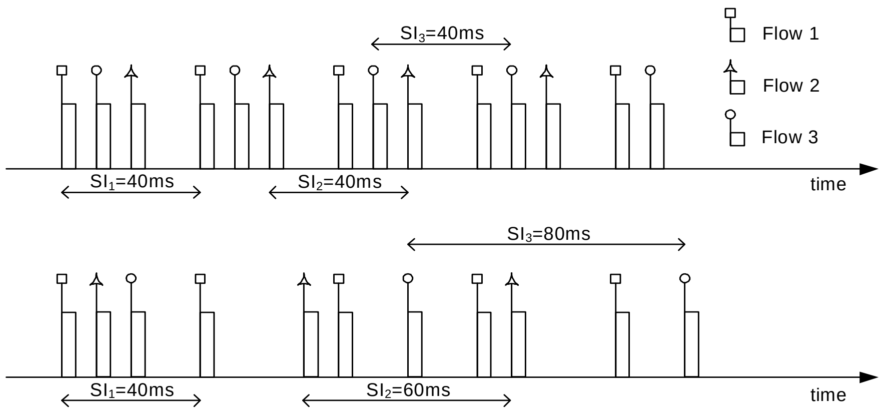


Figure 5.6: Resulting distributions with a max-min criteria.

5.2.4 Complexity of DRA

In this section we analyze the complexity of our proposed DRA algorithm in order to provide an upper bound on the computational load to be expected when implementing it in practice.

Looking at Algorithm 5.1 the worst case time complexity of DRA can be analysed in the following way. When a new flow has to be scheduled, DRA first computes $gcd(SI_i, SI_{N+1})$, $\forall i = 1...N$, having a complexity of $O(N \log SI_{max})$, where $SI_{max} = \max_i SI_i$ and N is the number of flows already scheduled in the system. After that, computing T' requires again a complexity of $O(N \log SI_{max})$ and computing ϕ_{min} requires a complexity of $O(N)$. After these initial operations the algorithm constructs the list containing the critical points of the absolute minimum distance function, $list\_critical\_points$, which takes complexity $O(M)$, being M the number of elements in this list. Once $list\_critical\_points$ is populated, it can be sorted with a worst case[1] complexity of $O(M \log M)$. Finally, the function F computes for each critical point the value of the absolute minimum distance[2], taking a worst case complexity of $O(M(N+1))$, as illustrated in the Appendix at the end of this chapter.

Therefore, a key parameter to evaluate the complexity of the DRA algorithm is the size M of $list\_critical\_points$. In order to obtain an upper bound on M an upper bound on the period of the

[1]Notice that in practice we use QuickSort which has a worse worst-case performance, $O(M^2)$, but provides the best performance in average.

[2]Computing $d_{eff\_min_{i,j}}$ takes linear complexity since it only involves a modulo operation.

absolute minimum distance function (T') is needed:

$$T' = lcm(gcd(SI_i, SI_{N+1}), ..., gcd(SI_N, SI_{N+1})) \leq SI_{N+1}$$

Hence, the maximum size of $list\_critical\_points$ can be upper bounded as:

$$M = \sum_{i=1}^{N} \frac{T'}{gcd(SI_i, SI_{N+1})} \leq SI_{N+1} \sum_{i=1}^{N} \frac{1}{gcd(SI_i, SI_{N+1})} \leq SI_{N+1}N$$

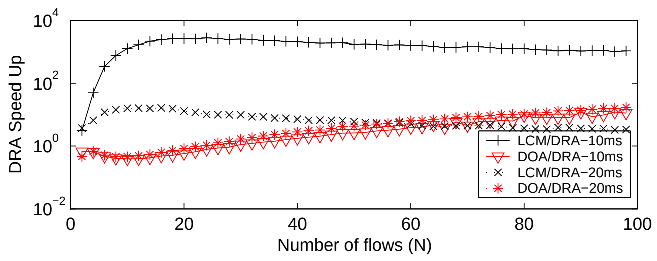
Based on the previous analysis we can express the overall worst case time complexity of DRA as bounded by $O(SI_{N+1}N(N + \log SI_{N+1}N) + 2N \log SI_{max})$. Notice that formally DRA has a pseudo-polynomial complexity because SI_{N+1} and SI_{max} are not necessarily polynomial in the size of the problem description. In a practical setting though, the typical service intervals used in wireless networks result in a reduced complexity of DRA, as it will be shown next.

Notice that the algorithm proposed in our early work in (106), which is based on the exploration of the aggregated period of the different flows, has a worst case time complexity bounded by $O(lcm(SI_i, ..., SI_{N+1})\frac{N}{precision})$ where lcm stands for *least common multiple* and $0 \leq precision \leq SI_{N+1}$ is a parameter used to explore the space of possible starting times. Instead, DRA breaks the dependency on the lcm of the different flows' service intervals. An alternative approach to DRA is the *Dissimilar Offset Assignment* (DOA) algorithm proposed in (116) in the context of Real Time Operating Systems, which, like DRA, assigns offsets to different flows in order to maximize their relative distance. This algorithm also breaks the dependency on the lcm of the flows' service intervals achieving a worst time complexity of $O(N^2(\log SI_{max} + \log N^2))$. Next, we experimentally evaluate the performance of these algorithms against the one of DRA both in terms of time complexity and achieved flow separation.

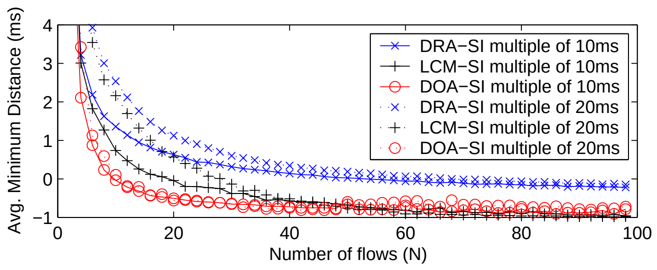
In our evaluation we consider that each flow has a service interval randomly selected between 10ms and 100ms, and a serving time randomly selected between 0ms and 1ms[1]. In addition, we consider a first setting where all flows have a service interval multiple of 10ms and a second one where all flows have a service interval multiple of 20ms. Figure 5.7 depicts the results of our evaluation. More than 500 independent scheduling instances were used to obtain the presented average values, however we have not depicted the 95% confidence intervals in the graphs because they were too small to add significant information.

Figure 5.7(a) illustrates the ratio between the computational time required by the algorithms under study and DRA, in order to schedule a new flow when N flows are already present in the system. It is clear in the figure that, as predicted by the analysis, the complexity of an LCM based approach explodes

[1]This service time is arbitrarily chosen for the purpose of the experiment. Notice that the values of the service time of the different flows have no impact on the complexity of the algorithm.



(a) Relative computational complexity.



(b) Average Minimum Distance.

Figure 5.7: Performance of DRA.

when the flows' service intervals are multiple of 10ms. Instead, DRA and DOA exhibit a stable and similar complexity regardless of the flows' service intervals and the number of scheduled flows. Finally, Figure 5.7(b) illustrates the average minimum left distance experienced with the three algorithms under study. As observed in the figure the ability to separate flows increases for all algorithms when all flows are multiple of 20ms, instead of 10ms, and DRA shows to always achieve the highest flow separation. Based on these computational load and scalability results we conclude that in general the DRA algorithm would be a feasible solution in practice. In addition, given that DRA is the algorithm providing the best trade-off between flow separation and time complexity, in the rest of this chapter we consider DRA as the algorithm representative of the *Distribution* approach.

5.3 Grouping of Resource Allocations (GRA)

In order to compare our proposed *Distributed Resource Allocation* algorithm (DRA) with a *Grouping* one, we designed an enhanced *Grouping of Resource Allocations* algorithm (GRA) suited for stations with strict power consumption requirements operating in S-APSD. The presented scheduler is based on the common idea of defining a basic service interval, SI_{basic}, and scheduling the HCCA transmissions

Algorithm 5.2: Grouping Resource Allocations Algorithm (GRA).

```
1  – Variables definition $SI_{basic} \leftarrow$ Basic service interval.
2  $next\_SST \leftarrow$ Variable indicating the next available slot.
3  $D, r, S \leftarrow$ Delay bound, Mean Data Rate and Average MSDU size specified in the TSPEC.
4  $Tx\_time \leftarrow$ Transmission time of an MSDU of size $S$ plus overhead.
5  – Routine for scheduling a new flow $SI \leftarrow \lfloor \frac{D}{SI_{basic}} \rfloor SI_{basic}$
6  if $next\_SST$ not initialized then
7      $next\_SST \leftarrow current\_time$
8  $SST \leftarrow next\_SST + \lceil \frac{current\_time - next\_SST}{SI} \rceil SI$
9  $next\_SST \leftarrow next\_SST + \lceil \frac{SI \times r}{S} \rceil Tx\_time$
```

consecutively, see Figure 5.1 for an example. However, we introduce two differences with respect to the reference scheduler described in the 802.11e standard (32): i) we do not force all flows to be polled every basic service interval, but allow flows to be polled at multiples of the basic service interval, and ii) we do not allow any flow to be polled prior to its nominal polling time. This last constraint is appropriate in the case of S-APSD where a station awakes to be polled at its nominal polling interval.

A pseudo-code implementation of our proposed grouping scheduler is shown in Algorithm 5.2. This algorithm is run in the AP every time a new HCCA flow is admitted and returns the Service Start Time (SST) and the Service Interval (SI) for the new flow.

The algorithm starts by determining the service interval that will be used to poll the new flow. For this purpose we make use of the delay bound, D, specified by the station in the TSPEC message. Specifically, the flow will be polled at the maximum service interval below D that is a multiple of the basic service interval SI_{basic}. Once SI has been determined, the algorithm computes the SST for the new flow. A reference variable, $next\_SST$, contains the next available polling slot. Note that $next\_SST$ does not refer to the current time but to a reference time that is determined by the first flow admitted in the system. The actual SST granted to the requesting station corresponds to the next nominal polling time given $next\_SST$ and SI. Finally, the $next\_SST$ variable is updated for the next requesting flow.

Our proposed grouping scheduler allocates for the current flow a transmission slot computed based on the expected average TXOP, which depends on the mean data rate, nominal MSDU size, assigned polling interval and experienced radio conditions. Different allocation polices, like a peak rate police, could also be used to compute the transmission slot, although a price would be paid in the maximum number of admitted flows. The admission control modules for both DRA and GRA schedulers are left out of the scope of this chapter.

5.4 Performance evaluation

In this section we compare the performance of the *Distributed* (DRA) and *Grouping* (GRA) schedulers presented in this chapter. The goals of this study are: i) Evaluate whether the additional complexity required by DRA is justified by relevant performance improvements and ii) Gain a deep insight on the reasons for the performance differences due to the DRA and GRA schedulers both on HCCA and EDCA stations. We consider specially relevant a scenario where both centralized and contention based schemes coexist in order to understand how HCCA, if eventually deployed, can affect the performance of the large number of devices already in the market implementing EDCA.

The analysis has been performed via simulations. We extended the 802.11 libraries provided by OPNET (64) to include the power saving extensions defined in 802.11 and 802.11e, EDCA, HCCA and our designed DRA and GRA scheduling algorithms.

We consider an infrastructure Wi-Fi where all the stations use the 802.11g physical layer transmitting at 54Mbps and the Access Point (AP) generates Beacon frames every 100ms.

Four different kinds of stations are considered in our evaluation that constitute a basic *Cluster* for our experiments. The traffic transmitted by each station in our cluster, together with the QoS and power saving protocols are described in Table 5.3.

	Description	**Medium Access**	**Power Saving**
Voice	Bidirectional Voice calls using G.711 (64Kbps) with a packetization interval of 20ms.	HCCA	S-APSD
Video	Bidirectional Video conference calls emulated by means of a VBR stream (65) at 30fps with an average rate of 1Mbps and a peak rate of 9Mbps.	HCCA	S-APSD
Voice	Bidirectional Voice calls using G.711 (64Kbps) with a packetization interval of 20ms.	EDCA (AC_VO)	U-APSD
FTP	App: FTP downloads of a 50MB file. We consider TCP New Reno and a RTT of 20ms between the AP and the server storing the file. The AP has a Power Save buffer with a size of 100 packets. The TCP advertised window is configured such that it does not limit the growth of the TCP congestion window.	EDCA (AC_BE)	U-APSD

Table 5.3: Applications Description.

U-APSD is configured differently for Voice and FTP stations. Voice stations running U-APSD generate trigger frames when an uplink data frame is available, or generate a QoS Null frame after 20ms without sending an uplink frame in order to retrieve the downlink data buffered in the AP. FTP U-APSD stations generate a trigger frame upon receiving a Beacon frame indicating that there are downlink frames buffered for them. The interested reader is referred to (47) or to the discussion in Chapter 3 for more details on this U-APSD implementation.

In order to decide the TXOP to be granted to a flow in HCCA every time it is polled we use the description of each flow included in the TSPEC message, a similar method is used in (114).

The following EDCA settings have been used at the stations and AP in our evaluation:[1]

EDCA	**AIFS**	**CWmin**	**CWmax**	**TXOP length**
AC_VO	2(1)	3	7	1.5 ms
AC_BE	3	15	1023(63)	0 ms

Table 5.4: EDCA configuration for the different ACs.

Regarding power consumption, we use our regular power model based on the chipset data disclosed in (66). The power consumption values used in our model are shown in Table 5.5[2].

Cisco Aironet^{TM}	**Sleep**	**Listen**	**Rx**	**Tx**
Current (mA)	15	203	327	539

Table 5.5: Current consumption levels of a popular PCMCIA card.

Based on this simulation setup, in order to compare the performance of the GRA and DRA schedulers we performed an experiment that consisted in increasing the number of clusters until a point where the Wi-Fi network can not meet the QoS demands of the Voice and Video stations. In addition, we study the effect of varying the service interval used to serve HCCA traffic by defining two different configurations. In the first configuration, referred as <20,40> configuration, HCCA Voice stations are polled every 20ms and HCCA Video stations every 40ms. In the second configuration, referred as <40,80> configuration, HCCA Voice stations are polled every 40ms and HCCA Video stations every 80ms. Each point in the following graphs has been obtained considering at least 15 independent simulation runs of 300 seconds. Average values are plotted with the correspondent confidence intervals at 95%, and cumu-

[1]Parenthesis indicate that a different value is used at the AP.

[2]For the sleep mode we used the value of a previous model of a Cisco PCMCIA card (Cisco Aironet 350) since no information was available for the referenced one.

lative distribution functions are obtained considering together the values obtained across all simulation runs.

In the following, we present our major findings based on these experiments which are divided for clarity reasons in three subsections: i) Performance of HCCA stations, ii) Performance of EDCA Voice stations and iii) Performance of EDCA FTP stations.

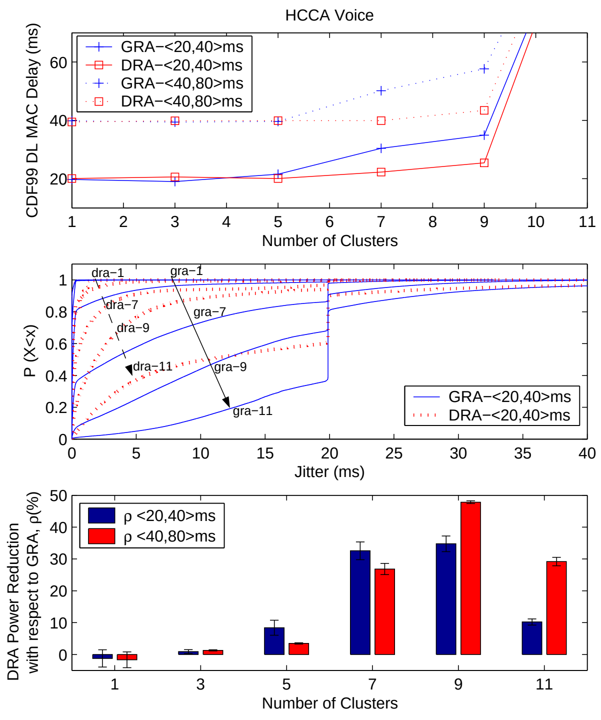
5.4.1 Performance of HCCA stations

The upper part of Figure 5.8(a) shows the delay value at 99% of the cumulative distribution function (cdf99) experienced by the Voice connections over HCCA for the two configurations under study: <20,40> and <40,80>. Due to space reasons only downlink (AP $\rightarrow$ STA) results are shown although similar results were observed in uplink (notice that there is no contention in the uplink for HCCA). We can see in the figure how when congestion increases (>5 clusters) DRA can still maintain a delay close to the configured deadline, while GRA results in deadline violations of around 10ms. The reason for the improved behavior of DRA is that the probability of a Voice connection being delayed by a big Video connection (VBR) is lower since DRA separates HCCA flows as much as possible.

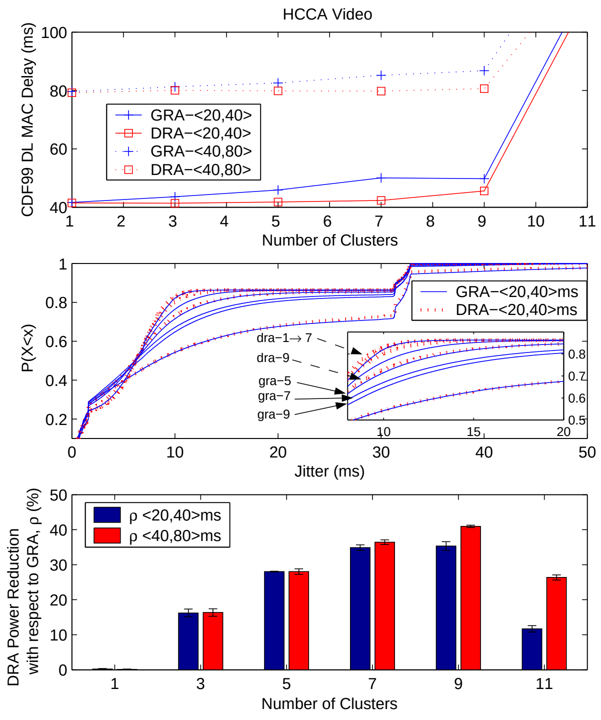
The middle part of Figure 5.8(a) illustrates the cdf of the instantaneous packet delay variation or jitter[1] experienced by the HCCA Voice connections, when the DRA and GRA schedulers are used. For the sake of clarity we only include the results for the <20,40> configuration; the results for the <40,80> configuration did not significantly differ from the presented ones. In addition, we include a cdf curve for each simulated scenario (number of clusters from 1 to 11), and we add a label next to each curve to identify the simulated scenario. Notice that since in the <20,40> configuration the polling rate of a Voice flow equals the flow's packet generation rate, the theoretical jitter should be zero. However, when the number of flows increases, and due to the variability in the service times of Video flows, the jitter experienced by a Voice flow increases substantially. It is remarkable though that due to a higher separation between service times, the jitter of a Voice stream under DRA is kept lower than under GRA. In addition, the presented cdf curves show a step at 20ms which corresponds to the Voice packet generation interval. This step appears when due to an excessive delay several Voice packets are downloaded in a single service period and is significantly higher in the case of GRA.

The lower part of Figure 5.8(a) plots the relation between the average power consumed by the Voice stations running S-APSD under the DRA and GRA schedulers. This ratio is computed as $\rho = \frac{Power_{GRA} - Power_{DRA}}{Power_{GRA}}$ and can be understood as the average power reduction obtained with DRA. The results show that the higher separation between flows obtained with DRA results in up to a 35-45%

[1] $jitter(k) = |Delay(k+1) - Delay(k)|$



(a) HCCA Voice Performance.



(b) HCCA Videoconference Performance.

Figure 5.8: Performance of HCCA stations.

power reduction for each configuration. The reason is that by separating the different flows we minimize the waiting time of a station since the moment it awakes until it is served. With a detailed look at the graph we can also see that the difference between GRA and DRA is maximum just before the network saturates.

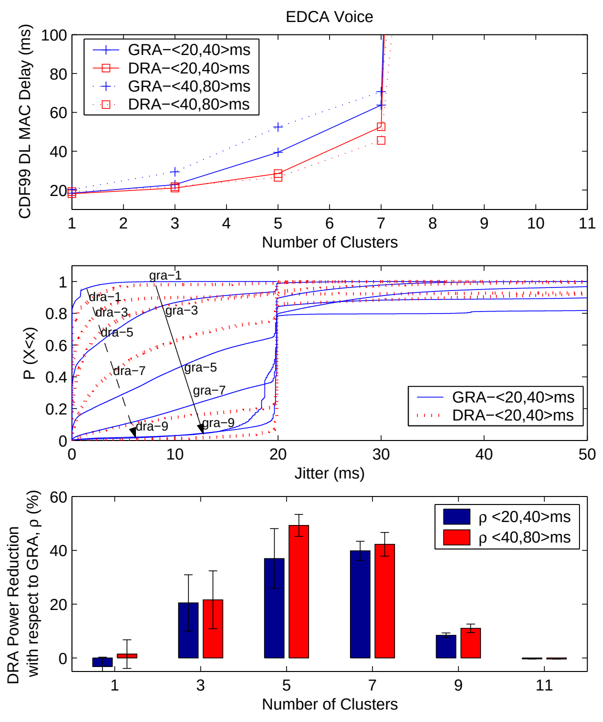
Regarding the Video conference connections, Figure 5.8(b) shows the performance obtained by DRA and GRA, where we can see that smaller but still significant improvements than for Voice are obtained for Video in terms of delay, jitter and power consumption. The reason for the smaller differences observed in the case of Video, specially in terms of delay and jitter, is the *bigger* size of the Video flows. In this case, the amount of traffic transmitted by the flow itself is the major contributor to the delay or jitter, rather than the time the flow has to wait to be served once it awakes, which is one of the main differences between the DRA and GRA schedulers.

5.4.2 Performance of EDCA voice stations

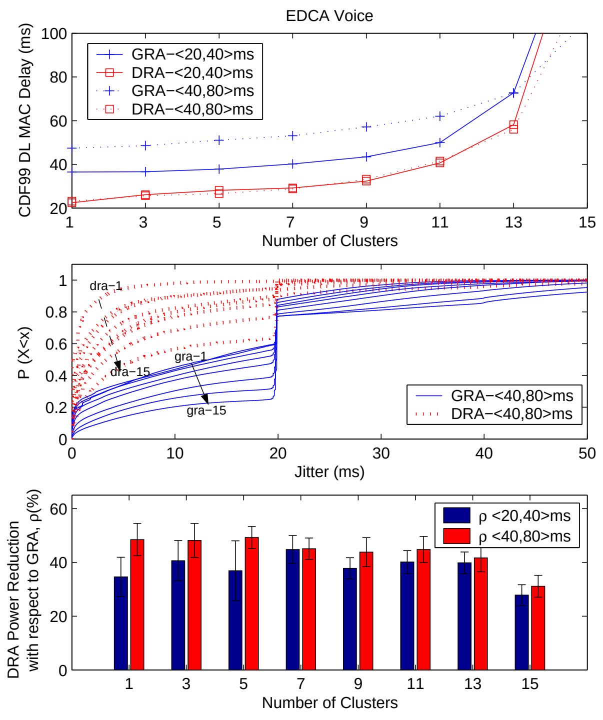
The upper part of Figure 5.9(a) depicts the downlink delay results experienced by the EDCA Voice stations in our experiment. A smaller delay and jitter is experienced by these stations when DRA is used, instead of GRA, for HCCA. In addition, looking at power consumption in the lower part of Figure 5.9(a) we observe that up to a 49% power consumption reduction can be achieved with DRA with respect to GRA.

In order to better understand the impact of GRA and DRA over EDCA Voice stations we perform a modification in our previous experiment. Instead of increasing at the same time all the stations belonging to our basic cluster, we fix the number of HCCA Voice and Video stations to 10 (5 stations of each), consuming approximately a 50% of the available bandwidth, and increase only the number of EDCA stations (Voice and FTP). The results of this modified experiment with fixed HCCA load are shown in Figure 5.9(b).

The upper part of Figure 5.9(b) shows the delay experienced by the EDCA Voice stations in our modified experiment. Interestingly, even when the EDCA load in the network is very small, e.g., 1 station, a bigger worst case delay is experienced by the EDCA Voice stations when GRA is used for HCCA instead of DRA. Indeed, the delay experienced by the EDCA Voice stations shows a strong dependency on the service interval used for HCCA in the case of GRA but not in the case of DRA. The reasons for the observed behavior are the following. In case of using GRA, which places consecutively all HCCA transmissions, and having allocated approximately 50% of the bandwidth for HCCA, an EDCA transmission may have to wait up to $1/2$ of the GRA basic service interval, which is 20ms in the <20,40> configuration and 40ms in the <40,80> configuration, before accessing the medium.



(a) Increasing cluster experiment.



(b) Fixed HCCA load experiment.

Figure 5.9: Performance of EDCA Voice stations.

Thus, when the service interval for HCCA increases, the contiguous HCCA allocation under GRA also increases and so does the worst case delay experienced by EDCA Voice stations. On the other hand, when DRA is used, the EDCA Voice stations see a medium without long allocations for HCCA, which results in a lower worst case delay and a smaller dependency on the HCCA service interval.

The previous effect has also a significant impact on jitter as observed in Figure 5.9(b). In this case, in order to illustrate the underlying dynamics we present the results of the <40,80> configuration, where we can see how blocking the channel for long periods as in GRA, results in backlogged EDCA Voice packets and a bigger jitter. This effect is minimized with DRA.

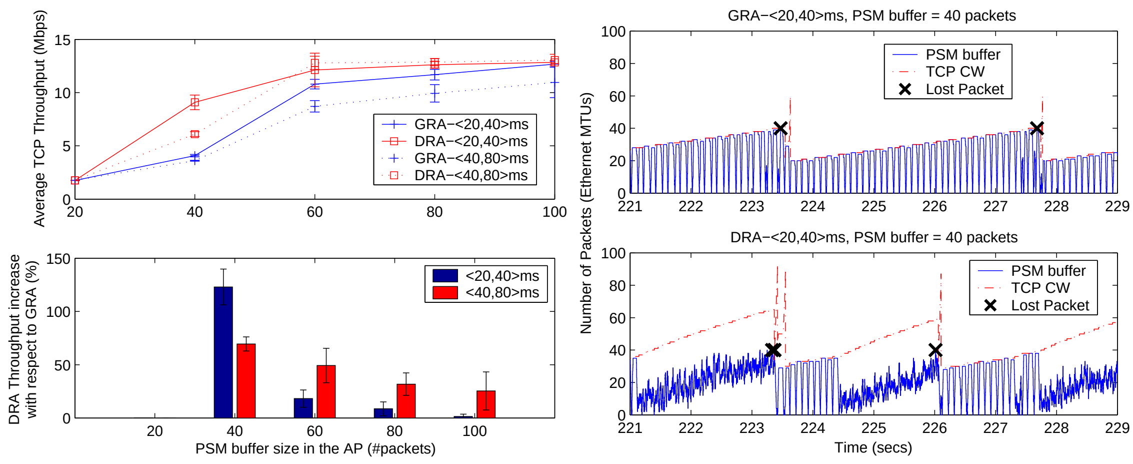
Finally, the lower part of Figure 5.9(b) depicts a consistent power consumption reduction with DRA between 35% and 45% until the network starts to saturate and slightly higher in the <40,80> configuration. We have observed that an important factor affecting the power consumed by EDCA Voice stations is the increased number of retransmissions experienced under the GRA scheduler. The reason is that the long HCCA allocations created by GRA synchronize the EDCA Voice stations trying to access the medium creating regions of a higher collision probability.

Notice that the long HCCA allocations created under GRA could be reduced by computing the transmission slots in GRA considering a worst case transmission rate for the HCCA stations. Thus, if HCCA stations would actually transmit at higher rates, *holes* between consecutive GRA allocations would be created that could be used by EDCA stations. A price though would be paid in the number of simultaneous flows that can operate under HCCA. Instead, DRA by its principle of operation reduces the access delay of EDCA stations to the channel even in the case that HCCA stations operate at their worst-case transmission rate.

5.4.3 Performance of EDCA FTP stations

No significant differences in the performance of EDCA FTP connections were observed in our basic experiment when the DRA or GRA schedulers were used. Therefore, in order to gain a deeper understanding on the behaviour of TCP with the two HCCA schedulers under study, we perform the following simplified experiment. We consider 5 HCCA Video conference and 5 HCCA Voice stations and a single EDCA station performing a FTP download and modify the size of the Power Save (PS) buffer at the AP. The RTT between the AP and the FTP server is kept at 20ms.

Figure 5.10(a) shows the average throughput experienced by the EDCA FTP connection when DRA and GRA are used to schedule the HCCA traffic considering the <20,40> and <40,80> configurations. In addition, in order to better understand the dynamics depcited in Figure 5.10(a), Figure 5.10(b) depicts for GRA and DRA a sample trace containing the number of packets in the PS buffer, the TCP Congestion Window and the lost TCP packets when the size of the PS buffer in the AP is 40 packets.



(a) FTP connection performance.

(b) TCP instantaneous behavior with GRA and DRA schedulers.

Figure 5.10: Performance of EDCA FTP stations.

Looking at the number of packets in the PS buffer for the GRA trace we can see two clear periodic behaviors superimposed. First, a slow saw-tooth pattern due to New Reno's congestion avoidance. Second and more interesting, we observe how the PS buffer empties and fills up periodically. The reason for this second behaviour is the power saving method being employed by the Wi-Fi station. The station awakes to receive the Beacon frame, downloads the buffered frames in the AP, generates the correspondent TCP ACKs in uplink and goes back to sleep before the next window of TCP packets arrives at the AP. This phenomena, first pointed out in (39), increases the effective RTT perceived by the TCP sender and reduces the experienced TCP throughput. However, a different behavior can be observed in the equivalent trace for DRA. Here, after cumulating certain number of packets in the queue, the next window of TCP packets arrives before the station can go back to sleep forcing the station to remain awake. This results in a reduction of the experienced RTT and in an increased throughput. The reason for the previous phenomena is related with the performance inversion effect described in (39) and is explained as follows. Under GRA, TCP traffic is transmitted immediately after a long HCCA allocation. This means that the medium is free for EDCA for a long time until the next HCCA allocation, and so the exchange of TCP packets between the AP and the EDCA FTP station can complete before the new window of TCP packets arrive at the AP, which results in the station going to sleep and the RTT of the TCP connection increasing due to the Beacon interval. Instead, under the DRA scheduler the

EDCA TCP transmissions are more often interrupted by HCCA transmissions. Thus, the transmission of the TCP buffered data takes longer to complete, which results in a new window of TCP data arriving at the AP before the station goes back to sleep, and TCP observing a reduced RTT. The previous effect is specially relevant in the <20,40> configuration.

Based on the previous effect, the dynamics exhibited in Figure 5.10(a) can be easily explained. When the buffering in the AP is small, the EDCA FTP station always goes back to sleep, in both DRA and GRA, before the new window of TCP packets arrives at the AP resulting in a lower throughput. In addition, when the buffering is large, in both DRA and GRA, the TCP congestion window can grow big enough, and the EDCA FTP station stays awake during the lifetime of the TCP connection resulting in an increased throughput. However, it is in the intermediate zone that DRA can provide a significant throughput gain due to the reasons previously stated. Note that a situation of limited buffering is specially relevant for mobile devices that usually support small TCP advertised windows.

With respect to power consumption, no significant differences were observed between DRA and GRA. The reason is that the total power consumed by the EDCA FTP station in this scenario mostly depends on the size of the file being downloaded which is the same in both cases.

5.5 Summary and conclusions

In this chapter we have proposed and analyzed an efficient and scalable scheduler (DRA) for the centralized IEEE 802.11e QoS (HCCA) and power saving (S-APSD) solutions that, unlike most of the proposals existent in the literature, distributes in the channel as much as possible the resource allocations of the different admitted flows. The performance of our *distribution* proposal, DRA, has been compared to a *grouping* one, GRA, and the results indicate that significant improvements are to be expected.

Specifically, the main conclusions that can be drawn from our study are fourfold: i) Distributing in the channel HCCA allocations benefits the isolation of the different flows resulting in significant power savings for stations running S-APSD, up to 45% in our scenarios ii) QoS stringent applications running EDCA and U-APSD, e.g., Voice, also benefit from the distribution of HCCA allocations by experiencing reduced worst case delays and increased power savings, up to 45% in our experiments, iii) Distributing HCCA transmissions in the channel results in a more uniform medium from the perspective of TCP connections in power saving running over EDCA, which significantly improves throughput, depending on the amount of buffering available in the AP, and iv) DRA can significantly increase the capacity of an admission controlled network as described in Appendix 5.B.

Appendix 5.A F function description

Main body of function F

The function F iterates over the list, L, containing the critical points of the absolute minimum distance function in order to find where the maximum of this function is. Since the absolute minimum distance function is composed by consecutive triangular shapes that have as starting and finishing points the critical points, refer to Figure 5.11, our implementation of function F iterates over all critical points in order to find out for which flow $min(dl\_eff\_min_{N+1,i}, dr\_eff\_min_{N+1,i})$, where $N+1$ is the new flow to schedule and i is an already scheduled flow, is minimum at each critical point. Notice though, that since $min(dl\_eff\_min_{N+1,i}, dr\_eff\_min_{N+1,i})$ has discontinuities at its own critical points, i.e. $\phi_{N+1,i} + k \cdot gcd(SI_i, SI_{N+1})$, there is an ambiguity about which value for $min(dl\_eff\_min_{N+1,i}, dr\_eff\_min_{N+1,i})$ to select at these points. Considering for instance critical point $L(j)$ and considering that $min(dl\_eff\_min_{N+1,i}, dr\_eff\_min_{N+1,i})$ has a discontinuity at that point the proper value to chose will depend on whether we are computing the maximum of the triangular shape in the interval $[L(j-1), L(j))$ or in the interval $[L(j), L(j+1))$. In order to handle these cases we define two lists that contain the proper minimum value to select at each critical point for each case, we name these lists the *left* and *right* lists; please refere to Figure 5.11 for an illustration of this idea.

Once the minimum value of $min(dl\_eff\_min_{N+1,i}, dr\_eff\_min_{N+1,i}) \forall i$ at each critical point is known, then each pair of consecutive critical points is analyzed to see which is the maximum of the triangular shape contained between them. The complexity of our implementation is $O(M(N+1))$, where M is the number of critical points in L and N is the number of flows already scheduled in the system. Algorithm 5.3 contains a pseudo-code implementation of function F.

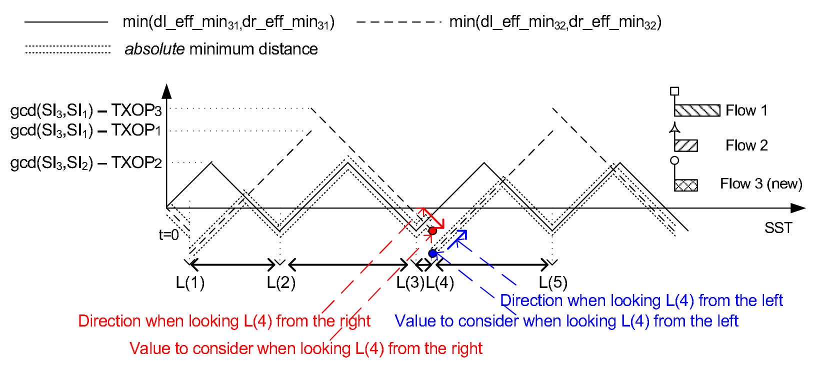

min(dl_eff_min31,dr_eff_min31)
min(dl_eff_min32,dr_eff_min32)
absolute minimum distance
gcd(SI3,SI1) – TXOP3
gcd(SI3,SI1) – TXOP1
gcd(SI3,SI2) – TXOP2
t=0
Flow 1
Flow 2
Flow 3 (new)
SST
L(1)
L(2)
L(3)
L(4)
L(5)
Direction when looking L(4) from the left
Direction when looking L(4) from the right
Value to consider when looking L(4) from the left
Value to consider when looking L(4) from the right

Figure 5.11: F function implementation

Algorithm 5.3: F: function that finds the maximum value of the absolute minimum distance

```
– Variables definition
$L \leftarrow$ Sorted list containing the critical points of the absolute minimum distance.
$partial\_gcd, next\_rel\_time, exec\_time \leftarrow$ Lists containing $gcd(SI_i, SI_{N+1})$, the next release time of each scheduled flow and the execution time of each scheduled flow.
$new\_exec\_time \leftarrow$ Execution time of the new flow.
$val\_crit\_point\_left, val\_crit\_point\_right \leftarrow$ Lists containing the minimum values at each critical point when looking the critical point from left or right.
$dir\_crit\_point\_left, dir\_crit\_point\_right \leftarrow$ Lists containing the directions, UP or DOWN, of the triangular shape having the minimum value in each critical point.
$N, M \leftarrow$ Number of already scheduled flows and size of list $L$.
$SST_{opt}, max\_of\_absolute\_min\_dist \leftarrow$ Values returned by $F$.
– Main body of function F
// Populating the _left and _right lists of minimum values in each critical point
for $i = 1$ to $M$ do
    for $j = 1$ to $N$ do
        $[value, direction, discontinuity, in\_max] \leftarrow$
        $compute\_value\_in\_crit\_point(L(i), next\_rel\_time(j), partial\_gcd(j), exec\_time(j), ...$
        $new\_exec\_time)$
        if $!discontinuity$ then
            if $value < min\_value\_left$ then
                $min\_value\_left \leftarrow value$
                $in\_max$ is $FALSE$ ? $direction\_left \leftarrow direction$ : $direction\_left \leftarrow DOWN$
            if $value < min\_value\_right$ then
                $min\_value\_right \leftarrow value$
                $in\_max$ is $FALSE$ ? $direction\_right \leftarrow direction$ : $direction\_right \leftarrow UP$
        else
            if $-exec\_time(j) < min\_value\_left$ then
                $min\_value\_left \leftarrow -exec\_time(j)$
                $direction\_left \leftarrow UP$
            if $-new\_exec\_time < min\_value\_right$ then
                $min\_value\_right \leftarrow -new\_exec\_time$
                $direction\_right \leftarrow DOWN$
    $val\_crit\_point\_left(i) \leftarrow min\_value\_left$, $dir\_crit\_point\_left(i) \leftarrow direction\_left$,
    $val\_crit\_point\_right(i) \leftarrow min\_value\_right$,$dir\_crit\_point\_right(i) \leftarrow direction\_right$
// Computing the maximum value between each two consecutive critical points
for $i = 1$ to $M - 1$ do
    if $dir\_crit\_point\_left(i)$ is $DOWN$ then
        $peak\_this\_gap \leftarrow val\_crit\_point\_left(i)$
        if $peak\_this\_gap > max\_of\_absolute\_min\_dist$ then
            $max\_of\_absolute\_min\_dist \leftarrow peak\_this\_gap$
            $SST_{opt} \leftarrow L(i)$
    else
        if $dir\_crit\_point\_right(i+1)$ is $UP$ then
            $peak\_this\_gap \leftarrow val\_crit\_point\_right(i+1)$
            if $peak\_this\_gap > max\_of\_absolute\_min\_dist$ then
                $max\_of\_absolute\_min\_dist \leftarrow peak\_this\_gap$
                $SST_{opt} \leftarrow L(i+1)$
        else
            $peak\_this\_gap \leftarrow \frac{|val\_crit\_point\_left(i) - val\_crit\_point\_right(i+1)| + L(i+1) - L(i)}{2} +$
            $\min\{val\_crit\_point\_left(i), val\_crit\_point\_right(i+1)\}$
            if $peak\_this\_gap > max\_of\_absolute\_min\_dist$ then
                $max\_of\_absolute\_min\_dist \leftarrow peak\_this\_gap$
                $SST_{opt} \leftarrow \frac{val\_crit\_point\_right(i+1) - val\_crit\_point\_left(i) + L(i+1) + L(i)}{2}$
```

Function compute_flow_value_in_crit_point()

This function is called by funtion F and computes the value of $min(dl\_eff\_min_{N+1,i}, dr\_eff\_min_{N+1,i})$, $N+1$ being the new flow to schedule and i a given scheduled flow, at a given critical point. Algorithm 5.4 contains a pseudo-code implementation of this function.

Algorithm 5.4: $compute\_flow\_value\_in\_crit\_point()$: Function that finds the value of $min(dl\_eff\_min_{N+1,i}, dr\_eff\_min_{N+1,i})$ at a given point

```
– Variables definition
$pos \leftarrow$ Point where $min(dl\_eff\_min_{N+1,i}, dr\_eff\_min_{N+1,i})$ has to be evaluated.
$next\_rel\_time \leftarrow$ Next release time of the already scheduled flow $i$.
$partial\_gcd \leftarrow gcd(SI_i, SI_{N+1})$.
$exec\_time, nex\_exec\_time \leftarrow$ Variables containing the execution times of the already scheduled flow $i$ and the new flow.
$value, direction, discontinuity, in\_max \leftarrow$ Returned variables.
– Body of function $compute\_flow\_value\_in\_crit\_point()$
//Translating the point into the reference period of its minimum distance signal, i.e. $min(dl\_eff\_min_{N+1,i}, dr\_eff\_min_{N+1,i})$.
$pos\_ref \leftarrow (pos - next\_rel\_time) \mod partial\_gcd$
if $pos\_ref$ is 0 then
    $discontinuity \leftarrow TRUE, in\_max \leftarrow FALSE$
else
    $discontinuity \leftarrow FALSE$
    //Computing the maximum of $min(dl\_eff\_min_{N+1,i}, dr\_eff\_min_{N+1,i})$ and its position.
    $max\_value \leftarrow \frac{|new\_exec\_time - exec\_time| + partial\_gcd}{2} - \max\{new\_exec\_time, exec\_time\}$
    $pos\_max \leftarrow \frac{partial\_gcd + exec\_time - new\_exec\_time}{2}$
    //Returning the value of $min(dl\_eff\_min_{N+1,i}, dr\_eff\_min_{N+1,i})$ at this critical point
    if $pos\_ref$ equals $pos\_max$ then
        $value \leftarrow max\_value$
        $in\_max \leftarrow TRUE$
    else
        if $pos\_ref < pos\_max$ then
            $value \leftarrow max\_value - (pos\_max - pos\_ref)$
            $direction \leftarrow UP, in\_max \leftarrow FALSE$
        if $pos\_ref > pos\_max$ then
            $value \leftarrow max\_value + (pos\_ref - pos\_max)$
            $direction \leftarrow DOWN, in\_max \leftarrow FALSE$
```

Appendix 5.B An initial admission control algorithm based on DRA

In this Appendix we discuss how an admission control algorithm based on our DRA algorithm could be designed, and present an initial performance evaluation of the gains in terms of admission capacity when DRA is employed.

5.B.1 An admission control algorithm for DRA

In order to design an admission control module, the first necessary step is to define a scheduling policy use by the Access Point (AP) whenever more than one flow is waiting for transmission. We will assume in our discussion that this scheduling policy is Earliest Deadline First (EDF), which has been proven optimal under many different settings (113).

Thus, we consider each flow in our system to be defined by the following parameters:

- *Service Start Time*, SST_j. This is initial scheduling time to serve flow j. This value may be determined by an algorithm like DRA.
- *Service Interval*, SI_j. This is the periodic interval at which flow j is granted access to the channel.
- *Transmission Opportunity*, $TXOP_j$. This is the air time granted to flow j every time that this flow is scheduled for transmission.
- *Deadline*, D_j. This is the deadline before which the $TXOP_j$ of flow j has to be completely served, since the moment flow j gets access to the channel. Notice thus that under this definition, $D_j \geq TXOP_j$.

The previous flow definition is illustrated in Figure 5.12.

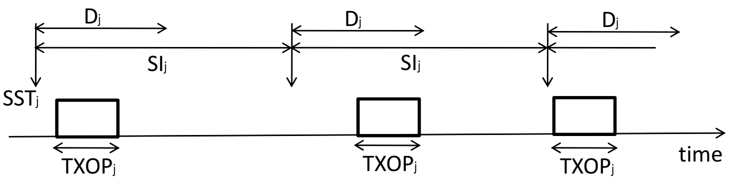


Figure 5.12: Definition of a flow. Notice that since its release time, a flow can be scheduled within a time D_j in order to avoid deadline violations.

Given the previous system definition, the EDF scheduling policy executed in the AP becomes the following. Among the flows pending for transmission at time t, the AP will select the one that fulfills:

$$\min_j\{SST_j + \lfloor\frac{t - SST_j}{SI_j}\rfloor SI_j + D_j - t\} \tag{5.8}$$

In addition, given the previous system model, an admission control algorithm can be defined as an algorithm that is able to answer the following question: *Given a set of flows defined by the 4-tuple* ***S*** $= \{< SST_j, SI_j, TXOP_j, D_j > \; j = 1...N\}$*, and assuming EDF as defined in Equation 5.8 as scheduling policy, do all N flows always fulfill their deadlines?.* If the answer to the previous question is affirmative, the new requesting flow will be granted admission.

Admission control algorithms able to answer the previous question have been extensively studied in Real Time Scheduling Theory (RTST). In particular, in (118) an algorithm is proposed that is able to perfectly answer the previous admission control test by means of essentially simulating the execution of the EDF algorithm during a time $T = lcm(SI_1, ..., SI_N)$, where lcm stands for *least common multiple*. In (118) it is also proved that this algorithm is NP-complete. However, given the usually limited number of flows in a Wi-Fi network (maximum in the order of hundred per AP) and the processing power available in current APs or network controllers, we consider such an algorithm to be feasible in practical Wi-Fi deployments. In addition, an alternative algorithm that allows to trade-off complexity with admission accuracy[1] while also considering our system definition is presented in (119).

Thus, we envision an AP implementing the HCCA building blocks illustrated in Figure 5.13:

1. When a new flow requests access to the network, the AP first runs an algorithm, like DRA, to decide on the SST_j to be assigned to this flow. The rest of parameters in the 4-tuple that defines a flow are explicitly defined or can be derived from the TSPEC sent by the requesting flow. Once, the 4-tuple defining the flow is obtained, the AP adds the requesting flow to the set of the already scheduled flows, and runs the NP complete algorithm defined in (118) to decide whether the new flow can be accepted or not.

2. Whenever several flows are requesting access to the channel at the same time, the AP runs EDF as defined in Equation 5.8 in order to schedule the different flows.

Our goal in this Appendix is to study how an algorithm like DRA, that tries to separate flows as much as possible, can increase the capacity of the network. For this purpose we consider the system

[1]Such an algorithm can result in false negatives, where flows are denied entry to the network when there would actually be enough capacity.

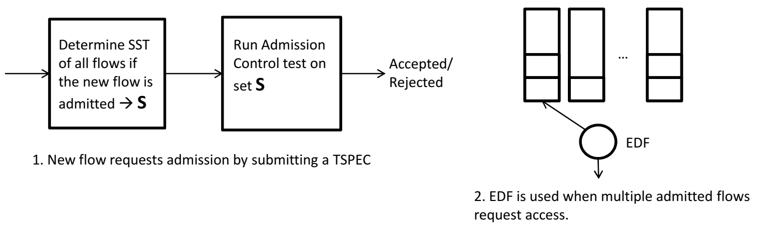

Determine SST of all flows if the new flow is admitted → S
Run Admission Control test on set S
Accepted/ Rejected
1. New flow requests admission by submitting a TSPEC
...
EDF
2. EDF is used when multiple admitted flows request access.

Figure 5.13: Considered HCCA architecture.

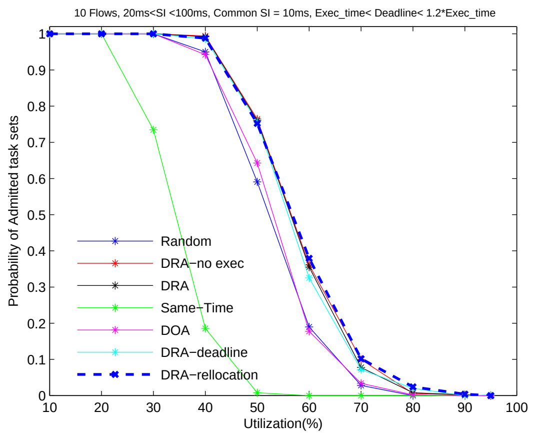
defined in Figure 5.13 and study different algorithms to decide on the SST_j to be assigned to each flow. These algorithms are:

- *Random*. Each flow is assigned an SST_j randomly, specifically: $SST_j = unif(0, SI_j)$.
- *Same-Time*. All flows are granted the same $SST_j = 0$.
- *DOA*. This algorithm corresponds to the Dissimilar Offset Assignment algorithm defined in (116).
- *DRA*. This algorithm corresponds to our DRA algorithm as defined in section 5.2.
- *DRA-no exec*. This algorithm corresponds to our DRA algorithm whithout considering execution times, i.e. $TXOP_j = 0$.
- *DRA-deadline*. This algorithm corresponds to our DRA algorithm where instead of maximizing the minimum effective distance between flows, i.e. d_{eff}, we maximize an abstract definition of distance. Note that the *left* and *right* distances used in DRA, can be re-defined respectively as $d'_{l\_min_{N+1,j}} = d_{l\_min_{N+1,j}} - p_j$ and $d'_{r\_min_{N+1,j}} = d_{r\_min_{N+1,j}} - p_{N+1}$, where p_j and p_{N+1} represent the priority of this flow in the optimization problem, i.e. the bigger p_j the higher the priority of this flow. Thus, in the algorithm at hand we consider $p_j = TXOP_j + TXOP_{N+1} - D_{N+1}$ and $p_{N+1} = TXOP_j + TXOP_{N+1} - D_j$, hence giving a higher priority to flows with smaller deadlines.
- *DRA-rellocation*. This algorithm corresponds to the DRA algorithm defined in section 5.2, where flows are pre-sorted before DRA schedules them. In particular DRA schedules flows in decreasing order of execution time, $TXOP_j$. In this way DRA tries to ensure that *big* flows (those with a big $TXOP_j$) are scheduled further away from each other. Notice that this algorithm assumes that existing flows can be rellocated once a new flow requests access to the system.

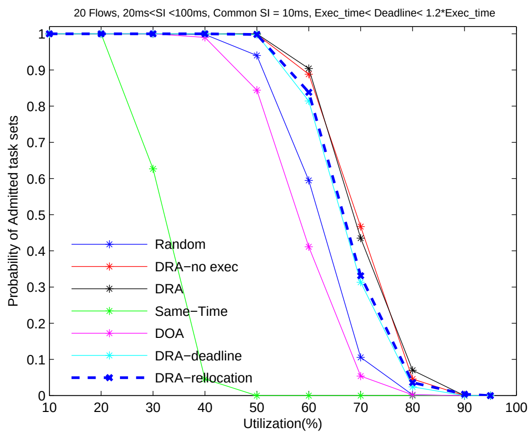
5.B.2 Performance evaluation

In order to evaluate the admission capacity of the previous algorithms under study we perform the following experiment. We fix the number of flows N, the network utilization that these flows incurr U, the maximum allowed service interval $SI_{max} = 100ms$, and a minimum service interval SI_{min} such that $SI_j = kSI_{min}$. Then we assign to each flow a random service interval $SI_j = kSI_{min}$, where $k = unif(1, \frac{SI_{max}}{SI_{min}})$ with $k \in \mathbb{Z}$. We assign to each flow a random $TXOP_j = unif(0, SI_j)$ which is then normalized such that $\sum_{j=1}^{N} \frac{TXOP_j}{SI_j} = U$. Finally, we assign to each flow a random deadline D_j which we allow to be up to 20% higher than $TXOP_j$, i.e. $D_j = TXOP_j \times unif(1, 1.2)$. Notice that using tight deadlines is a way to guarantee energy efficiency because stations will immediately go back to sleep after tranmitting or receiving their data. In our experiments we vary the utilization level U from 0 to 1 while observing the percentage of scheduling sets **S** that can be accepted in the network without any deadline violation, using our different algorithms under study. The results of this experiment are depicted in Figure 5.14 for $SI_{min} = 10ms, 20ms$ and $N = 10, 20$ flows.

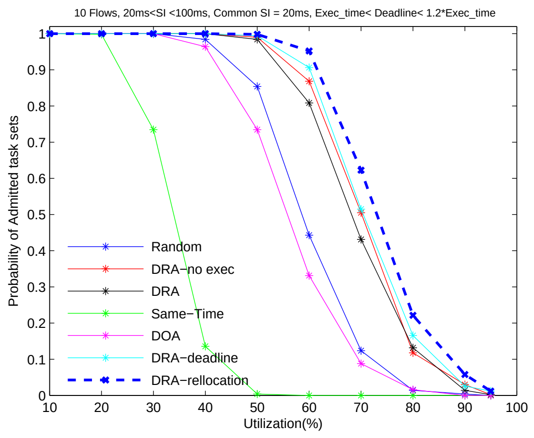
As clearly observed in Figure 5.14, DRA and its correspondent variants can significantly increase the capacity of the network, being specially noticeable the improvement achieved by the *DRA-rellocation* algorithm when $SI_{min} = 20ms$ and $N = 20$ flows. The intuition of why DRA can improve the admission capacity is simple, by scheduling flows further away from each other the likelihood that several flows try to access the channel at the same and a deadline violation occurs is reduced. Further work is needed to further understand the implications of DRA in terms of admission capacity.



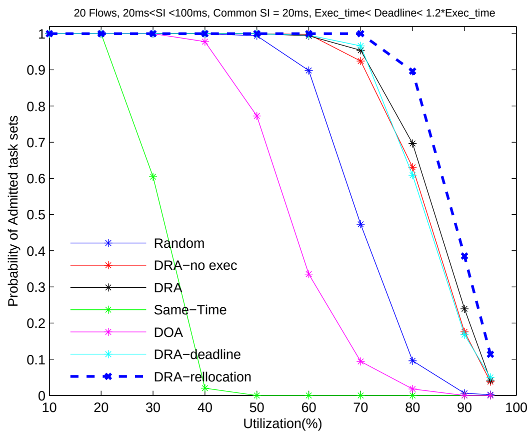
(a) 10 Flows and $SI_{min} = 10$ ms



(b) 20 Flows and $SI_{min} = 10$ ms



(c) 10 Flows and $SI_{min} = 20$ ms



(d) 20 Flows and $SI_{min} = 20$ ms

Figure 5.14: Admission region using an EDF scheduler and a perfect Admission Control.

6

Enhancing the performance of TCP over Wi-Fi Power Saving Mechanisms

Having addressed in the previous chapters the performance and energy efficiency of real-time applications, in this chapter we turn our attention towards the performance and energy trade-off of Data applications in Wi-Fi, where as representative *Data* applications we consider File Transfers and Web browsing. Notice that only the distributed QoS and power saving protocols in Wi-Fi, i.e. EDCA and 802.11 PSM/U-APSD, are suited to transport data traffic, hence these are the protocols considered in this chapter.

It was soon realized that, given the extra latency they introduce, the operation of Wi-Fi power saving protocols could be detrimental to the performance of data applications. Notice that if a station is in power save mode, data arriving at the Access Point may have to wait until the next Beacon transmission time to be delivered, therefore the RTT experienced by a TCP connection may increase and performance may degrade[1]. This degradation can be particularly severe in applications dominated by latency like Web traffic.

In order to address this concern, an extended practice in the industry is to configure a Wi-Fi device to operate in power save mode only until there is traffic to be sent, and then switch the device to active mode until a timeout expires without having sent or received any traffic[2]. An obvious drawback of this approach though, is that when there is a continuous flow of traffic, the device stays all the time in active mode, achieving no energy saving. As discussed in Chapter 2, several works in the literature have also tried to address this concern. The proposed solutions though, either target very specific applications

[1]Note that in a lossy environment the throughput achieved by TCP New Reno degrades as $\sim \frac{1}{RTT}$ (104).

[2]Notice that a Wi-Fi station can alternate between active mode and power save mode by using the power management bit in data frames transmitted to the AP.

like Web, or relay on information coming from the higher layers. We believe that the previous two facts hinder the deployment of these solutions in the market. Therefore, in this chapter we focus on the problem of how to make Data applications energy efficient in Wi-Fi, while adhering to the design guidelines postulated in Chapter 2 which should ensure deployability.

Specifically, our work in this chapter focuses on studying the detailed interactions between Wi-Fi power saving protocols and TCP. To the best of our knowledge this is the first attempt in the literature of such a detailed study. The contents of this chapter have been submitted to (29) and its main contributions are the following:

- We study by means of analysis and simulations, the effect that Wi-Fi power saving protocols have on the performance/energy trade off of long lived TCP traffic. Our study unveils that the efficiency of Wi-Fi power saving protocols critically depends on the bottleneck bandwidth experienced by a TCP connection.
- Based on the obtained insight, we design and evaluate a novel algorithm, BA-TA, which runs in a Wi-Fi station, does not require any modification to existing 802.11 standards, and using only information available at layer two, optimizes the performance/energy trade off experienced by TCP traffic.

This chapter is organized as follows: Section 6.1 analyses the effects of Wi-Fi power saving protocols on long lived TCP traffic. Based on the insights obtained in this section, Section 6.2 introduces the design of our BA-TA algorithm, which is then thoroughly evaluated in Section 6.3. Finally, Section 6.4 summarizes and concludes the chapter.

6.1 TCP performance over *Wi-Fi PSM*

In order to study the performance of TCP over Wi-Fi power saving protocols we consider the scenario depicted in Figure 6.1. This scenario represents a typical deployment, where a battery limited handheld device accesses the Internet through a Wi-Fi Access Point, which is in turn provisioned by an access technology, typically xDSL. Our goal is to study the performance/energy trade-off achieved by Wi-Fi power saving protocols when used with data applications. In particular, we focus on long lived TCP connections[1], since the effect of power saving protocols on short lived connections, like Web, has already been thoroughly studied in the literature (39). In our study, we have modeled a long lived TCP connection using a 50MB File Transfer, which is close to the median size of a video File in the Internet

[1]We define long lived connections as those that enter congestion avoidance.

(91), and have used TCP New Reno as transport protocol which was the most widely spread TCP flavor around 2005 (97)[1].

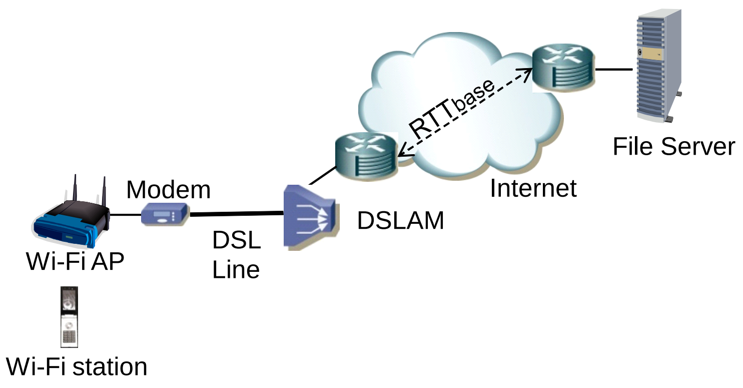


Figure 6.1: Scenario under study.

As it will be shown throughout this section, the performance of such a File Transfer depends on several key parameters in the scenario depicted in Figure 6.1, e.g. the bandwidth of the access and Wi-Fi networks, the amount of available buffering, or the RTT experienced by the connection. In order to analyze the influence of these parameters, we study the scenario depicted in Figure 6.1 by means of packet level simulations using OPNET (64).

Regarding the choice of access technology, we select three representative scenarios that capture today's heterogeneous broadband market. In particular, the selected scenarios span from what are nowadays considered slow access technologies to what are considered fast access technologies. These scenarios are described in Table 6.1.

	DL/UL Rate	**Description**
Slow DSL	1Mbps/128Kbps	Close to the average connection speed in China (99).
Fast DSL	16Mbps/1Mbps	Close to the average connection speed in South Korea (99).
WiFi Bottleneck	100Mbps/100Mbps	Speeds in this range represent FTTH deployments in South Korea or Japan (99).

Table 6.1: Access Technologies under study.

[1]Notice that although new high-speed TCP versions are becoming popular in the last years, e.g. TCP CUBIC is now the default congestion control algorithm in the Linux kernel (98), these high speed TCP versions usually behave like TCP New Reno when they operate in a low bandwidth regime, as in Wi-Fi.

The choice of the Wi-Fi technology is another important factor to consider when studying the performance of power saving protocols. However, there is also a wide variation in today's deployed Wi-Fi technologies, ranging from legacy 802.11b networks to the latest 802.11n networks. In order to capture this variation, we consider in our study that the Wi-Fi network depicted in Figure 6.1 can operate according to two different configurations: i) a *Slow Wi-Fi* configuration, where a low over-the-air priority is used and no Transmission Opportunities (TXOP) are allowed[1], and ii) a *Fast Wi-Fi* configuration, where a high over-the-air priority is used and TXOPs are allowed. The used Wi-Fi configurations are described in Table 6.2.

	Data/Control Rate	**AIFS/CWmin/CWmax/TXOP**
Slow WiFi	54/24 Mbps	3/15/1023/0ms
Fast WiFi	130/24 Mbps	2/7/15/3ms

Table 6.2: Wi-Fi Configurations under study.

For each of the previous scenarios we will evaluate the performance/energy trade-off of a generic File Transfer, using two different algorithms. The first one represents the common industry practice of switching to *Active Mode* once traffic is detected, which should be optimal in terms of performance. Hereafter we refer to this algorithm as the *Active Mode* algorithm. The second algorithm consists of having the station all the time in power save mode, which should be optimal in terms of energy. We refer to this algorithm as the *Wi-Fi PSM* algorithm. In particular, we use U-APSD as Wi-Fi power saving protocol, where the station wakes up at every Beacon to check if there is buffered traffic and in that case retrieves it using a single signaling trigger. More details on this U-APSD implementation can be found in (47).

In order to evaluate energy consumption, we make use of our regular power consumption model that consists of four basic states: Sleep, Listen, Reception and Transmission. Energy is computed by integrating the power that a Wi-Fi device spends in each of the previous states over a certain target time, which in our evaluation will be the time to transfer a File. The power consumption values used are shown in Table 6.3 (24).

Broadcom 4311^{TM}	**Sleep**	**Listen**	**Rx**	**Tx**
Power (mW)	20	390	1500	2000

Table 6.3: Power consumption levels used in our study.

[1]TXOPs allow a Wi-Fi device to transmit several frames with a single access to the channel, hence significantly reducing overhead. A description of the EDCA protocol is included in Chapter 2.

Finally, for each of the scenarios and algorithms under study, we have performed two different experiments that aim to represent the different conditions that can be experienced by a TCP connection, specifically:

- *Buffer Experiment*, where we configure $RTT_{base} = 20ms$ in Figure 6.1[1], and vary the buffering available in the DSLAM or in the AP (depending on the considered scenario), from 20 to 160 packets. Notice that given the potential harm caused by large buffers (101), it is important to study the performance delivered by the protocols under study with small buffers.
- *RTT experiment*, where we fix the buffering available in the DSLAM and AP to 50 packets and 100 packets respectively, and vary the value of RTT_{base} in Figure 6.1 from 10ms to 310ms.

6.1.1 Buffer experiment

We start analyzing the results of our first experiment, where we fix $RTT_{base} = 20ms$ in Figure 6.1 and vary the amount of buffering available in the DSLAM and in the AP in our scenarios under study.

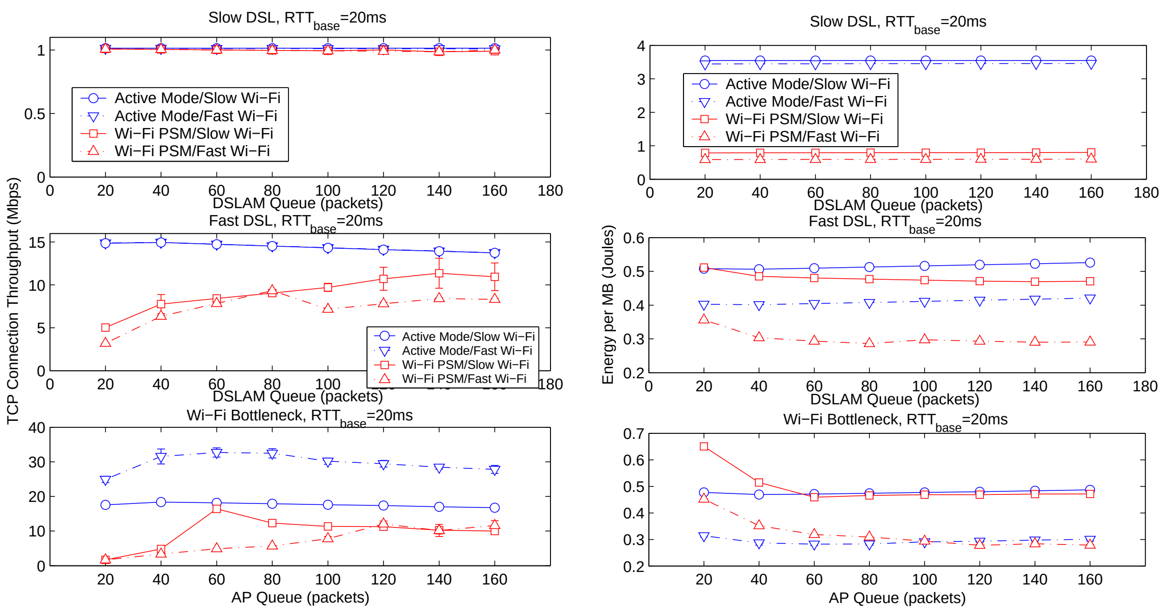
6.1.1.1 Slow DSL scenario

The upper graphs in Figures 6.2(a) and 6.2(b) depict respectively the throughput and energy consumption experienced by a Wi-Fi station when retrieving a 50MB File from a server located in the Internet, as depicted in Figure 6.1. It is clearly seen in these figures how all the algorithms under study deliver a similar throughput, which corresponds to the bottleneck bandwidth (1 Mbps), and how being in power save mode during the File Transfer drastically reduces the energy required by the Wi-Fi station to retrieve the File (up to a five fold decrease).

In order to understand why *Wi-Fi PSM* achieves such a significant energy reduction at no performance penalty, Figure 6.3(a) depicts the time evolution of the TCP congestion window (blue line), the queue occupancies at the DSLAM (red line) and the Wi-Fi AP (black line), and the power state of the Wi-Fi station (awake or sleep) during the File Transfer. We can see in the graph, how the Wi-Fi station wakes up before the Beacon is sent (dotted vertical lines in the graph), quickly empties the AP buffer using the high bandwidth available in the Wi-Fi network, and efficiently sleeps until the next Beacon. Instead, in the *Active Mode* algorithm, the Wi-Fi station stays most of the time awake but inactive, waiting for the next TCP packet[2].

[1]Typical RTT in a domestic Internet path (86).

[2]A 1.5KB packet needs 12ms to traverse a 1Mbps DSL line.



(a) File transfer throughput.

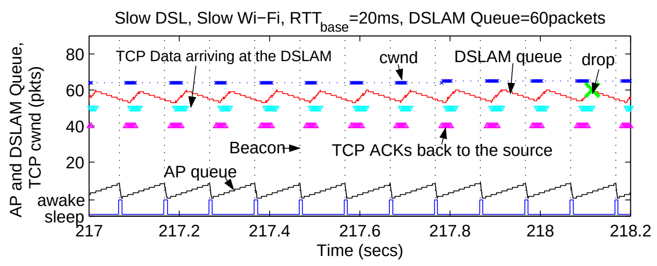
(b) Energy spent in the File transfer.

Figure 6.2: Buffer Experiment. For each of the algorithms and scenarios under study, throughput is depicted on the left graph and energy on the right one. Notice that the same legend applies to all sub-figures.

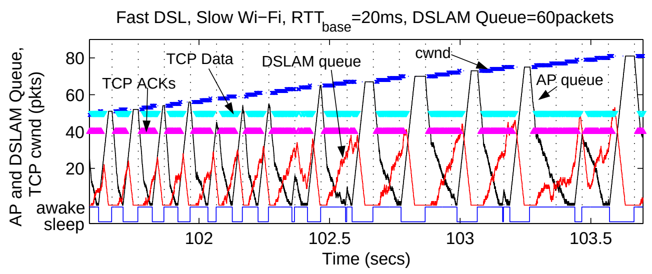
The key to understand why the *on/off* behavior introduced by the power saving protocol does not reduce performance, is to realize that in this experiment the bottleneck queue (red line in Figure 6.3(a)) *never gets empty*. Thus, a necessary condition to avoid degrading throughput in *Wi-Fi PSM*, is that the TCP source's congestion window (hereafter referred to as *cwnd*) needs to grow big enough in order to compensate for the increased RTT introduced by the power saving protocol, i.e. $RTT_{eff} = \lceil \frac{RTT_{base}}{BI} \rceil BI$, where RTT_{eff} stands for effective RTT, BI is the Beacon Interval and RTT_{base} is depicted in Figure 6.1. Thus, the minimum $cwnd$, in MTU-sized packets, required to avoid any throughput degradation equals to:

$$cwnd > \frac{RTT_{eff} \times R_{DSL_{DL}}}{MTU} = \frac{\lceil \frac{RTT_{base}}{BI} \rceil BI \times R_{DSL_{DL}}}{MTU} \tag{6.1}$$

Given the low speed of the DSL link ($R_{DSL_{DL}} = 1Mbps$) and the small RTT_{base} ($20ms$) considered in this scenario, Equation 6.1 results in $cwnd > 8.53$ packets, which is easily achieved given the considered buffer sizes.



(a) Dynamics with a Slow DSL access network.



(b) Dynamics with a Fast DSL access network.

Figure 6.3: TCP dynamics. The upper graph depcits the *Slow DSL* scenario, and the lower one the *Fast DSL* scenario.

6.1.1.2 Fast DSL scenario

We now repeat the previous experiment in the *Fast DSL* scenario. Throughput and energy results are depicted respectively in the middle graphs of Figures 6.2(a) and 6.2(b), and reveal a very different behavior than the one observed in the *Slow DSL* scenario. In this case the File Transfer throughput significantly degrades with *Wi-Fi PSM*, specially when the DSLAM buffer is small, and interestingly it degrades even more when more bandwidth is available in the Wi-Fi network (*Fast Wi-Fi* configuration). Regarding energy, *Wi-Fi PSM* continues to be more efficient than the *Active Mode* algorithm, although a smaller gain than in the *Slow DSL* scenario is obtained.

In order to get a deeper understanding on the underlying dynamics, Figure 6.3(b) illustrates, when the station uses *Wi-Fi PSM*, an example of the AP and DSLAM queue dynamics (red and black lines) and the evolution of the TCP sender's *cwnd* (blue line). We can immediately see how the dynamics in this case are much more involved that in the *Slow DSL* scenario. In particular, the performance of TCP over *Wi-Fi PSM* depends in this case on the following three effects: i) The increased bandwidth delay product introduced by *Wi-Fi PSM*, ii) Bursty increases in the DSLAM buffer due to ACK compression, and iii) Extended awake periods experienced by the Wi-Fi station. Next, we describe these effects in detail.

We can clearly see in Figure 6.3(b), how the DSLAM queue (red line) often returns to zero leading to periods where the DSL line is underutilized. Using the lessons learned in the *Slow DSL* scenario, we can see that in order to keep the DSL line always busy in this scenario, the TCP sender's $cwnd$ should be at least $cwnd_{min} > \frac{\lceil \frac{RTT_{base}}{BI} \rceil BI \times R_{DSL_{DL}}}{MTU} = 133.3$ packets, which is not the case in Figure 6.3(b).

Notice thus, that when *Wi-Fi PSM* is used it is critical for the TCP sender to be able to achieve a high $cwnd$ in order to compensate for the increased badwidth delay product introduced by the power saving protocol. The maximum $cwnd$ achieved by a TCP sender depends on the bandwidth delay product of the connection and on the buffering available in the bottleneck. However, another important factor limits the achievable $cwnd$ when *Wi-Fi PSM* is used. This factor is the ACK compression introduced by the power saving protocol.

We can see in Figure 6.3(b), how when *Wi-Fi PSM* is used, the AP holds TCP packets for the station until the Beacon time, and then transmits these packets to the station at the Wi-Fi rate, which is faster than the bottleneck rate (the DSL rate). The Wi-Fi station then reflects back to the TCP source TCP ACKs at the rate of the received data, which result in a *compressed* burst of new TCP packets arriving at the DSLAM that causes a bursty queue increase, as clearly observed in Figure 6.3(b) (red line). This process is graphically described in Figure 6.4, where it can be seen how the ACK compression rate is typically limited by the upstream DSL bandwidth.

In order to understand how ACK compression affects the maximum $cwnd$ achieved by the TCP source, it is possible to compute the maximum burst size N_{max} (in MTU-sized packets) that results in no packet loss when arriving at the DSLAM buffer. Considering that, as illustrated in Figure 6.4, a new window of TCP data hits the bottleneck node at a rate equal to $R_{ACKcomp}$ MTU-sized packets/sec, and that the rate at which the bottleneck queue is drained is in our case $R_{DSL_{DL}}$ MTU-sized packets/sec, it is easy to see that while a new window of TCP data hits the bottleneck, this queue grows at a rate of $R_{qbott} = R_{ACKcomp} - R_{DSL_{DL}}$. Thus, if the previous queue growth is sustained during a time T_{win}, such that $T_{win} R_{qbott} > Q_{DSLAM}$, there will be a loss in the bottleneck queue limiting the size of $cwnd$. Considering that packets within a TCP window arrive at the bottleneck with an interarrival time of $T_{arrival} = \frac{1}{R_{ACKcomp}}$, with $R_{ACKcomp}$ defined in Figure 6.4, we have that $T_{win} = \frac{N_{max}}{R_{ACKcomp}}$, and the maximum burst size, N_{max}, that results in no drop in the bottleneck's queue is:

$$N_{max} < Q_{DSLAM} \frac{R_{ACKcomp}}{R_{qbott}} = \frac{Q_{DSLAM}}{1 - \frac{R_{DSL_{DL}}}{R_{ACKcomp}}} \tag{6.2}$$

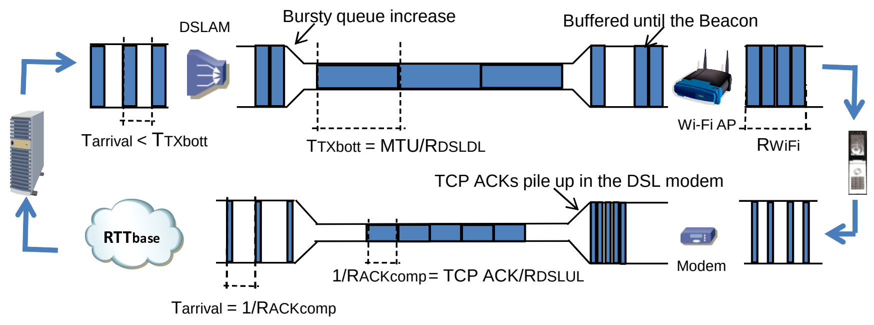


Figure 6.4: ACK compression in a TCP connection, caused by Wi-Fi power saving protocols.

Where the previous equation only holds if $R_{ACKcomp} > R_{DSL_{DL}}$. Notice that Equation 6.2 results in $N_{max} < 89$ packets in the *Fast DSL* scenario, which limits the maximum value of $cwnd$. In the next sections we will study in more detail the relation between N_{max} and $cwnd$.

It is interesting to notice, looking at Equation 6.2, that when the amount of ACK compression increases, e.g. if the DSL upstream capacity increases in Figure 6.4, then N_{max} decreases and the TCP connection throughput should degrade. Later we will evaluate the validity of this claim.

Finally, there is a third effect affecting the TCP connection throughput. Unlike the previous two effects though, this effect will tend to increase TCP throughput, and is described as follows. If the time required by the Wi-Fi station after the Beacon to retrieve the buffered TCP packets and generate the correspondent TCP ACKs, is above RTT_{base} in Figure 6.1, then the Wi-Fi station will still be awake when a new window of TCP data from the File Server arrives to the AP, and so this new window of data will not suffer from an increased RTT. This fact is often observed in Figure 6.3(b), for instance between 103.2 secs and 103.4 secs. Notice though, that since in this scenario the Wi-Fi rate is higher than the DSL rate, the Wi-Fi station eventually manages to empty the AP queue and return to sleep, resulting again in an increased RTT and reduced throughput. In addition, notice that the previous effect is more likely to occur when the bandwidth in the Wi-Fi network reduces, because in this case the AP takes longer to transmit data to the station. This effect explains why when *Wi-Fi PSM* is used, the *Slow Wi-Fi* configuration outperforms the *Fast Wi-Fi* one as depicted in Figure 6.2(a).

To conclude the analysis of the *Fast DSL* scenario, we turn our attention to Figure 6.2(b) that depicts the energy spent by the Wi-Fi station to retrieve the 50MB File. The reason why the energy reduction obtained by *Wi-Fi PSM* is less significant than in the *Slow DSL* scenario, is that the DSL line is closer to saturate the Wi-Fi network, hence the station does not spend so much time inactive when being in active mode. Finally, it is worth noticing that the *Fast Wi-Fi* configuration reduces energy consumption because TCP packets are transmitted faster.

6.1.1.3 Wi-Fi bottleneck scenario

We finalize the analysis of our Buffer experiment looking at the throughput and energy performance obtained in the *Wi-Fi Bottleneck* scenario. These results are depicted in the lower graphs of Figures 6.2(a) and 6.2(b). Looking first at throughput, we can see that the effects observed in the *Fast DSL* scenario appear now even more magnified: i) *Wi-Fi PSM* significantly degrades throughput, specially when the buffer in the AP is small, and ii) the *Fast Wi-Fi* configuration provides the expected throughput increase when the station uses *Active Mode*, but results in a severe throughput degradation when the station uses *Wi-Fi PSM*.

Wi-Fi PSM also introduces ACK compression in this scenario, but, since upstream bandwidth is not a problem in this case, the amount of ACK compression depends on the particular mechanisms employed in the Wi-Fi network. For instance, when the *Slow Wi-Fi* configuration is used, the AP and the station compete in a fair way after each Beacon in order to transmit the buffered TCP packets and the corresponding TCP ACKs. Therefore the average time between TCP ACK and TCP data transmissions is the same, and no significant ACK compression is introduced. However, when the *Fast Wi-Fi* configuration is used, the AP first transmits to the station all its buffered TCP packets packed within a TXOP. This transmission is then followed by transmission where the station packs all the TCP ACKs within another TXOP. Thus, assuming that AP and station use the same data rate, the ACK compression rate in this case can be approximated as $\frac{R_{ACKcomp}}{R_{Wi-Fi}} = \frac{Size\ TCP\ Packet}{Size\ TCP\ ACK}$.

The overlapping between new TCP arrivals and the awake periods of the *Wi-Fi PSM* station also occurs in this case, but mostly in the *Slow Wi-Fi* configuration. The high data rates and bursty transmissions used in the *Fast Wi-Fi* configuration reduce the likelihood of this positive effect.

We highlight now another degradation[1] introduced by *Wi-Fi PSM*, that is caused by an increase in the time required by the TCP connection to complete the initial *Slow Start+Fast Recovery* phase. In our experiments, a TCP connection typically finalizes the initial Slow Start phase when tree duplicated ACKs are received at the TCP source; at that moment TCP New Reno triggers a fast retransmission and the connection enters Fast Recovery until all outstanding data is successfully transmitted. Notice that during the initial Slow Start, the TCP connection typically saturates the network, forcing the *Wi-Fi PSM* station to stay awake. However, when Fast Recovery starts, the TCP source stops transmitting data until half a window of duplicated ACKs is received (103), which allows the station in power save mode to drain the AP buffer and return to sleep. After that moment, since Fast Recovery only injects new data or retransmissions upon receiving duplicate or partial ACKs, transmissions only occur once every 100ms

[1]This phenomena also occurred in the *Slow DSL* and *Fast DSL* scenarios, but had a smaller impact.

when the station wakes up after the Beacon, hence slowing down the completion of this phase. Figure 6.7(b) depicts a typical initial *Slow Start+Fast Recovery* phase when the station uses *Active Mode* and when the station uses *Wi-Fi PSM*, in an experiment where $RTT_{base} = 20ms$.

Finally, we can see in Figure 6.2(b), that when the Wi-Fi network is the bottleneck the *Active Mode* algorithm turns out to be the most energy efficient algorithm, because the Wi-Fi station spends all its time transmitting and receiving useful data.

6.1.2 RTT experiment

We describe now the results of our second experiment, where we fixed the maximum buffering in the DSLAM and AP and observed the throughput/energy trade-off when varying RTT_{base} from 10ms to 310ms. We considered a maximum buffer of 50 packets in the DSLAM in the *Slow DSL* and *Fast DSL* scenarios, and of 100 packets in the AP in the *Wi-Fi Bottleneck* scenario.

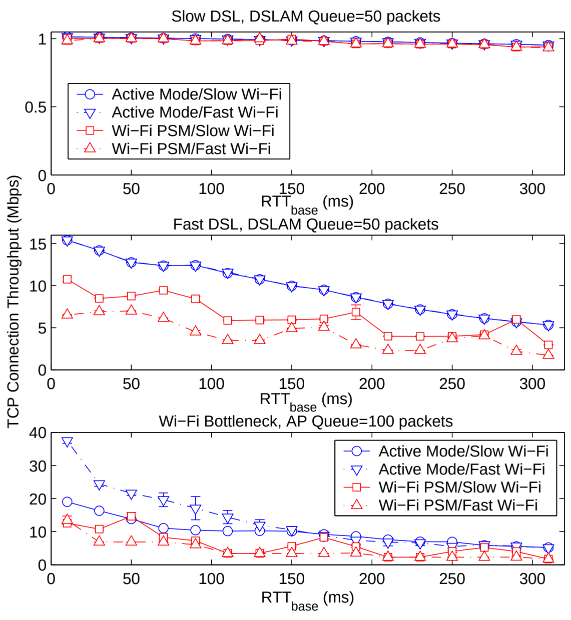
6.1.2.1 Slow DSL scenario

As illustrated in the upper part of Figures 6.5(a) and 6.5(b), the behavior of the algorithms under study in the *Slow DSL* scenario continues to be very similar than the one observed in our first experiment. Following Equation 6.1 we can see that even when $RTT_{base} = 310ms$ the $cwnd$ required to keep the DSL line always busy is $cwnd > 33.3$ packets, which is easily achieved considering the configured DSLAM buffer of 50 packets.

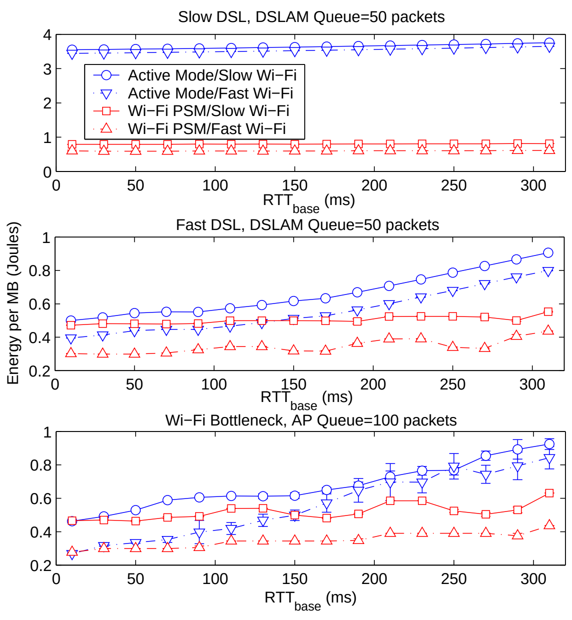
6.1.2.2 Fast DSL scenario

The middle graph in Figures 6.5(a) and 6.5(b) depict respectively the throughput and energy performance of the algorithms under study in the *Fast DSL* scenario. It is interesting to observe in this case that when the station is in power save mode, there is a ripple effect in the connection's throughput as RTT_{base} varies. Notice that this ripple is contrary to the common knowledge that TCP throughput degrades when the RTT increases, as clearly observed when the station uses *Active Mode*. Next, we present an analytical model aimed at capturing the interactions between the TCP congestion control mechanisms and the Wi-Fi power saving protocols that result in the observed ripple effect. Our simplified model is based on the following assumptions:

i. We consider the throughput experienced by the TCP connection during the congestion avoidance phase.



(a) File transfer throughput.



(b) Energy spent in the File transfer.

Figure 6.5: RTT Experiment. For each of the algorithms and scenarios under study, throughput is depicted on the left graph and energy on the right one. Notice that the same legend applies to all sub-figures.

ii. The Wi-Fi station retrieves all the TCP packets buffered at the AP and sends the corresponding TCP ACKs before a new window of TCP data arrives. Notice that this behavior is more likely as the Wi-Fi bandwidth increases (i.e. *Fast Wi-Fi* configuration).

iii. We consider that a packet is never lost in the AP but only in the DSLAM (i.e. Q_{AP} is big enough).

iv. The rate at which TCP ACKs arrive at the TCP source is assumed to be constrained by the uplink DSL modem, i.e. $T_{arrival} = \frac{Size\ TCP\ ACK}{R_{DSL_{UL}}}$ (see Figure 6.4).

Under the previous conditions, and leveraging Equation 6.2, we can define $N_{max} = \frac{Q_{DSLAM}}{1-\frac{R_{DSL_{DL}}}{R_{ACKcomp}}}$, as being the maximum burst size (in MTU-sized packets) that can hit the DSLAM node without resulting in a packet drop. In addition, we will define Δ as being the *maximum* number of TCP data packets arriving from the DSLAM to the the AP that a Wi-Fi station can retrieve after the Beacon frame before going to sleep[1]. Δ depends on the relation between RTT_{base} and the Beacon Interval (BI), and on the Wi-Fi rate. Specifically, we will approximate Δ as $\Delta = M + L$, where $M = \frac{BI - BI \mod RTT_{base}}{T_{TX_{bott}}}$ is the maximum number of packets transferred from the DSLAM to the AP since the first packet of the new TCP window hits the DSLAM until the next Beacon time, and $L = \frac{T_{TX_{Wi-Fi}}(M)}{T_{TX_{bott}}}$ is the number of packets that can be transferred from the DSLAM to the AP while the Wi-Fi station is retrieving the first M packets; $T_{TX_{bott}} = \frac{MTU}{R_{DSL_{DL}}}$ is defined in Figure 6.4, and $T_{TX_{Wi-Fi}}(M)$ is the time to transmit M packets and generate the corresponding TCP ACKs in the Wi-Fi network. These variables are illustrated in Figure 6.6(a).

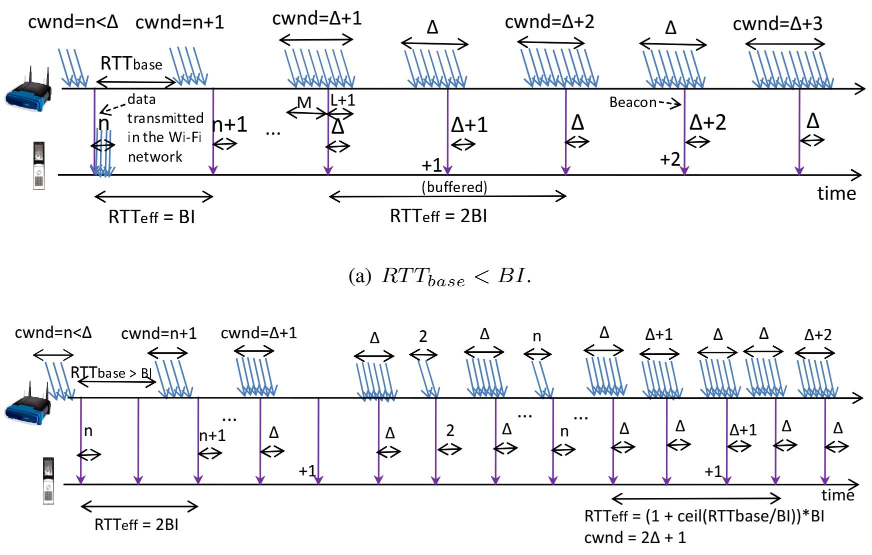
The key to understand the ripple effect observed in Figure 6.5(a), is to realize that when the station is in power save mode, increasing RTT_{base} may result in the TCP source being able to achieve a higher $cwnd_{max}$ and therefore a higher throughput. Specifically, the maximum size of $cwnd$ depends on the relation between N_{max} and Δ.

We start considering the case where $N_{max} \leq \Delta$, i.e. a new window of TCP packets is entirely transferred to the station before this goes back to sleep. Under this condition $cwnd_{max} = N_{max}$, and the RTT experienced by the connection is $RTT_{eff} = \lceil \frac{RTT_{base}}{BI} \rceil BI$. This scenario is illustrated in the left hand side of Figure 6.6(a), from where it is easy to see that the throughput of the TCP connection in congestion avoidance can be computed as:

$$Thr = \frac{1}{2}\frac{cwnd_{max} + cwnd_{min}}{RTT_{eff}} = \frac{3}{4}\frac{N_{max}}{\lceil \frac{RTT_{base}}{BI} \rceil BI} \tag{6.3}$$

Consider now that $N_{max} > \Delta$ and $RTT_{base} \leq BI$. The dynamics in this case are again illustrated in Figure 6.6(a). Notice in the figure how when $cwnd$ grows above Δ, the station goes back to sleep

[1]Notice, that if $R_{Wi-Fi} > R_{DSL_{DL}}$ there is always going to be such maximum.



(a) $RTT_{base} < BI$.

(b) $RTT_{base} > BI$.

Figure 6.6: TCP model. The two graphs depict the data arrivals from the DSLAM to the AP, the Beacon transmissions from the AP to the Wi-Fi station, and the delivery of the buffered data after the Beacon.

before retrieving the complete TCP window, and a fraction of this TCP window remains in the AP and is transmitted in a subsequent Beacon (see third Beacon in Figure 6.6(a)). Therefore, when $cwnd > \Delta$, the TCP connection effectively transfers $cwnd + \Delta$ packets every $2BI$. Instead, when $cwnd < \Delta$ the connection transfers a $cwnd$ worth of packets every BI. Finally, notice that the maximum value of $cwnd$ is $cwnd_{max} = N_{max}$, and $cwnd_{min} = \frac{N_{max}}{2}$. Hence, after the appropriate algebraic manipulation, the TCP connection throughput in this case can be computed as:

$$Thr = \begin{cases} \frac{\Delta + \frac{3N_{max}}{4}}{2BI} & cwnd_{min} \geq \Delta \\ \frac{1}{2}\frac{N_{max}+(3-\alpha)\Delta+(1-\alpha)}{2BI} & cwnd_{min} < \Delta \end{cases} \tag{6.4}$$

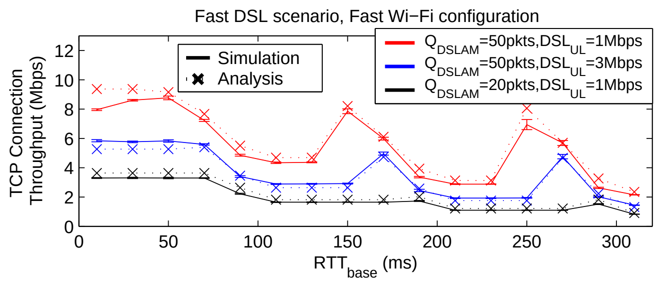
Where $\alpha = \frac{\Delta - cwnd_{min}}{cwnd_{max} - cwnd_{min}}$.

Finally, let us consider that $N_{max} > \Delta$ and $RTT_{base} > BI$. The dynamics in this case are illustrated in Figure 6.6(b), where it can be seen that every time that $cwnd$ reaches a value multiple of Δ, i.e. $cwnd = k\Delta$ with $k \geq 1$, packets start to be transmitted in a subsequent Beacon. Thus, only when $cwnd > \lceil \frac{RTT_{base}}{BI} \rceil \Delta$ the amount of packets hitting the DSLAM after a Beacon, starts growing above Δ until N_{max}. Therefore, in this case $cwnd_{max} = N_{max} + \lceil \frac{RTT_{base}}{BI} \rceil \Delta$, and the effective RTT experienced by the TCP connection is $RTT_{eff} = (1 + \lceil \frac{RTT_{base}}{BI} \rceil)BI$. Recalling that $N_{max} > \Delta$ and

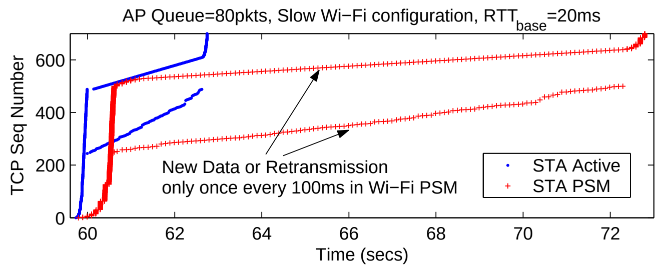
$RTT_{base} > BI$, it can be derived that $cwnd_{min} > \Delta$, therefore:

$$Thr = \frac{3}{4}\frac{N_{max} + \lceil \frac{RTT_{base}}{BI} \rceil \Delta}{(1 + \lceil \frac{RTT_{base}}{BI} \rceil) BI} \tag{6.5}$$

We can now understand the reasons behind the ripple effect depicted in Figure 6.5(a). We have shown how small changes in the connection parameters, e.g. RTT_{base}, can result in a different phenomena dominating the dynamics of RTT_{eff} and $cwnd$, and in a completely different throughput, as identified in Equations 6.3, 6.4 and 6.5. In order to validate the presented model, we compare in Figure 6.7(a) the throughput predicted by the model against the throughput obtained in simulations[1]. It is also interesting to notice in Figure 6.7(a), how when the upstream DSL bandwidth increases ($R_{DSL_{UL}} = 3Mbps$), the connection throughput degrades due to an increase in ACK compression.



(a) Validation of model accuracy.



(b) Dynamics of the initial Slow Start and Fast Recovery phases.

Figure 6.7: TCP dynamics.

Regarding energy, it is worth to notice looking at the lower part of Figure 6.5(b), that an always active configuration severely penalizes energy consumption when RTT_{base} increases, because TCP can not keep the DSL always busy and hence the station is often inactive in this case.

[1] In our simulations, we choose the *Fast Wi-Fi* configuration, because it better fulfills condition ii. of our model, and consider only the congestion avoidance phase of the connection.

6.1.2.3 Wi-Fi bottleneck scenario

Finally, the lower graphs in Figures 6.5(a) and 6.5(b) depict respectively for our RTT experiment the throughput and energy performance in the *Wi-Fi Bottleneck* scenario. We can see that, like in the *Fast DSL* scenario, a ripple effect appears in throughput, but this time only in the case of the *Slow Wi-Fi* configuration.

The analysis presented in the previous section can also be used in this case to understand the TCP dynamics. For instance, recall from the previous section that $M = \frac{BI - BI \mod RTT_{base}}{T_{arrival}}$ packets must be buffered at the AP before a Beacon frame is sent, where $T_{arrival}$ is the interarrival time of packets arriving at the bottleneck. In the *Fast DSL* scenario we had that $T_{arrival} = \frac{Size\ TCP\ ACK}{R_{DSL_{UL}}}$, however in this scenario packets may hit the bottleneck (the AP) with smaller interarrival times. For instance, as previously explained, when the *Fast Wi-Fi* configuration is used the Wi-Fi station transmits TCP ACKs within a TXOP which results in packets arriving at the AP with small $T_{arrival}$ times. Thus, if $T_{arrival}$ is small enough so that M grows above Q_{AP}, a drop will occur at the AP's power save buffer which limits the maximum $cwnd$ to $cwnd_{max} < Q_{AP}$. Indeed, this is what happens in the case of the *Fast Wi-Fi* configuration, and therefore the achieved throughput (during the congestion avoidance phase) in this case is free of ripples and is described by:

$$Thr = \frac{3}{4} \frac{Q_{AP}}{\lceil \frac{RTT_{base}}{BI} \rceil BI} \tag{6.6}$$

The behavior is different in the *Slow Wi-Fi* configuration, where TCP packets do not arrive at the AP so close to each other, i.e $T_{arrival}$ is bigger, and therefore the ripple effect arises. For instance, when $RTT_{base} = 110ms$ a new window of TCP data arrives at the AP approximately 90ms before the next Beacon time, which results in the AP having to be able to buffer 90ms worth of TCP Data arrivals to avoid a packet drop. This is not possible given the considered buffer of $Q_{AP} = 100$ packets, and therefore $cwnd_{max} = Q_{AP}$ in this case. However, when $RTT_{base} = 150ms$ the AP only has to be able to buffer 50ms worth of arrivals to avoid a packet drop, and so $cwnd_{max}$ can grow above Q_{AP}, obtaining a higher throughput in this case.

Regarding energy, a similar behavior like in the *Fast DSL* scenario is observed in this case.

Finally, we would like to notice that in this study we have only reported on the performance of downlink File Transfers, i.e. from the File Server to the Wi-Fi station. We have however also studied how uplink File Transfers behave, observing that, given the limited upstream bandwidths, *Wi-Fi PSM* usually delivers a fairly good throughput/energy trade off for uplink connections.

6.2 BA-TA: An adaptive triggering algorithm for TCP over Wi-Fi PSM

As we have seen in Section 6.1, the performance of TCP over *Wi-Fi PSM* critically depends on the characteristics of the network behind the AP. Therefore, our goal in this section is to design an adaptive algorithm running in a Wi-Fi station, that will tune the operation of the Wi-Fi power saving protocol according to the botleneck bandwidth experienced by a TCP connection. Hereafter, we refer to this algorithm as the *Bottleneck Aware - Triggering Algorithm* (BA-TA).

We have seen in Section 6.1 that the default trigger interval used by *Wi-Fi PSM* (100ms), can severely degrade the performance of a TCP connection, specially as the capacity of the bottleneck increases. Therefore, the intuition behind BA-TA is to adapt this trigger interval in the following way. If the connection's bottleneck has a *small* bandwidth, the Wi-Fi station should use large trigger intervals which do not decrease performance and are energy efficient. On the other hand, if the connection's bottleneck has a *large* bandwidth, the Wi-Fi station should use short trigger intervals (and potentially switch to active mode), since this is the most efficient configuration in this case.

We establish the following constraints in the design of BA-TA:

- The algorithm has to run in the Wi-Fi station without any support from the Access Point, and should not require modifications to existing standards. This will allow a Wi-Fi station to immediately use BA-TA in currently deployed Wi-Fi networks.
- The algorithm should run entirely at Layer two. This will allow to easily re-use BA-TA in any device with a Wi-Fi interface.

BA-TA is composed of two independent modules. A first module that *estimates* the peak rate of the TCP connection's bottleneck, and a second module that given the previous estimation selects an appropriate trigger interval in the power saving protocol. These two modules are described next.

6.2.1 A bottleneck bandwidth estimation algorithm

The goal of this algorithm is to estimate in a Wi-Fi station in power save mode, the *peak* bandwidth of a TCP connection's bottleneck, e.g. 1 Mbps and 16 Mbps in our *Slow DSL* and *Fast DSL* scenarios.

The intuition behind our peak bandwidth estimation algorithm is described in Figure 6.8. This figure depicts the occupancy of the bottleneck line behind the AP, e.g. the DSL line in Figure 6.1, between two (not necessarily consecutive) trigger frames sent by the station in power save mode. Notice that when

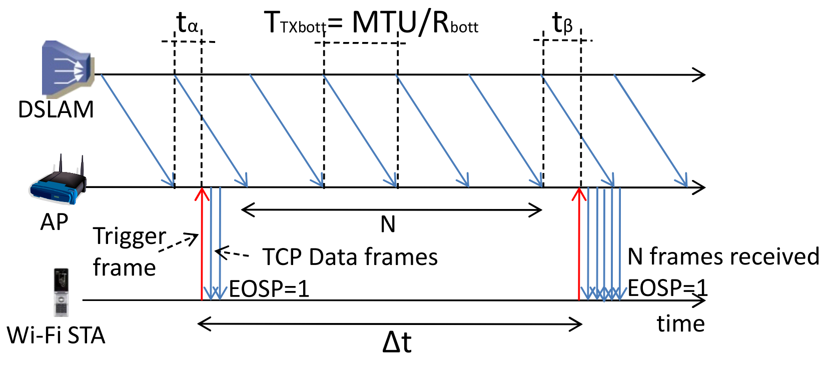


Figure 6.8: Intuition behind the Peak Bandwidth Estimation Algorithm.

a service period completes[1], the Wi-Fi station knows that the buffer in the AP is empty. Therefore, the station can use the amount of data received between these two trigger frames in order to estimate the botleneck's peak bandwidth in the following way. Consider the botleneck line to be always busy between the two triggers sent by the station. We have in this case that $N \times T_{TX_{bott}} = \Delta_t + t_\alpha - t_\beta$, where N is the number of frames received by the station between the two triggers, and $T_{TX_{bott}}$, t_α and t_β are defined in Figure 6.8. Notice that $0 \leq t_\alpha, t_\beta \leq T_{TX_{bott}}$.

Considering now that $T_{TX_{bott}} = \frac{MTU}{R_{bott}}$, where MTU is the size in bits of a packet transmitted over the bottleneck link and R_{bott} is the bottleneck peak transmission rate that we want to estimate, the following lower and upper bounds on R_{bott} can be derived:

$$\begin{cases} T_{TX_{bott}} \geq \frac{\Delta_t}{N+1} \rightarrow R_{bott} \leq \frac{(N+1)MTU}{\Delta_t} \\ T_{TX_{bott}} \leq \frac{\Delta_t}{N-1} \rightarrow R_{bott} \geq \frac{(N-1)MTU}{\Delta_t} \end{cases} \tag{6.7}$$

Notice that the upper bound on R_{bott} only holds if the bottleneck line is all the time busy between the two triggers sent by the station. However, the lower bound on R_{bott} holds true even if the bottleneck is not always busy.

Thus, our peak rate estimation algorithm works in the following way. When a service period completes and more than T_{update} seconds have past from the last algorithm update, BA-TA executes the procedure described in Algorithm 6.1, which records a *lower bound* estimation on the bottleneck's peak rate as described in Equation 6.7. In addition, in order to get as close as possible to the true R_{bott}, Algorithm 6.1 continuously updates the bottleneck rate estimation to the highest observed R_{bott} lower bound (line 7 in Algorithm 6.1).

[1]In U-APSD a service period completes when the AP sets the EOSP bit to 1 in a transmitted frame, indicating to the station that it has no more buffered frames.

Algorithm 6.1: Routine executed when receiving EOSP=1, if $t_{now} > t\_last_{peak} + T_{update}$.

```
1  – Variable definitions
2  $rcvd\_data_{peak}$ ← Amount of data (bits) received since last update
3  $t\_last_{peak}$ ← Time of last update
4  $first\_pkt\_size$ ← Size of the first packet received in the last interval
5  – estimate_peak_rate()
6  $lower\_bound \leftarrow \frac{rcvd\_data_{peak} - first\_pkt\_size}{t_{now} - t\_last_{peak}}$
7  $peak\_rate \leftarrow \max\{peak\_rate, lower\_bound\}$
8  if $!ActiveMode$ then
9      $t\_last_{peak} \leftarrow t_{now}$
10     $rcvd\_data_{peak} \leftarrow 0$
```

There is an extra issue to be considered in Algorithm 6.1. If, as it will be explained in the next section, the Wi-Fi station switches from power save mode to active mode, then the AP will start transmitting the station's remaining buffered packets at the rate of the Wi-Fi network, which can be above the bottleneck rate. In order to avoid overestimating the bottleneck rate in this case, our peak rate estimation algorithm always uses as previous reference a time where the Wi-Fi station was in power save mode, as can be seen in line 9 in Algorithm 6.1.

6.2.2 A trigger interval adaptation algorithm

We now describe how BA-TA controls the trigger interval used by a Wi-Fi station in power save mode. Our trigger adaptation algorithm is essentially a *proportional controller* (102), that controls the station's trigger interval in order to stabilize the connection's throughput around a configurable operation point. This operation point is an input to the algorithm, specified in terms of the desired ratio between the connection's throughput and the bottleneck peak rate, i.e. $0 < ratio_{min} < 1$. Our detailed algorithm is described in Algorithm 6.2, and is summarized next.

A Wi-Fi station implementing BA-TA maintains two estimates: i) a $peak\_rate$ estimate provided by our bottleneck bandwidth estimation algorithm, and ii) an estimate of the instantaneous connection throughput, $instant\_thr$. This estimate is updated using an Exponentially Weighted Moving Average (EWMA) filter with weighting factor α (line 15 in Algorithm 6.2).

A BA-TA station executes Algorithm 6.2 at regular intervals (every T_{update}), in order to update the $peak\_rate$[1] and $instant\_thr$ estimates. In addition, after $count_{max}$ consecutive estimation updates, BA-TA evaluates whether the currently used trigger interval is appropriate. Notice that the configurable

[1]Notice that the $peak\_rate$ estimate is only updated here when the station is in active mode, because in this case the station does not generate triggers and does not receive frames with EOSP=1.

$count_{max}$ parameter enables BA-TA to use different intervals to update the throughput estimates, and the used trigger interval.

In order to update the used trigger interval, BA-TA uses a proportional law. For this purpose, it computes $ratio = \frac{instant\_thr}{peak\_rate}$ as the currently used fraction of bottleneck bandwidth (line 16), and compares it to the input value $ratio_{min}$. Specifically, BA-TA updates the used trigger interval in the following way (line 27):

$$interval(n+1) = (1 + G(ratio - ratio_{min}))interval(n) \tag{6.8}$$

Where G is the gain used in the control law, and $ratio - ratio_{min}$ can be understood as the current error incurred by the controller. Notice in Equation 6.8 that when $ratio$ is below $ratio_{min}$, the trigger interval decreases, which reduces the RTT experienced by the TCP connection and should improve throughput. In addition, when $ratio$ is above $ratio_{min}$ the trigger interval increases, which is more energy efficient. In the Appendix at the end of the chapter it is formally shown that this controller converges with no steady state error when $0 < G < \frac{2}{ratio_{min}}$. In addition, the trigger interval is only allowed to vary between a minimum and maximum trigger intervals, int_{min} and int_{max} (line 38), which are also inputs to the algorithm that can be configured according to the characteristics of each particular Wi-Fi chipset.

Having described the basic operation of BA-TA, there are several issues that have to be considered in a practical implementation. The first of these issues, is how to appropriately switch the station between active mode and power save mode. Notice that as seen in Section 6.1, being in active mode can potentially result in a big energy waste, therefore BA-TA should configure a station in active mode only when there is a clear throughput gain in doing so. BA-TA uses the following heuristic for this purpose:

- *Enter active mode*: If the interval selected in Equation 6.8 is below the minimum allowed interval (int_{min}), and the average ratio increase experienced when the station is in active mode, $\hat{\Delta}_{ratio_{AM}}$, is significantly[1] above the average ratio increase when the station is in power save mode, $\hat{\Delta}_{ratio_{PS}}$, (line 32).

- *Leave active mode*: Under two conditions. First, if $ratio$ is below $ratio_{min}$, but being in active mode is not effective, i.e. $ratio$ does not increase in at least a minimum amount (γ) (line 36). Second, when $ratio$ is above $ratio_{min}$, i.e. we have achieved the desired throughput, and the station is not saturating the network, i.e. $util_{last} \leq util_{min}$ (line 29). Recall from Section 6.1

[1]We force $\hat{\Delta}_{ratio_{AM}} > \hat{\Delta}_{ratio_{PS}} + |\hat{\Delta}_{ratio_{PS}}|$ as a heuristic way to ensure that switching to active mode is indeed significantly better than operating in power save mode (line 33).

Algorithm 6.2: BA-TA executed every T_{update}.

```
– Variable definitions
$ratio_{min} \leftarrow$ Desired operation point
$int_{min}, int_{max} \leftarrow$ Minimum and maximum trigger intervals
$count_{max} \leftarrow$ Update interval
$N_{max} \leftarrow$ Maximum number of attemtps
$G \leftarrow$ Controller gain
$\alpha \leftarrow$ Weights used in the EWMA averages
$\gamma \leftarrow$ Comparison margin on $\Delta_{ratio}$
$rcvd\_data_{int} \leftarrow$ Amount of data (bits) received since last update
$util_{min} \leftarrow$ Minimum utilization to remain in Active Mode
– update_trigger_interval()
if $ActiveMode$ then
    $util_{last} \leftarrow \frac{TX\_time+RX\_time+Backoff\_Time}{T_{update}}$
    if $t_{now} > t\_last_{peak} + T_{update}$ then
        estimate_peak_rate
$instant\_thr \leftarrow \alpha \times instant\_thr + (1-\alpha)\frac{rcvd\_data_{int}}{t - t\_last_{int}}$
$ratio \leftarrow \min\{\frac{instant\_thr}{peak\_rate}, 1\}$
if $peak\_rate, instant\_thr > 0$ and $n\_trigs < n\_trig_{max}$ then
    $count \leftarrow count + 1$
    if $ratio - ratio_{last} < -2\gamma$ then
        $count \leftarrow count_{max}$
    if $ActiveMode$ then
        $\hat{\Delta}_{ratio_{AM}} \leftarrow \alpha\hat{\Delta}_{ratio_{AM}} + (1-\alpha)\Delta_{ratio}$
    else
        $\hat{\Delta}_{ratio_{PS}} \leftarrow \alpha\hat{\Delta}_{ratio_{PS}} + (1-\alpha)\Delta_{ratio}$
    if $count = count_{max}$ then
        $count \leftarrow 0, \Delta_{ratio} \leftarrow ratio - ratio_{last}, ratio_{last} \leftarrow ratio$
        $interval \leftarrow (1 + G(ratio - ratio_{min}))interval$
        if $ratio \geq ratio_{min}$ then
            if $ActiveMode = true$ and $util_{last} \leq util_{min}$ then
                Leave active mode
        else
            if $ActiveMode = false$ and $interval < int_{min}$ and $rcvd\_data_{int} > 0$ then
                if $\hat{\Delta}_{ratio_{AM}} > \hat{\Delta}_{ratio_{PS}} + |\hat{\Delta}_{ratio_{PS}}|$ or $n_{AM} = N_{max} - 1$ then
                    Enter active mode
                $n_{AM} \leftarrow (n_{AM} + 1) \mod N_{max}$
            if $ActiveMode = true$ and $\Delta_{ratio} < \gamma$ then
                Leave active mode
        $interval \leftarrow \max\{\min\{interval, int_{max}\}, int_{min}\}$
else if $rcvd\_data_{int} = 0$ or $n\_trigs \geq n\_trig_{max}$ then
    $ratio_{last} \leftarrow ratio$
    if $ActiveMode$ then
        Leave active mode
    else
        $interval \leftarrow \min\{interval + step_{min}, int_{max}\}$
else
    $interval \leftarrow int_{max}$
$n\_trigs \leftarrow 0, rcvd\_data_{int} \leftarrow 0, t\_last_{int} \leftarrow t_{now}$
$TX\_time, RX\_time, Backoff\_Time \leftarrow 0$
```

that staying in active mode is effective while the station saturates the Wi-Fi network. Therefore, while in active mode a BA-TA station tracks whether its traffic is saturating the Wi-Fi network by measuring network utilization as the fraction of *used* time during the last T_{update} period (line 12). For further details on how to measure network utilization in a Wi-Fi station, the interested reader is referred to (100).

The $\hat{\Delta}_{ratio_{AM}}$ and $\hat{\Delta}_{ratio_{PS}}$ estimates are also obtained with an EWMA filter of weighting factor α, and allow BA-TA to measure the effect that being in active mode or in power save mode has on the experienced throughput (lines 21 to 24). Notice however, that the previous estimates may contain innacuracies. For instance, if while being in active mode TCP suffers a loss and decreases its congestion window, BA-TA will observe a decrease in throughput and may wrongly believe that being in active mode is not efficient. Thus, in order to avoid BA-TA from getting permanently stalled in a sub-optimal configuration, a switch to active mode is always allowed after N_{max} refrained attempts (line 33). In addition, if a sudden[1] decrease in $ratio$ is detected, BA-TA quickly updates the used trigger interval (line 19).

Finally, another issue to consider in BA-TA is the fact that certain applications, like Web, may simply not offer enough load to achieve the desired bottleneck link utilization, i.e. maintain $ratio$ above $ratio_{min}$. The previously described logic would in this case drive a Wi-Fi station to use aggressive trigger intervals, or even switch to active mode, which is undesirable from an energy point of view. In order to prevent the previous from happening, BA-TA limits the maximum number of triggers that a station can send to the AP without receiving any data in return (n_trig_{max}). If that limit is surpassed, BA-TA considers that the used trigger interval is too small and immediately increases it by a pre-configured amount ($step_{min}$) (lines 39 to 44).

6.3 BA-TA Performance Evaluation

In this section we evaluate the performance of BA-TA and study the effect of its configurable parameters. This section is divided in three sub-sections: i) a first sub-section illustrating the basic dynamics of BA-TA, ii) a second sub-section studying how to configure the different BA-TA parameters, and iii) a final sub-section that will extensively evaluate the performance of BA-TA compared to the algorithms studied in Section 6.1. For the purpose of this evaluation, we consider only the *Fast Wi-Fi* configuration defined in Section 6.1. This configuration poses a bigger challenge to BA-TA because, as previously seen, it

[1]Defined as a decrease in $ratio$ above 2γ.

delivers the best energy efficiency, but results in the highest throughput degradation when used with *Wi-Fi PSM*.

Unless otherwise stated, BA-TA is configured in the following way. The maximum and minimum trigger intervals are set to $int_{max} = 100ms$, which is equivalent to the operation of *Wi-Fi PSM*, and $int_{min} = 20ms$, which is a trigger interval commonly used by current Wi-Fi chipsets with Voice traffic (47). The update interval, T_{update}, is set to $100ms$, which is a convenient value that allows to synchronize the operation of BA-TA with other periodic operations done by a station, like receiving the Beacon frame. The weight α used in the EWMA filters is empirically set to $\alpha = 0.9$, and $util_{min}$ is set to 0.8 in order to keep a station in Active Mode only if it can saturate the channel. Finally, The parameter γ is set to $\gamma = 0.1$, which results in BA-TA requiring a 10% increase in $ratio$ when being in active mode (line 36 in Algorithm 6.2), and forcing a sudden interval update if $ratio$ decreases by more than a 20% (line 19 in Algorithm 6.2).

The effect of the rest of BA-TA parameters, i.e. $ratio_{min}$, the controller gain G, the update interval $count_{max}$, the parameter N_{max}, and the parameters n_trig_{max} and $step_{min}$, will be studied in this section.

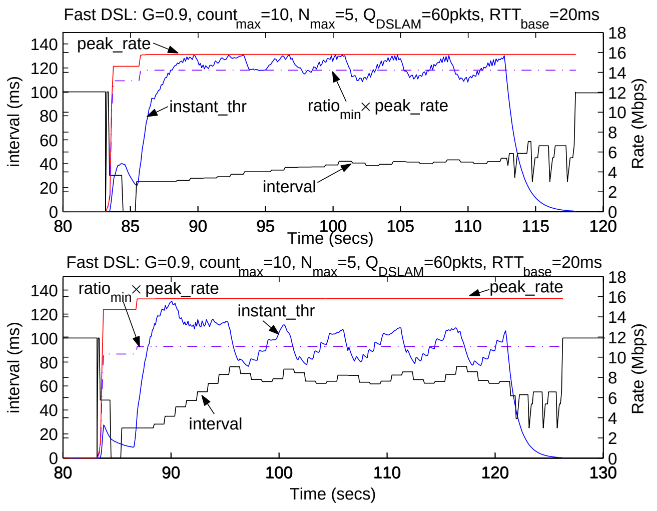
6.3.1 Basic dynamics of BA-TA

Figures 6.9 and 6.10 depict the dynamics of BA-TA when a Wi-Fi station using BA-TA retrieves a 50MB File from the Internet. We start our analysis by looking at Figure 6.9 that illustrates the dynamics of BA-TA in the *Fast DSL* scenario with $ratio_{min} = 0.9$ (upper graph) and $ratio_{min} = 0.7$ (lower graph). The value of the rest of configuration parameters is included on the top of each figure. Notice that a double y-axis is used in the figure, where the left y-axis plots the trigger interval used by BA-TA while retrieving the File (black line), and the right y-axis plots the estimated bottleneck rate (red line) and the instantaneous throughput (blue line) estimated by BA-TA.

We can see in Figure 6.9 how BA-TA operates with an interval of 100ms until the File Transfer begins. From that point on, BA-TA starts adjusting the trigger interval (black line) in order to drive the obtained throughput (blue line) around the desired utilization level, i.e. $ratio_{min} \times peak\_rate$ (purple dashed line). After a transient period, BA-TA converges around the configured throughput, requiring, as expected, a smaller trigger interval for $ratio_{min} = 0.9$ (upper graph) than for $ratio_{min} = 0.7$ (lower graph). After the File Transfer completes, BA-TA switches off and the station returns to wake up only at every Beacon frame.

The upper part of Figure 6.10 illustrates now how BA-TA behaves in the *Wi-Fi Bottleneck* scenario. We can see that after an initial transient period, TCP manages to saturate the Wi-Fi network and therefore

BA-TA configures the station to operate all the time in active mode, which as seen in Section 6.1 is the most efficient operation mode in this case.

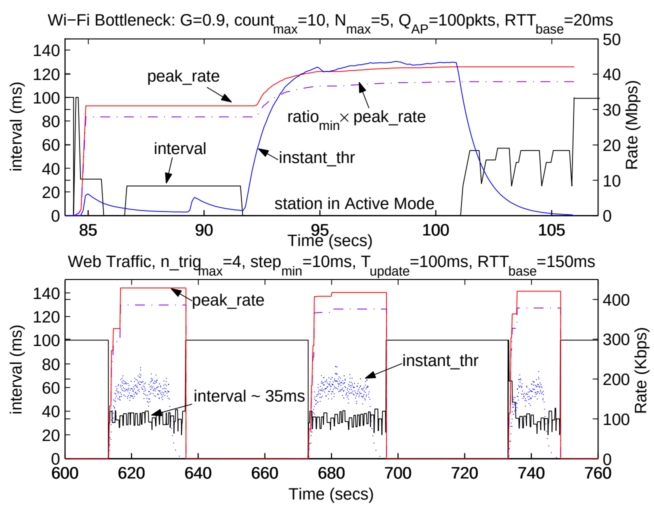


(a) *Fast DSL*, $ratio_{min} = 0.9$ (upper graph), $ratio_{min} = 0.7$ (lower graph).

Figure 6.9: Basic BA-TA Dynamics. The figures represent the variation over time of BA-TA used interval (black line), the $instant\_thr$ estimation (blue line), and the $peak\_rate$ estimation (red line).

For the sake of space we do not report the dynamics of BA-TA in the *Slow DSL* scenario, where BA-TA delivered the bottleneck throughput using the maximum trigger interval, i.e. $int_{max} = 100ms$. Instead, the lower part of Figure 6.10 depicts the dynamics of BA-TA with Web traffic. Web traffic is modeled using HTTP 1.1 where the Wi-Fi station establishes a persistent TCP connection with the Web server in Figure 6.1, and retrieves all the objects in a Web page in a sequential way. A Web page is modeled according to the statistics provided in (91).

The figure depicts the interval used by BA-TA while retrieving three Web pages, where the value of RTT_{base} between the AP and the Web server in Figure 6.1 is set to $150ms$, i.e. there is a latency of at least $150ms$ between two consecutive Web page objects retrieved from the Web server. Interestingly, the interval used by BA-TA in this case converges to a value around $35ms$. The reason is the following. We configured in this scenario $n\_trig_{max} = 4$ and $step_{min} = 10ms$, recall now that if $\frac{T_{update}}{n\_trig_{max}} \leq interval \leq \frac{T_{update}}{n\_trig_{max}-1}$, and the application does not offer enough load, e.g. Web, BA-TA may receive n_trig_{max} empty triggers within a T_{update} interval and will increase the used trigger interval by $step_{min}$ (line 44 in Algorithm 6.2). Therefore, the used interval must converge to a value around $\frac{T_{update}}{n\_trig_{max}}$ and $\frac{T_{update}}{n\_trig_{max}-1} + step_{min}$. The previous fact can be used as a configuration guideline for the parameters



(a) *Wi-Fi Bottleneck* (upper graph), Web traffic (lower graph).

Figure 6.10: Basic BA-TA Dynamics. The figures represent the variation over time of BA-TA used interval (black line), the $instant\_thr$ estimation (blue line), and the $peak\_rate$ estimation (red line).

n_trig_{max} and $step_{min}$, which we hereafter configure as $n\_trig_{max} = 4$ and $step_{min} = 10ms$.

6.3.2 Effect of BA-TA parameters

We divide the study of the effect of the BA-TA parameters in two different parts. First, we benchmark the effect of the BA-TA parameters when a station retrieves a File from the Internet. Second, we study the effect of these parameters in a scenario where multiple flows share the bottleneck link.

6.3.2.1 Effects on a single flow

Figure 6.11 depicts the results of a set of experiments, where the evaluation methodology described in Section 6.1 is applied, while varying different BA-TA parameters. In particular, the following three experiments are shown:

- *Exp. 1*: Configure $G = 0.1, 0.5, 0.9$, which according to the analysis in the Appendix at the end of this chapter should guarantee monotonic convergence, while fixing $count_{max} = 10$ and $N_{max} = 5$ (Figure 6.11(a)).
- *Exp. 2*: Configure $count_{max} = 5, 10, 20$ while fixing $G = 0.5$, and $N_{max} = 5$ (Figure 6.11(b)).
- *Exp. 3*: Configure $N_{max} = 2, 5, 8$ while fixing $G = 0.5$, and $count_{max} = 10$ (Figure 6.11(c)).

For the sake of space Figure 6.11 only depicts the results of our *Buffer Experiment* for the *Fast DSL* (upper graph) and *Wi-Fi bottleneck* (lower graph) scenarios. In addition, the throughput (solid lines) and energy (dashed lines) performance depicted in the graph is normalized to the maximum throughput and minimum energy achieved by the algorithms considered in Section 6.1 for the same experiment, i.e. $\hat{Thr} = \frac{Thr_{BA-TA}}{max\{Thr_{Active}, Thr_{PSM}\}}$, $\hat{Enrg} = \frac{min\{Enrg_{Active}, Enrg_{PSM}\}}{Enrg_{BATA}}$. Thus, ideally BA-TA would provide a performance for both indexes with a value as close as possible to 1, or bigger.

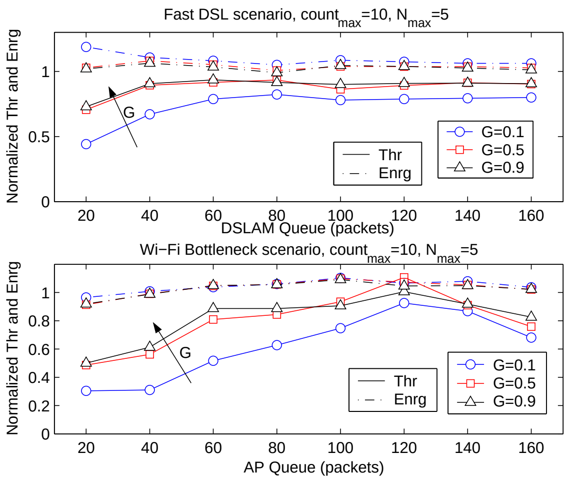
The following configuration guidelines are identified from Figure 6.11. First, a too small value of the controller gain, $G < 0.5$, can degrade throughput performance because the transient period takes too long in this case. Second, big values of N_{max} or $count_{max}$ slightly degrade the throughput achieved by BA-TA in the *Wi-Fi Bottleneck* scenario. The reason is that BA-TA often switches to active mode in this case, and if it gets stalled in a suboptimal configuration, due to a misestimation of $\hat{\Delta}_{ratio_{AM}}$ and/or $\hat{\Delta}_{ratio_{PS}}$, a time equal to $N_{max} \times count_{max} \times T_{update}$ seconds is required to recover. Therefore, smaller values of N_{max} and $count_{max}$ allow BA-TA to recover earlier in these cases.

6.3.2.2 Effects on sharing scenarios

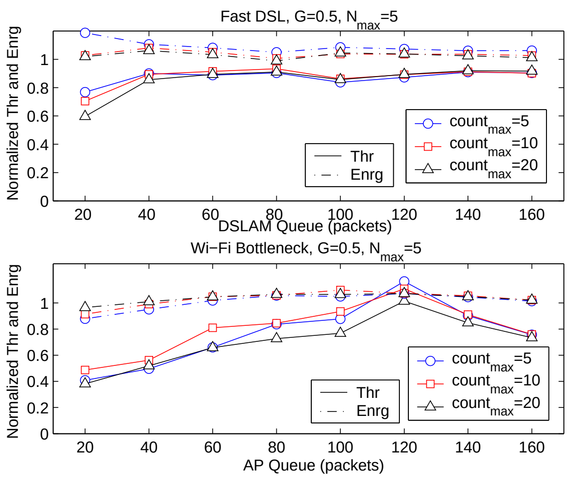
We evaluate now the performance of BA-TA in scenarios where a Wi-Fi station running BA-TA shares the bottleneck link with another station, which we configure to be in active mode since this should be the most common situation in practice. The goal of this section is to show that a station running BA-TA is able to share a bottleck link with an active mode station, without any station starving.

Figure 6.12(a) depicts an example where a station using BA-TA and a station in active mode share, between 230 and 300 seconds, the DSL link in the *Fast DSL* scenario. Notice that in this case, BA-TA's proportional law drives the power saving station towards using small trigger intervals or switching to active mode, because the power saving station's throughput can not reach the configured fraction of the bottleneck peak bandwidth (because the link is shared). However, being in active mode is not efficient in this case, i.e. $\hat{\Delta}_{ratio_{AM}}$ is not significantly bigger than $\hat{\Delta}_{ratio_{PS}}$, and BA-TA switches the power saving station between active mode and power save mode. Notice that it is possible to control in these sharing scenarios how often a power saving station switches to active mode by adjusting the parameter N_{max}, because BA-TA always enters active mode after N_{max} refrained attempts. Finally, when the flow from the station in active mode completes, after 300 seconds, BA-TA quickly returns to its normal operation.

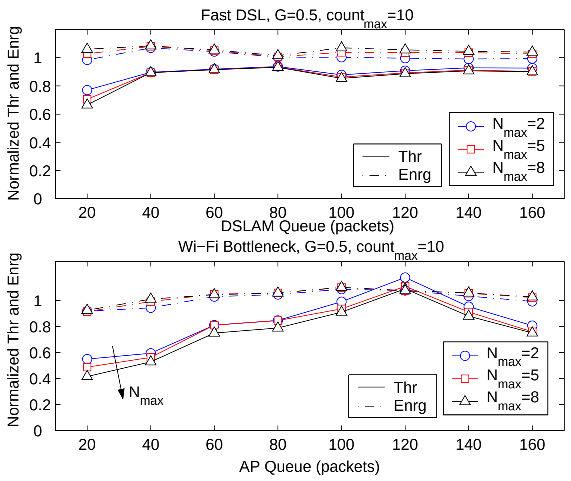
Figure 6.12(b) depicts, between 300 seconds and 350 seconds, an effect that occurs when a station in power save mode receives a flow through the DSL link, while at the same time another Wi-Fi station in active mode retrieves a File from an Ethernet network behind the AP. Interestingly, if the power saving station uses *Wi-Fi PSM* (purple line), its throughput reduces to almost zero while the station in active



(a) *Exp. 1*: Effect of G.

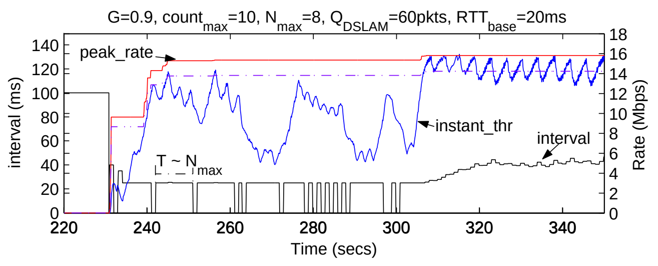


(b) *Exp. 2*: Effect of $count_{max}$.

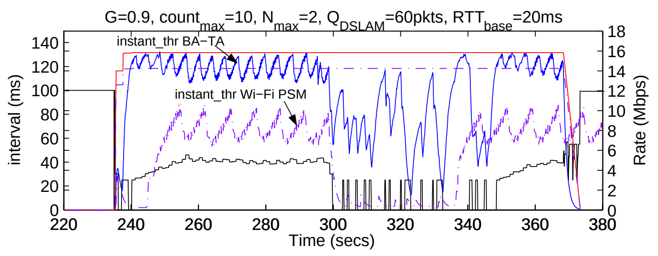


(c) *Exp. 3*: Effect of N_{max}.

Figure 6.11: Effect of BA-TA parameters. The upper part of each figure depicts the *Fast DSL* scenario, and the lower part of each figure depicts the *Wi-Fi Bottleneck* scenario. In addition, notice that the depicted throughput and energy metrics are normalized.



(a) A Wi-Fi PSM flow and an Active Mode flow through a 16Mbps DSL link with $N_{max} = 8$.



(b) A Wi-Fi PSM flow through a 16Mbps DSL link and an Active Mode flow through 100Mbps Ethernet.

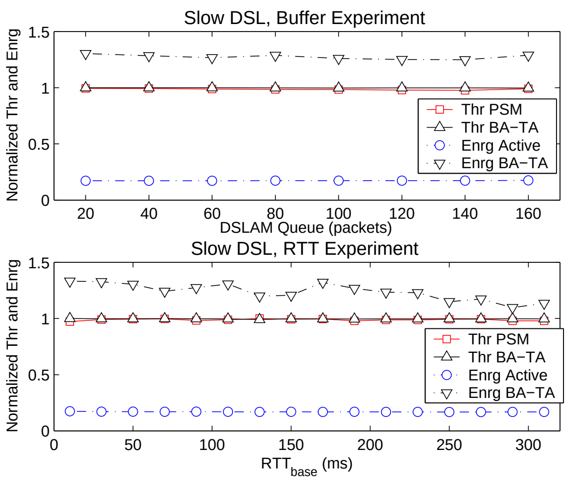
Figure 6.12: BA-TA dynamics with multiple flows.

mode is retrieving its File. This unfairness occurs because the flow from the station in active mode saturates the Wi-Fi network. Thus, when the station in power save mode wakes up and sends a trigger to the AP to retrieve its buffered frames, the AP dequeues a frame from the power saving buffer but often can not transmit it because the AP's transmission queues are full with packets from the station in active mode. This unfairness could be avoided by deploying enhanced scheduling algorithms in the AP, however these algorithms are not generally implemented in existing APs. Instead, notice in Figure 6.12(b) that BA-TA avoids the previous unfairness regardless of the scheduling strategy used in the AP, by maintaining the power saving station most of the time in active mode while the other flow is active.

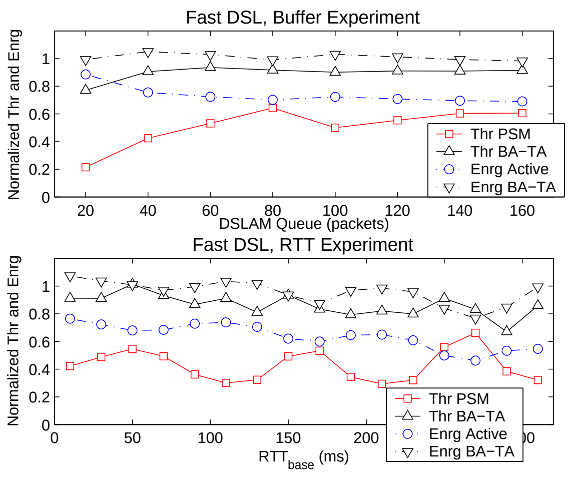
6.3.3 BA-TA extensive evaluation

In this section we report the performance of BA-TA using the same evaluation setup described in Section 6.1. Based on the insight obtained in the previous section we configure BA-TA with the following parameters, $G = 0.9$, $count_{max} = 10$, $N_{max} = 3$, $n\_trig_{max} = 4$ and $step_{min} = 10ms$.

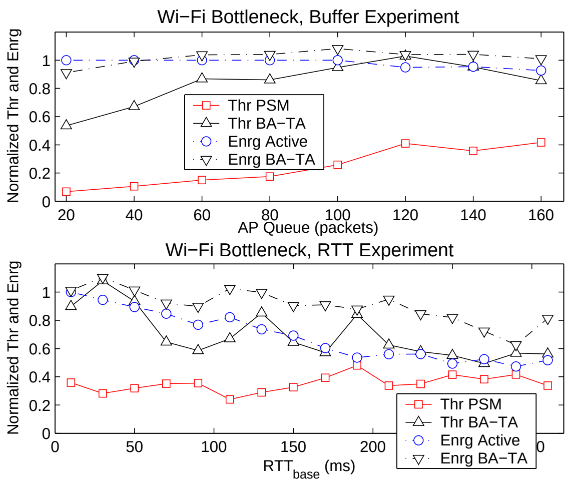
Figure 6.13 reports the performance delivered by BA-TA in the *Buffer* and *RTT* experiments defined in Section 6.1 for the *Slow DSL*, *Fast DSL* and *Wi-Fi Bottleneck* scenarios, where we have used the



(a) Slow DSL Scenario.



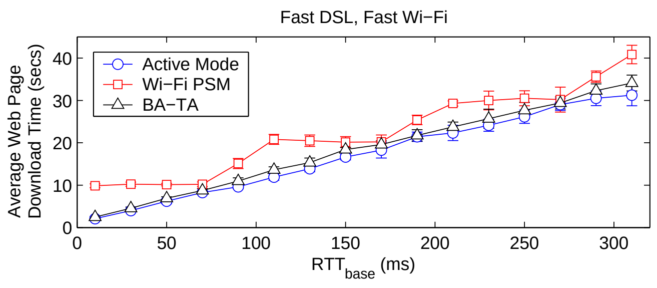
(b) Fast DSL Scenario.



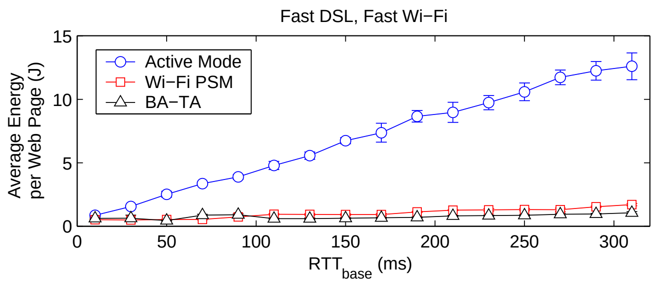
(c) Wi-Fi Bottleneck Scenario.

Figure 6.13: BA-TA Extensive Evaluation. The upper part of each graph depicts the results for the Buffer Experiment, and the lower part of each graph depicts the results for the RTT Experiment. In addition, notice that the depicted throughput and energy metrics are normalized.

previously defined normalized throughput and energy metrics, i.e. $\hat{Thr}$ and $\hat{Enrg}$. In addition, to put BA-TA's performance in perspective, the graphs also depict the normalized throughput obtained by the *Wi-Fi PSM* algorithm, and the normalized energy obtained by the *Active Mode* algorithm. As observed in Figure 6.13, BA-TA delivers a very good performance, i.e. both normalized metrics are close or above 1, for all considered scenarios and parameter range. Only when RTT_{base} increases significantly or very small buffers are considered in the *Wi-Fi Bottleneck* scenario, BA-TA struggles to match the *Active Mode* throughput, but still clearly outperforms *Wi-Fi PSM* in terms of throughput and *Active Mode* in terms of energy. Notice that with a big RTT_{base}, the parameters n_trig_{max} and $step_{min}$ can be adjusted to trade off throughput and energy, as explained in the case of Web.



(a) Average Web Page download time.



(b) Average Energy per Web Page.

Figure 6.14: BA-TA performance with Web Traffic.

Finally, although we have focused in this paper in the performance of long lived TCP connections, we believe it is important to illustrate how BA-TA behaves with short TCP flows, like Web traffic. For this purpose, Figures 6.14(a) and 6.14(b) illustrate respectively the average Web page download time and the average energy consumption per Web page, when a Wi-Fi station uses the different algorithms under study. Since Web traffic is mostly affected by latency, we only report the results obtained in the RTT experiment for our *Fast DSL* scenario. We can see in Figure 6.14(a) and 6.14(b) how, thanks to the BA-TA dynamics for Web traffic previously explained, that drive the trigger interval to converge to a value around $\frac{T_{update}}{n\_trig_{max}}$ and $\frac{T_{update}}{n\_trig_{max}-1} + step_{min}$, *BA-TA* delivers Web page download times similar to the ones of the *Active Mode* algorithm, at an energy cost similar to the one of *Wi-Fi PSM*.

6.4 Conclusions

In this chapter we have studied, by means of analysis and simulation, the effect that current Wi-Fi power saving protocols have on the throughput/energy trade off experienced by long lived TCP traffic. Our study unveils that the efficiency of Wi-Fi power saving protocols critically depends on the bottleneck bandwidth experienced by a TCP connection.

Based on the obtained insights, we have also designed a novel algorithm, BA-TA, which runs in a Wi-Fi station, does not require any modification to existing Wi-Fi standards, and using only information available at layer two, optimizes the throughput/energy trade off of long and short TCP connections. Finally, a thorough performance evaluation has been presentend that illustrates how BA-TA manages to significantly improve the performance energy trade-off of data traffic over Wi-Fi with respect to alternative approaches in the state of the art.

Appendix 6.A BA-TA convergence analysis

In order to simplify the analysis while capturing the essence of the controller used in BA-TA we abstract the behavior of TCP in the following way. We notice that the RTT experienced by a TCP connection depends on the trigger interval in BA-TA. Therefore, we assume that between interval updates in BA-TA, a long lived TCP connection delivers a throughput that is inversely proportional to the trigger interval used by BA-TA, i.e. $thr(n) = \frac{A}{int(n)}$, where A is assumed to be a constant value in this simplified model (TCP is assumed to converge within a time equal to $T_{update} \times count_{max}$), and n represents the *n-th* update interval of BA-TA. Under this premise, the $ratio$ value computed by BA-TA in the *n-th* interval update can be expressed as:

$$ratio(n) = \frac{thr(n)}{peak\_rate \times ratio_{min}} = \frac{B}{int(n)}$$

Where B is again a constant. Now recall from Equation 6.8 that the error incurred by BA-TA in the *n-th* interval update can be computed in the following way:

$$error(n) = ratio(n) - ratio_{min} = \frac{B}{int(n)} - ratio_{min}$$

We can now see how the proportional controller used in BA-TA evolves in time, noting that:

$$int(n+1) = int(n)(1 + Gerror(n)) = int(n)(1 - Gratio_{min}) + GB$$

Where G is the gain used in the control law. Therefore, it can be proved by induction that:

$$int(n+k) = int(n)(1 - Gratio_{min})^k + GB\sum_{j=0}^{k-1}(1 - Gratio_{min})^j =$$
$$= (int(n) - \frac{B}{ratio_{min}})(1 - Gratio_{min})^k + \frac{B}{ratio_{min}}$$

Therefore, when $k \to \infty$ the selected interval converges as $(1 - Gratio_{min})^k$, where $0 < ratio_{min} < 1$. It is then easy to see that $int(n+k)$ has the following convergence properties as $k \to \infty$:

$$\begin{cases} G \le 0 \to Unstable \\ 0 < G < \frac{1}{ratio_{min}} \to Monotonic\ convergence \\ \frac{1}{ratio_{min}} < G < \frac{2}{ratio_{min}} \to Oscillatory\ convergence \\ G \ge \frac{2}{ratio_{min}} \to Unstable \end{cases}$$

Thus, when $0 < G < \frac{2}{ratio_{min}}$ the controller is stable and the selected interval converges to:

$$\lim_{k\to\infty} int(n+k) = \frac{B}{ratio_{min}}$$

Where, as seen in Figure 6.9, the selected interval logically converges to a value that is inversely proportional to $ratio_{min}$. In addition, when the algorithm converges the steady state error is:

$$\lim_{k\to\infty} error(n+k) = \frac{B}{\frac{B}{ratio_{min}}} - ratio_{min} = 0$$

7

Designing Energy Efficient Access Points with Wi-Fi Direct

In the previous chapters of this thesis we have addressed the challenge of having a Wi-Fi station operate in an energy efficient way while considering a variety of applicantions and protocols. The reason for focusing on a Wi-Fi station stems from the fact that mobile computing devices have traditionally played this role in a Wi-Fi network. However, as explained in Chapter 2, the Wi-Fi Direct technology enables novel device to device use cases by allowing Wi-Fi devices to negotiate the roles of AP and client in order to set up an infrastructure-like network, without the presence of a traditional AP. Therefore achieving an energy efficient operation in Wi-Fi APs will become a critical issue in the near future.

Therefore, our work in this chapter focuses on the AP power saving protocols defined by Wi-Fi Direct. Specifically, we target the case where the Wi-Fi Direct device acting as AP is a battery limited device (e.g. a mobile phone) and offers its connected clients access to an external network (e.g. a cellular network). Figure 7.1 depicts an example of our target scenario. In this context, we design and evaluate algorithms that address the trade-off between the power consumed by the Wi-Fi Direct AP and the performance experienced by its connected clients. The work presented in this chapter has been published in (30).

The rest of the chapter is organized as follows. Section 7.1 provides an overview of the AP power management protocols defined by Wi-Fi Direct. Sections 7.2 and 7.3 design and evaluate, respectively, our proposed Wi-Fi Direct Adaptive Single Presence Period (ASPP) and Adaptive Multiple Presence Periods (AMPP) algorithms. Finally, Section 7.4 concludes this chapter.

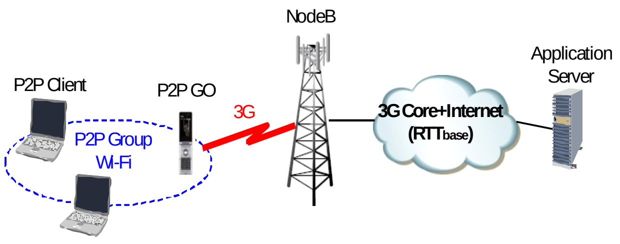


Figure 7.1: Our target scenario. A mobile phone sharing access to a 3G network with a set of connected devices using Wi-Fi Direct.

7.1 AP power management in Wi-Fi Direct

Wi-Fi Direct devices, named Peer to Peer (P2P) devices (11), must be able to act both as a Wi-Fi AP or as a Wi-Fi Client. In particular Wi-Fi Direct defines the concept of a P2P Group, where a P2P Group Owner (P2P GO) acts as an AP for a set of connected P2P Clients. There are two possible ways to decide which P2P device will act as P2P Group Owner:

1. A P2P device autonomously initiates a P2P Group.

2. Two P2P devices run a negotiation protocol after discovering each other.

Once the P2P Group is established new P2P devices can discover and join the group using active or passive scanning mechanisms like the ones used in traditional Wi-Fi networks. Acting as P2P Group Owner provides certain advantages. For instance, a P2P Group Owner is allowed to cross-connect a P2P Group with an external network, e.g. a cellular network if the P2P Group Owner has a 3G interface. However, becoming a P2P Group Owner requires performing certain functions (e.g. beaconing, forwarding) that will result in a higher power consumption than the one of a P2P Client, which can benefit from the power saving protocols already defined in IEEE 802.11. In order to address the power consumption imbalance between a P2P Group Owner and a P2P Client, and to allow battery powered devices to efficiently operate as P2P Group Owners, the Wi-Fi Direct specification defines two new protocols that can be used by a P2P Group Owner: i) the *Opportunistic Power Save* protocol, and ii) the *Notice of Absence* (NoA) protocol.

7.1.1 Opportunistic Power Save protocol

The Opportunistic Power Save protocol (OPS) allows a P2P Group Owner to *opportunistically* save power when all its associated clients are sleeping. This protocol has a low implementation complexity but, given the fact that the P2P Group Owner can only save power when all its clients are sleeping, the power savings that can be achieved by the P2P Group Owner are limited. OPS is based on the design of the traditional power save mode used by clients in an infrastructure network. A P2P Group Owner can save power by defining a limited *presence period* after every Beacon transmission, known as *CTWindow*, where P2P Clients are allowed to transmit. If at the end of the CTWindow all associated P2P Clients are sleeping, the P2P Group Owner is allowed to sleep until the next Beacon time. However, if any P2P Client stays in active mode at the end of the CTWindow the P2P Group Owner is forced to remain awake until the next Beacon time. The operation of the Opportunistic Power Save protocol is depicted in Figure 7.2(a).

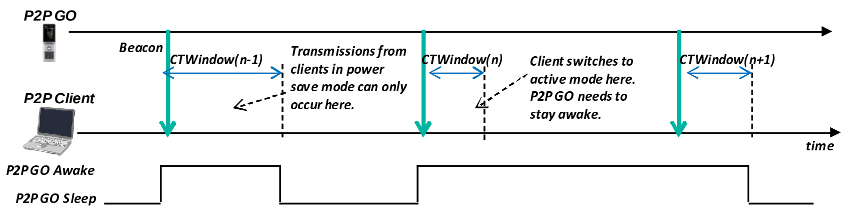
7.1.2 Notice of Absence protocol

Unlike Opportunistic Power Save, the Notice of Absence (NoA) protocol can be used by a P2P Group Owner to save power regardless of the power state of its associated clients. The NoA protocol requires a higher implementation complexity than Opportunistic Power Save but delivers to a P2P Group Owner a higher control on its power consumption. The idea behind the NoA protocol is to let a P2P Group Owner advertise a set of *absence periods* where its associated P2P clients are not allowed to transmit. Thus, a P2P Group Owner may sleep during these absence periods in order to save power.

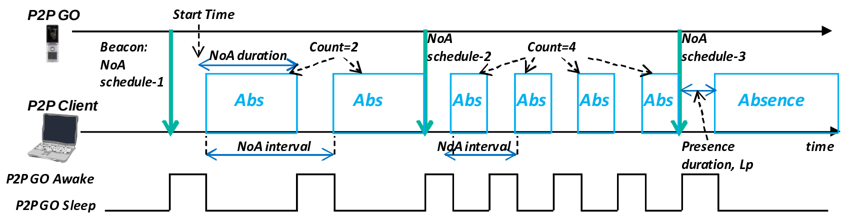
The NoA protocol provides a P2P Group Owner the means to signal a flexible *absence schedule*. In particular, a NoA absence schedule is defined by a 4-tuple: $start\ time$, $duration$, $interval$ and $count$.

- *Start time* defines the start time of the next absence period.
- *Duration* indicates the duration of an absence period in the schedule.
- *Interval* indicates the time between consecutive absence periods.
- *Count* indicates the number of absence period occurrences until the advertised schedule expires. If $count$ is set to 255 the advertised schedule repeats until is explicitely cancelled.

In order to keep complexity low, the Wi-Fi Direct specification allows a P2P Group Owner to advertise at any point in time a single NoA schedule in Beacon frames and Probe Responses. A P2P Group Owner can update the current NoA schedule simply by modifying the correspondent signaling element in Beacon frames and Probe Responses, or can cancel it by omitting the correspondent signaling element. A P2P Client always adheres to the most recently observed NoA schedule advertised by the P2P Group Owner. Finally, the NoA protocol includes a mechanism, the P2P Presence Request/Response handshake, that allows a P2P Client to request a P2P Group Owner to be present at certain intervals. Although not mandatory, such a request mechanism is useful when a P2P Client runs applications that require QoS guarantees, like VoIP. Figure 7.2(b) depicts an exemplary NoA schedule and illustrates the function of each parameter.


P2P GO
P2P Client
Beacon
CTWindow(n-1)
Transmissions from clients in power save mode can only occur here.
CTWindow(n)
Client switches to active mode here. P2P GO needs to stay awake.
CTWindow(n+1)
time
P2P GO Awake
P2P GO Sleep

(a) Example of Opportunistic Power Save operation.



(b) Example of Notice of Absence operation.

Figure 7.2: Example operation of the Wi-Fi Direct power saving protocols for a P2P Group Owner.

In order to allow for product differentiation, the Wi-Fi Direct specification does not define how a P2P Group Owner has to build a Notice of Absence schedule, or how *CTWindow* has to be adjusted in case of Opportunistic Power Save. Therefore, the focus of this paper is to define algorithms that a P2P Group Owner can use to reduce energy consumption while minimizing any effect on the performance of its associated P2P Clients. In particular we designed and evaluated two novel algorithms:

- *Adaptive Single Presence Period* (ASPP) which can be used with both the Opportunistic Power Save and the Notice of Absence protocols.
- *Adaptive Multiple Presence Periods* (AMPP) which can be used only with the Notice of Absence protocol but improves the performance of ASPP.

7.2 ASPP: Adaptive Single Presence Period

In this section we present the design of the *Adaptive Single Presence Period* (ASPP) algorithm. This algorithm is applicable to both power saving protocols described in the previous section, i.e. Opportunistic Power Save and Notice of Absence. Its basic idea is to adaptively adjust the size of the *Presence*

Period that a P2P Group Owner advertises at every Beacon frame, as illustrated in Figure 7.2(a), based on the consideration of the trade-off between service quality of experience and device power saving.

7.2.1 ASPP algorithm design

We start the design of our ASPP algorithm by establishing the architectural constraint that ASPP should operate using only information available at layer two, i.e. running in the Wi-Fi driver of the P2P Group Owner, without requiring any interaction with higher layers or other network interfaces. Note that this design decision generalizes the applicability of the developed algorithm since it can be then applied even if new network interfaces are incorporated at the P2P Group Owner or even if the P2P Group Owner is not directly connected to the external network. An example of the previous case would be for instance a laptop acting as P2P Group Owner where a 3G card is used to provide access to the Internet.

The main challenge to be solved by ASPP is hence how to *dimension* the advertised presence periods, L_P, every Beacon interval, in order to efficiently balance the energy consumed by the P2P Group Owner and the performance experienced by the associated P2P Clients.

We study how to address the previous objective in the context of our scenario of interest depicted in Figure 7.1, which consists of a mobile phone acting as P2P Group Owner and sharing access to a 3G network among its associated P2P Clients. It is fair to assume in this scenario that the bandwidth available in a 3G link will normally be lower than the bandwidth available in the P2P Group. Note that current Wi-Fi networks provide peak rates above 54Mbps (up to 300Mbps with 802.11n), while the majority of deployed 3G networks with HSDPA have peak rates of up to 7.2Mbps in the downlink (127). Thus, if the previous assumption holds, the maximum data rates achieved in the connections established by the P2P Clients connecting through the mobile phone in Figure 7.1 will be limited by the bandwidth available in the 3G network. Therefore, an ideal dynamic algorithm would advertise the smallest presence periods that can deliver in the P2P Group the bandwidth available in the 3G link. In this way a P2P Group Owner would maximize its sleep periods without the P2P Clients noticing any performance degradation.

However, in order to implement this ASPP dynamic algorithm two challenges need to be addressed:

1. The bandwidth available in the 3G link can be highly variable and, given our Layer 2 information only constraint, is not known by the P2P Group Owner.

2. Even if the bandwidth in the 3G link was perfectly known to the P2P Group Owner, applications may not offer enough load to saturate the 3G link. Therefore, a P2P Group Owner should avoid

over-dimensioning its advertised presence periods and instead, size them in order to satisfy the minimum between the application's required data rate, and the bandwidth available in the 3G link.

In order to adress the aforementioned challenges, ASPP will adapt the length of the presence periods based on the amount of traffic flowing between the P2P Group and the 3G network. The assumption here is that TCP connections will saturate in the 3G link, and thus reacting to the amount of traffic flowing between the P2P Group and the 3G network, ASPP will indirectly follow the variations in the available 3G bandwidth. The implications of this assumption will be carefully analyzed in the ASPP algorithm evaluation presented in Section 7.2.2.

Thus, we want to design an algorithm that dimensions presence periods in the following way:

- First, if the bandwidth available in the 3G link increases and the applications offer enough load, the P2P Group Owner will measure an increase of traffic in the P2P Group and will increase L_P accordingly.
- Second, if either the bandwidth available in the 3G link or the load offered by the applications decreases, the P2P Group Owner will measure a decrease of traffic in the P2P Group and will in turn decrease L_P.

The previous behavior can be implemented using the following proportional controller (a similar controller was used in (54) in the context of solar powered APs, or by our BA-TA algorithm introduced in Chapter 6):

$$L_P(n+1) = L_P(n) + K(U_{last} - U_{target})L_P(n) \tag{7.1}$$

Where K is a constant used to trade-off convergence speed and stability, U_{last} is the utilization measured in the P2P Group during the last presence period $L_P(n)$, i.e. $U_{last} \sim \frac{used\ time}{L_P}$, and $0 \leq U_{target} \leq 1$ is the algorithm's target utilization.

It can be observed that the previous controller adjusts L_P according to the measured utilization, U_{last}, in order to maintain future utilizations around U_{target}[1]. For instance, a small value of U_{target} in Equation 7.1 will lead to high presence intervals even when the network utilization is light, which can result in a good traffic performance but in an increased energy consumption in the P2P Group Owner. In the next section we will study how to appropriately set the U_{target} and K parameters. A P2P Group Owner will execute Equation 7.1 prior to Beacon transmissions in order to decide on the presence duration to be advertised.

[1]A formal proof of the convergence for this controller can be obtained re-using the convergence analysis of BA-TA included in the Appendix of Chapter 6.

A critical aspect in the previous controller is the way a P2P Group Owner measures utilization in the P2P Group. This is indeed a non trivial issue in contention based networks like Wi-Fi. For instance, if low priority contention settings are used, a P2P Group Owner would sense less transmissions than if higher priority contention settings are used, although in both cases a P2P Client might have a queue full of packets. Therefore, we propose to compute U_{last} in Equation 7.1 in the following way:

$$U_{last} = \frac{tx\_time + contention\_time}{L_P(n)} \tag{7.2}$$

Where a P2P Group Owner accumulates the duration of transmitted and received frames during a presence period in the variable tx_time. In order to capture the effect of contention, an estimate of the average access delay, which is defined as the time since a packet is first in the transmission queue until it is successfully transmitted, is kept in the variable $Acc\_Del[AC]$ for each Access Category (AC). The $Acc\_Del[AC]$ estimates are updated by means of an Exponentially Weighted Moving Average (EWMA) updated every time the P2P Group Owner transmits a frame through an AC:

$$Acc\_Del[AC](n) = \alpha \cdot Acc\_Del[AC](n-1) + (1-\alpha)Last\_Acc\_Del[AC] \tag{7.3}$$

where we empirically set $\alpha = 0.9$. Thus, for each transmitted or successfully received frame[1], the P2P Group Owner accumulates in the variable $contention\_time$ the correspondent $Access\_Delay[AC]$.

Finally, in addition to the previous steps, presence periods are limited between $L_{P_{max}}$, the size of the Beacon interval, and $L_{P_{min}}$, a minimum configurable by the device manufacturer. Thus, when there is no traffic, small presence periods are advertised achieving a significant energy reduction. As soon as some traffic appears in the network the size of the presence periods is adjusted in order to accomodate it until $L_{P_{max}}$ is reached.

7.2.2 ASPP: Algorithm evaluation

In this section we evaluate the performance of our proposed ASPP algorithm by means of packet level simulations. We divide this evaluation in two stages, a first stage where we illustrate the algorithm dynamics, and a second stage where we extensively evaluate the performance of the algorithm with popular data applications like file transfers and Web traffic.

[1]Notice that a P2P Group Owner can discover the Access Category of a received frame looking at the User Priority field present in the Wi-Fi header.

7.2.2.1 Simulation framework

Our performance evaluation is carried out by means of packet level simulations using OPNET (64). We implemented in OPNET the Opportunistic Power Save and the Notice of Absence protocols defined in (11) and all the relevant Wi-Fi protocols required in Wi-Fi Direct for QoS and power saving, i.e. WMM and WMM-PS (35). Table 7.1 contains all the Wi-Fi related simulation parameters employed in our evaluation.

	Data/Control/Mgmt Rate	**AIFS/CWmin/CWmax/TXOP**
WiFi AC_BK	54/24/1 Mbps	7/31/1023/0ms
WiFi AC_VI	54/24/1 Mbps	2/7/15/3ms

Table 7.1: Wi-Fi Configurations under study.

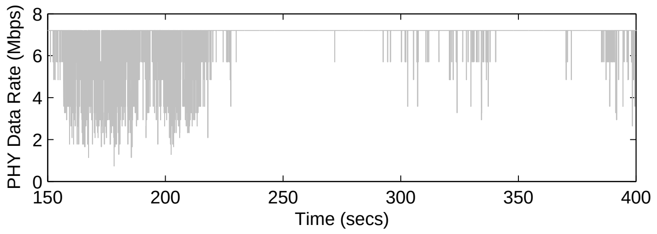
Our simulations reproduce the scenario depicted in Figure 7.1, where in order to model the 3G link we ported to OPNET the 3G simulation framework defined by the Eurane project (122), which simulates a HSDPA link. For the purpose of our evaluation we use three baseline 3G channel models, which capture a wide spectrum of possible 3G channels:

- *Pedestrian-A* channel model which represents a scenario of reduced mobility and *good* radio conditions
- *Typical Urban* channel model which represents a scenario of moderate mobility and *average* radio conditions
- *Vehicular-A* channel model which represents a scenario with high mobility and *poor* radio conditions.

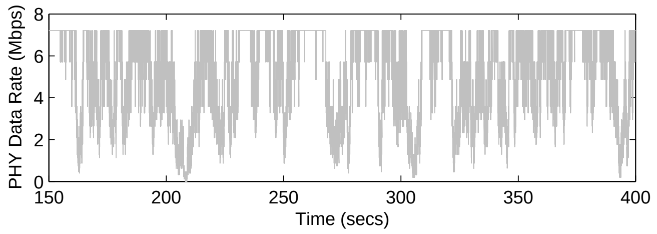
Figure 7.3 depicts an example of how the available PHY data rate varies in time with each of the previous channel models. Table 7.3 contains all the 3G related parameters used in our evaluation. In addition, the core network connecting the NodeB to the application servers in Figure 7.1 is modeled using a node that introduces a configurable delay, hereafter referred to as RTT_{base}.

We consider Web browsing and File Transfers, e.g. Video streaming, as the most relevant applications to be evaluated in our scenario of interest, i.e. a mobile phone providing 3G access through Wi-Fi Direct. TCP New Reno is the transport protocol used in our evaluation.

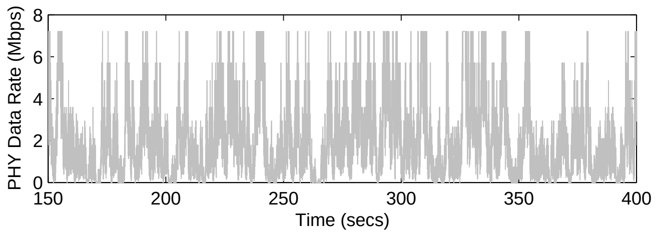
In order to evaluate the energy consumed by the Wi-Fi interface of the P2P Group Owner, we make use of our model that captures the energy consumed by a Wi-Fi chipset. This model consists of four basic states: Sleep, Listen, Reception and Transmission. Energy is computed by integrating the power



(a) Pedestrian-A, 3Km/h, 100m, *good* channel.



(b) Typical Urban, 50Km/h, 250m, *average* channel.



(c) Vehicular-A, 120km/h, 500m, *poor* channel.

Figure 7.3: Sample of PHY rate variation in the HSDPA Channels considered in our evaluation.

that a Wi-Fi device spends in each of the previous states over a certain target time. In our evaluation this target time will be the time to transfer a file or a web page. The power values used were obtained from a known embedded Wi-Fi chipset (23) and are shown in Table 7.2.

Wi-Fi Chipset	**Sleep**	**Listen**	**Rx**	**Tx**
Power (mW)	0.3	432	432	640

Table 7.2: Power Consumption levels in the Wi-Fi chipset.

Finally, in order to gain statistical confidence, we run every simulation with 15 independent seeds and plot the 95% confidence intervals on the obtained average values. Notice though that sometimes these confidence intervals are too small to be clearly observed.

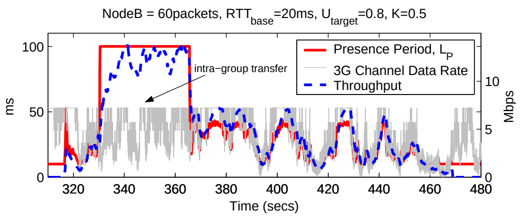
Uplink	384Kbps CBR Bearer
Downlink	HSDPA (UE Category=7, max. PHY rate=7.2Mbps)
TTI	2ms
HARQ Feedback Delay	3 TTI
HARQ Retransmission Delay	6 TTI
HARQ Max Retransmissions	3
Reordering Buffer Size	32 PDUs
T1 Timer	100ms
Channel Models	Pedestrian-A Typical Urban Vehicular-A
Distance from BS	100m 250m 500m
Terminal speed	3 Km/h 50 Km/h 120 Km/h

Table 7.3: Default 3G Simulation Parameters.

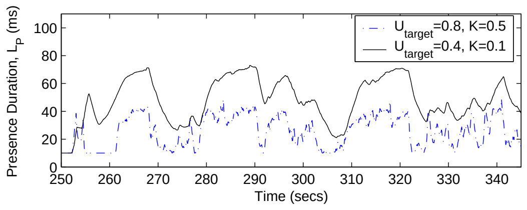
7.2.2.2 Algorithm dynamics

In this section we study the dynamics of the ASPP algorithm and provide a deeper understanding on the effect of its configuration parameters, i.e. K and U_{target} in Equation 7.1. In addition, the maximum and minimum allowed presence durations are set respectively to $L_{P_{max}} = 100ms$ and $L_{P_{min}} = 10ms$.

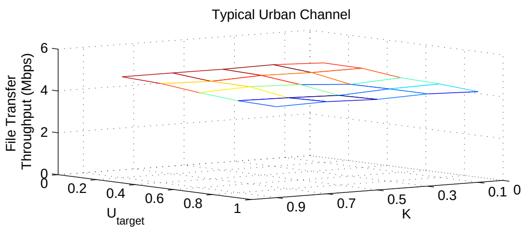
To illustrate the dynamics of ASPP we start performing an experiment where three P2P Clients connect to a P2P Group Owner offering access to a 3G network. The first P2P Client downloads a 50MB file (close to the median video file size in the Internet (91)) from the 3G network, and the other two P2P Clients exchange another 50MB file over the P2P Group. The Typical Urban channel model is used in this example. Figure 7.4(a) depicts the results of this experiment, where a double y-axis is used. The presence durations advertised by the P2P Group Owner are plotted against the left y-axis, while the instantaneous PHY data rate offered in the 3G channel and the throughput being forwarded by the P2P Group Owner are plotted against the right y-axis. As observed in the figure, when only the File Transfer through the 3G link is active (e.g. after 370 seconds), the P2P Group Owner dimensions the presence periods according to the bandwidth variations in the 3G link, in such a way that the P2P Group Owner is only awake the minimum time required to deliver the bandwidth available in the 3G link. However, when the intra-group transfer takes place (330-370 secs), this transfer saturates the Wi-Fi network and the P2P Group Owner stays constantly awake (advertises a presence period of 100ms). Notice that this approach, besides not slowing down the File Transfer, is energy efficient because, if the Wi-Fi network is saturated, the P2P Group Owner spends all its awake time transmitting useful data.



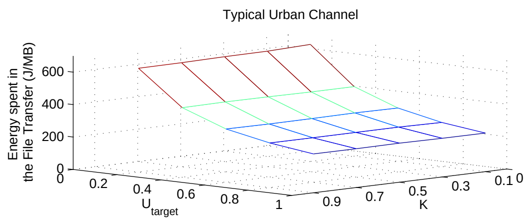
(a) ASPP Dynamics in a File transfer over the Typical Urban channel model.



(b) Effect of U_{target} and K on the advertised presence durations.



(c) Effect of U_{target} and K on the throughput achieved by the P2P Client.



(d) Effect of U_{target} and K on the energy spent by the P2P Group Owner.

Figure 7.4: Dynamics of the *ASPP* algorithm.

We can see in Figure 7.4(a) that ASPP is indeed able to follow variations in the bandwidth available in the 3G link. In a general setting though this will depend on the characteristics of the 3G channel and on the parameters used to configure ASPP, i.e. K and U_{target}. To illustrate the effect of these parameters Figure 7.4(b) depicts the dynamics of the presence periods advertised by the P2P Group Owner, in a setting where a P2P Client retrieves a 50MB file through the 3G network characterized again using the Typical Urban channel model. This time though, for the sake of clarity, the 3G PHY data rate or the achieved throughput are not depicted. Figure 7.4(b) depicts the result of using two different parameter configurations: i) $U_{target} = 0.8/K = 0.5$ and ii) $U_{target} = 0.4/K = 0.1$. Although the throughput experienced by the P2P Client was very similar in both configurations, it is clear in the figure that the advertised presence durations in each configuration significantly differ. The effect of each parameter is the following. On the one hand decreasing U_{target} increases the advertised presence periods, hence increasing power consumption. The reason is that given a certain amount of load, if U_{target} decreases, higher presence periods are needed to maintain the utilization around U_{target} (note that the duration of presence periods is proportional to $\frac{1}{U_{target}}$). On the other hand, K controls how fast ASPP adapts to variations in the traffic flowing between the P2P Group and the 3G network. Clearly, a small K decreases the ability of ASPP to adapt to changes and a too high K might result in a unwanted large presence periods that penalize power consumption.

In order to configure ASPP, Figures 7.4(c) and 7.4(d) illustrate respectively the throughput achieved by the P2P Client while transferring the 50MB file used in our previous example, and the energy spent by the P2P Group Owner during this file transfer. In these figures we vary U_{target} and K between 0 and 1, and report the results obtained with the Typical Urban channel model. Similar results were obtained with the other channel models. Looking at Figure 7.4(d), we can see how a small U_{target} penalizes energy consumption in the P2P Group Owner. However, as depicted in Figure 7.4(c), the throughput achieved by the P2P Client remains relatively stable, meaning that ASPP is able to deliver the required throughput even for high values of U_{target}. In the rest of the paper we will use $U_{target} = 0.8$. Regarding K, we can see that its effect, both in throughput and energy, is relatively small, thus in the rest of the paper we will configure $K = 0.5$.

Finally, it is worth to notice that the same algorithm dynamics hold if an uplink File Transfer is considered. For the sake of clarity though we have not included these results.

7.2.2.3 Steady State Evaluation

In this section we present the results of a steady state set of experiments where we asses the impact of the proposed algorithm on the energy consumption of a P2P Group Owner and on the performance of popular applications like File Transfers and Web browsing.

The performance of a File Transfer transmitted over TCP depends on several parameters: i) the bottleneck bandwidth, ii) the amount of buffering in the bottleneck, and iii) the path delay. In order to account for the effect of each of these parameters we perform two different experiments.

1. *NodeB Buffering Variation Experiment*: TCP file transfer of 50 MB. For each considered 3G channel, we vary the amount of buffering in the NodeB[1] while maintaining a fixed RTT_{base} set to $20ms$.

2. *Path Delay Variation Experiment*: TCP file transfer of 50 MB. For each considered 3G channel, we vary the path delay (RTT_{base} in Figure 7.1), while setting the maximum buffering in the NodeB equal to 30 packets[2] per flow.

In both experiments, we compare ASPP to two competing algorithms:

- *Active* algorithm: The P2P Group Owner switches to active mode once traffic is detected in the network. This algorithm should provide an upper bound with respect to performance and a worst case bound with respect to energy consumption.

- *Static* algorithm: The P2P Group Owner advertises a fixed presence duration equal to 25ms ($\frac{Beacon\ Interval}{4}$).

Finally, in order to asses the effect that the amount of bandwidth available in the Wi-Fi network has on the performance of the algorithms under study, we consider two different Wi-Fi configurations, AC_BK and AC_VI, which are described in Table 7.1. The intent of these two Wi-Fi configurations is to capture the effect of the bandwidth available in the Wi-Fi network on the algorithms under study. The two configurations are named after the Wi-Fi priority[3] used to transport data traffic, and are configured following the recommendations given in the 802.11 standard (120). Thus, the AC_BK configuration uses slower contention settings and should provide a smaller bandwidth. Instead, the AC_VI configuration uses more aggressive contention settings and allows to aggregate several packets in one Transmission Opportunity (TXOP), hence providing a higher bandwidth. Packet aggregation is also a key technology in 802.11n (36), therefore we expect the performance of the AC_VI configuration to provide hints on the performance to be expected with 802.11n.

[1]In practice, buffering could also be limited by the mobile phone acting as P2P Group Owner instead of the NodeB. Notice though that this would not affect the validity of the presented results.

[2]A MTU of 1500B is considered in our evaluation.

[3]Notice that 802.11 defines four priorities for its contention based channel access, i.e. AC_VO, AC_VI, AC_BE and AC_BK.

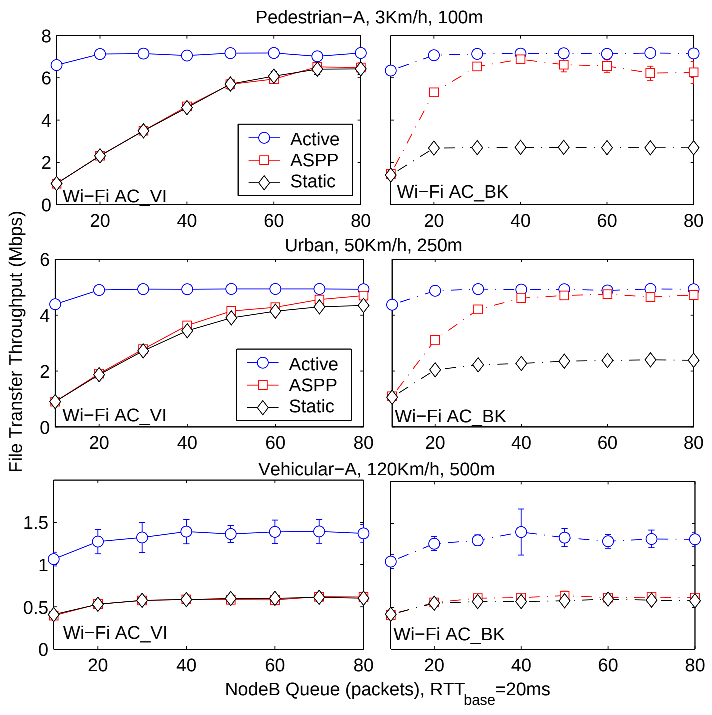
NodeB Buffering Variation Experiment

Figures 7.5(a) and 7.5(b) depict the P2P Group Owner performance under our first *experiment* in terms of connection throughput and energy consumption. The upper, middle and lower plot show, respectively, the performance in the Pedestrian-A channel model (good), the Typical Urban channel model (average), and the Vehicular-A channel model (poor). In addition, the left column in Figures 7.5(a) and 7.5(b) depicts the results obtained with the Wi-Fi AC_VI configuration, and the right column the results obtained with the Wi-Fi AC_BK configuration.

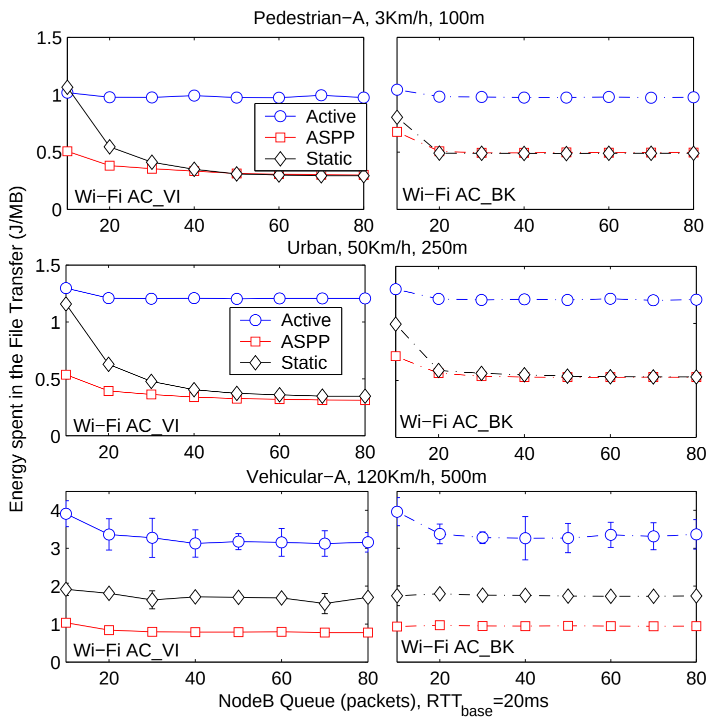
The algorithms under study exhibit a similar behavior in the Pedestrian and Typical Urban channel models (upper and middle plots). Using these channel models, the Active algorithm, as expected, results in the highest throughput (specially when the buffering in the NodeB is small) and in the highest energy consumption. On the other hand, ASPP and the Static algorithm deliver similar throughput to a P2P Client when the AC_VI configuration is used, but significantly differ when the AC_BK configuration is used (with ASPP providing a much higher throughput). Regarding energy, ASPP results in the lowest energy consumption, followed by the Static algorithm. Both algorithms achieve a much lower energy consumption than the Active algorithm ($> 50\%$).

Based on these results the following questions open up: i) Why does throughput degrade so much in ASPP and the Static algorithm when the NodeB buffer is small?, ii) Why does the AC_BK configuration, which in theory should deliver less bandwidth in the Wi-Fi network, outperform the AC_VI configuration in the case of ASPP?, and iii) Why do ASPP and the Static algorithm provide a very similar performance with AC_VI but significantly differ with AC_BK?

In order to answer these questions, Figure 7.6 depicts the dynamics of a TCP connection traversing a P2P Group Owner running a power saving algorithm like ASPP. As it can be seen in the figure, scheduling one presence period every Beacon interval might increase the RTT experienced by the TCP connection. Thus, if the TCP congestion window can not grow large enough to cover the increased bandwidth delay product, as it is likely to be the case when the amount of buffering in the NodeB is small, the 3G link becomes underutilized and the maximum achievable throughput reduces. In order to understand why ASPP performs better with AC_BK than with AC_VI, it should be noted that the higher the bandwidth in the Wi-Fi network, the smaller the presence durations that the P2P Group Owner needs to schedule in order to maintain the required level of utilization, U_{target}, in Equation 7.1. Notice that, although reducing power consumption, this behavior reduces the probability that a new window of TCP data arrives at the P2P Group Owner during an active presence period thus resulting in bursty behavior and an increased RTT, see in Figure 7.6 the second Beacon interval of the AC_BK case (lower P2P Client) for an explanation of why throughput increases in AC_BK. A similar interaction between TCP



(a) Average File Transfer throughput.



(b) Average Energy spent by the P2P Group Owner in the file transfer.

Figure 7.5: File transfer performance when varying the buffering in the NodeB. For each figure the left column represents the results when using the Wi-Fi AC_VI configuration, and the right column the results when using the Wi-Fi AC_BK configuration.

and the 802.11 Power Save Mode (PSM) used by Wi-Fi clients was already pointed out in (39), and was also described in Chapter 6. Finally, in the case of AC_VI, ASPP and the Static algorithm provide a similar throughput performance, and ASPP is slightly more efficient in terms of energy. The reason for the similar throughput performance is that in this case a presence period of $25ms$ turns out to be enough for the P2P Group Owner to deliver the packets that it has buffered at every Beacon frame. Indeed, when the NodeB buffer is below 20 packets even a presence period below 25ms would be sufficient, which is why the Static algorithm wastes some energy in this case. On the other hand, when AC_BK is used the P2P Group Owner needs much more than 25ms to deliver its buffered packets due to the reduced Wi-Fi bandwidth. This is the reason for the lower throughput experienced by the P2P Client when the Static algorithm is used.

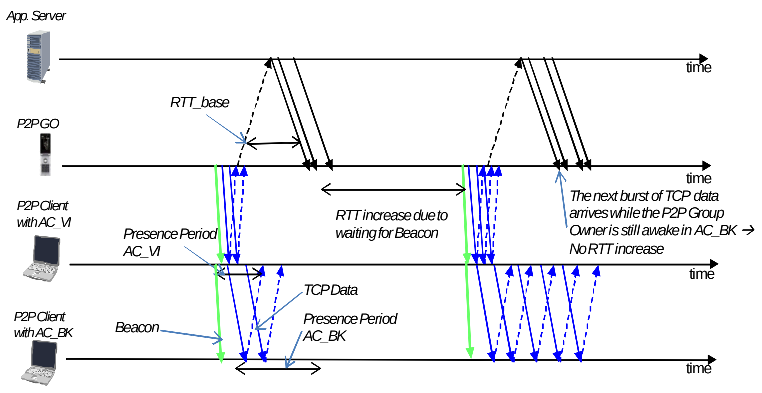

App. Server
P2P GO
P2P Client with AC_VI
P2P Client with AC_BK
time
time
time
time
RTT_base
RTT increase due to waiting for Beacon
The next burst of TCP data arrives while the P2P Group Owner is still awake in AC_BK → No RTT increase
Presence Period AC_VI
TCP Data
Presence Period AC_BK
Beacon

Figure 7.6: Dynamics of a TCP transfer when the P2P Group Owner runs a power saving protocol. Notice that the two P2P Clients illustrate a different case: the upper one a case where the file is transferred over AC_VI, and hence the P2P Group Owner can use smaller presence periods, and the lower one a case where the file is transferred over AC_BK and so the P2P Group Owner uses larger presence periods.

Coming back to Figures 7.5(a) and 7.5(b) and focusing now on the performance achieved in the Vehicular-A (poor) channel model, we can see how ASPP and the Static algorithm deliver a similar throughput to the P2P Client, but ASPP does it in a more energy efficient manner. The reason is that presence periods of $25ms$ are larger than necessary in this case to achieve the delivered bandwidth. In addition, no difference is observed between the AC_VI and AC_BK configurations. This is due to the fact that in this case the 3G link exhibits a *good* condition during such short times that although AC_BK results in slightly higher presence periods, TCP has no time to benefit from them. Finally, it is worth to notice how much the energy consumption in the P2P Group Owner increases when the radio conditions in the 3G link degrade and the Active algorithm is used: $\sim 3.5J/MB$ in the Vehicular-A channel compared to $\sim 1J/MB$ in the Pedestrian-A chanel. The reason is that when the 3G radio conditions are poor and the Active algorithm is used the P2P Group Owner stays most of the time idle, hence wasting power, in the Wi-Fi link.

Path Delay Variation Experiment

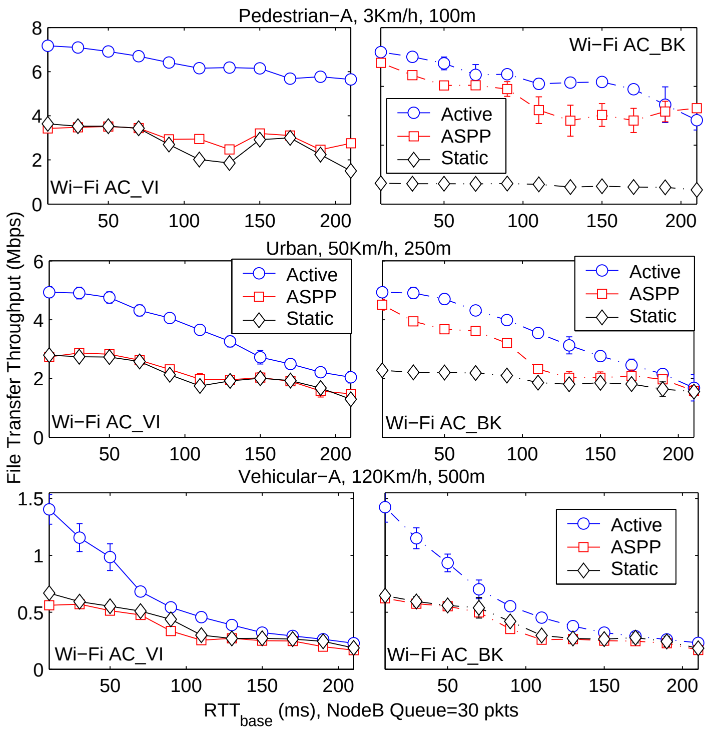
To conclude our File Transfer application performance evaluation with the different algorithms under study, we analyze in this experiment the effect of varying the path delay experienced by the TCP connection (RTT_{base}). The corresponding results are depicted in Figure 7.7. It can be noticed that the lessons learned from the previous experiment where we varied the amount of buffering in the NodeB, can also be applied to explain the behavior observed in this case. A remarkable result from this experiment is the very high energy consumption incurred by the P2P Group Owner in the Vehicular-A channel when the Active algorithm is used ($\sim 7.5J/MB$). The reason is the poor radio conditions in the 3G link plus the big path delays experienced by the TCP connection which result in the P2P Group Owner being almost always idle in the Wi-Fi network.

Web Traffic Experiment

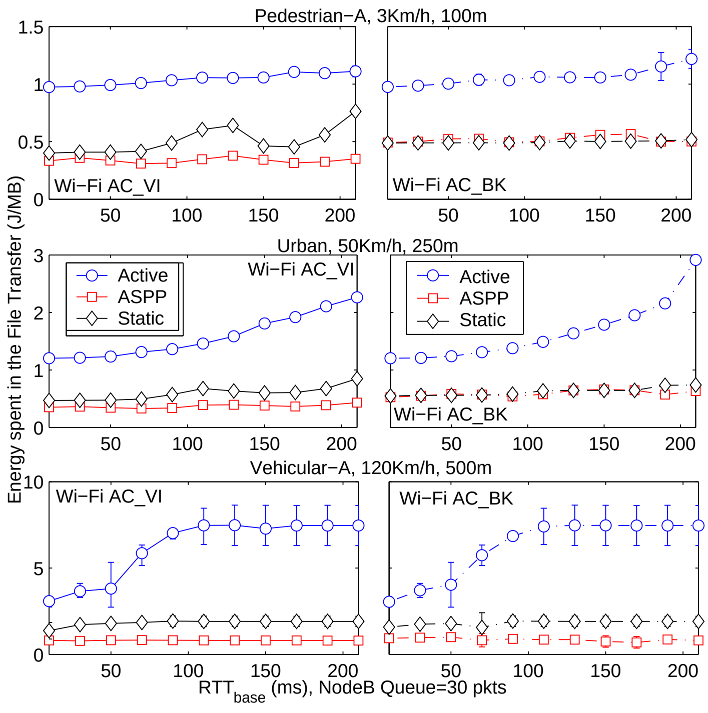
We look now at how the different algorithms under study perform with Web traffic. In order to model Web traffic, we consider HTTP1.1 and have a P2P Client periodically[1] requesting a new web page to the Web server in the Internet, while varying RTT_{base}. The size and number of embedded objects of a Web page are modelled according to the statistics reported in (91). Unlike in the case of the File Transfer, we only depict the results for the Typical Urban (average) channel in the case of Web. The reason is that Web is mostly dominated by the path delay, i.e. RTT_{base}, and hence the results obtained showed the same dynamics in our different 3G channel models. For the same reason only the AC_VI configuration is considered in the case of Web. However, we introduce a variation with respect to our File Transfer experiment, we consider the P2P Client to be in active mode or in power save mode (PSM). The reason is that the performance of Web traffic already degrades when a Wi-Fi client is in power save mode (39), therefore it is important to asses whether a power saving protocol used by a P2P Group Owner degrades even further the performance experienced by a P2P Client in power save mode.

Figures 7.8(a) and 7.8(b) depict, respectively, the average web page transfer time and the average energy consumed by the P2P Group Owner per Web page. Looking at Figure 7.8(a), we can see that the P2P Client experiences with ASPP a Web page transfer time equivalent to the one that the P2P Client experiences when it is in power save mode. This is because in both cases the P2P Client can only transmit data during a certain period after the Beacon frame. Regarding the energy spent by the P2P

[1]More than 1000 Web pages are requested by the P2P Client in each simulation run.

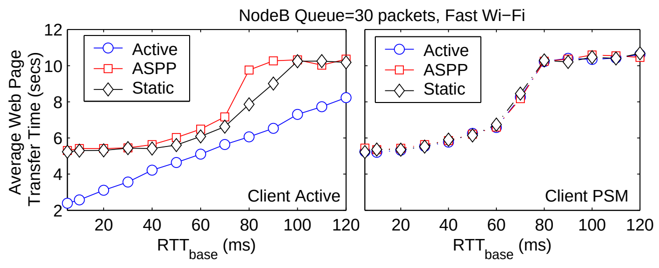


(a) Average File Transfer throughput.

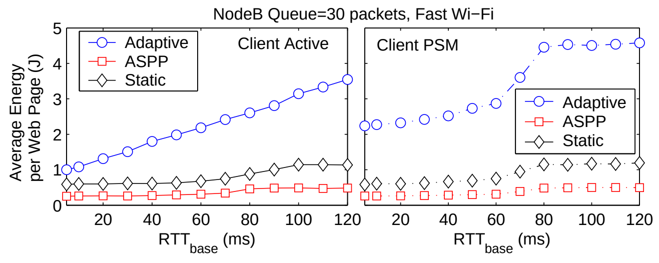


(b) Average Energy spent by the P2P Group Owner in the file transfer.

Figure 7.7: File transfer performance when varying the path delay (RTT_{base}). For each figure the left column represents the results when using the Wi-Fi AC_VI configuration, and the right column the results when using the Wi-Fi AC_BK configuration.



(a) Average Web page download time.



(b) Average Energy spent by the P2P Group Owner per Web page.

Figure 7.8: Web traffic performance.

Group Owner to transfer a Web page, Figure 7.8(b) shows that a very significant energy reduction in the P2P Group Owner can be achieved with *ASPP* with respect to the *Active* and *Static* algorithms, specially as RTT_{base} grows. The reason is that the little traffic generated by the Web application together with the sequential nature of HTTP allows ASPP to operate with presence intervals as low as $L_{P_{min}} = 10$ms, while the P2P Group Owner remains idle for long times if the *Active* or *Static* configuration are used.

7.3 AMPP: Adaptive Multiple Presence Periods

In the previous section we have presented and analyzed the ASPP algorithm which schedules a single presence period per Beacon interval and is applicable to the two Wi-Fi Direct power management schemes described in Section 7.1: Opportunistic Power Save and Notice of Absence. We have seen that ASPP has a low implementation complexity and significantly reduces energy consumption in a P2P Group Owner. However, sometimes it could result in a degraded user experience with respect to an algorithm that configures the P2P Group Owner to be in active mode upon detecting traffic in the network. Our analysis has shown that the main reason why ASPP may degrade throughput is the fact that it schedules a single presence period per Beacon interval which results in an increased RTT experienced by TCP connections. Therefore, a possible way to improve upon the performance of ASPP is to design an algorithm that schedules not only one but *multiple* presence periods within a Beacon interval[1]. In this section we present the design of such an algorithm, which is hereafter referred to as the *Adaptive Multiple Presence Periods (AMPP)* algorithm. Note that while ASPP could be implemented in both Opportunistic Power Save and Notice of Absence protocols, AMPP can only be implemented using the Notice of Absence protocol since this is the only one that allows to schedule multiple presence periods within a Beacon interval.

7.3.1 Algorithm design

The main goal of AMPP is to improve the performance experienced by P2P Clients compared to ASPP, whilst keeping as much as possible the power saving achieved by the P2P Group Owner with ASPP. For this purpose the design of AMPP is based on the following building blocks:

1. Adaptively dimension the size of the (multiple) presence periods advertised by AMPP. This is done based on the ASPP algorithm.

2. Design of an algorithm that estimates the raw bandwidth available in the 3G link. Note that the analysis of ASPP has shown that when the buffering in the NodeB is small a TCP connection does not always saturate the 3G link, and the amount of data received in the P2P Group significantly underestimates the bandwidth available in the 3G link.

3. Design of an algorithm that, based on the estimation of the 3G link available bandwidth, adjusts the interval between consecutive presence periods in the Notice of Absence protocol. Hence, controlling how many presence periods should be scheduled within a Beacon interval.

Next, we describe the design of the two new modules used by AMPP: i) the external bandwidth estimation algorithm, and ii) the presence interval adaptation algorithm. During our description we assume that more traffic is transmitted in downlink (network$\rightarrow$P2P Client) than in uplink. Later in in this section we will discuss how the presented algorithm could be tailored to the uplink case.

7.3.1.1 External network bandwidth estimation algorithm

In this section we describe the algorithm used by AMPP designed to estimate the bandwidth available in an external network. Hereafter we will focus in the case of the external network being a 3G link. However, the same bandwidth estimation algorithm could be applied to different external networks.

[1]Notice that we had a similar problem in Chapter 6 but from the perspective of the station. In that case BA-TA was used to reduce the trigger interval used by the station.

According to the architectural constraint set in the previous section, the designed bandwidth estimation algorithm should not interact with the 3G interface. Therefore, we design a passive algorithm that derives the bandwidth available in the 3G link by observing, in the Wi-Fi driver of the P2P Group Owner, the interarrival times of packets arriving from the 3G link[1]. Note that such an algorithm will not interfere with the traffic flowing between the P2P Group and the 3G network.

The design of the bandwidth estimation algorithm is based on the following observations. If the buffer in the NodeB would be always full, the 3G link would modulate the arrival times of packets at the P2P Group Owner, and the 3G bandwidth could be easily estimated. In general though, the buffer in the NodeB may become empty during a TCP connection (specially if this buffer is small), and in this case the interarrival times of packets at the P2P Group Owner will not follow the 3G bandwidth. Indeed, since TCP packets are buffered by the P2P Group Owner and then transmitted during a presence period at the rate of the Wi-Fi network (usually higher than the bottleneck rate), TCP ACK compression[2] can cause bursty increases in the bottleneck queue (NodeB) (123), see Figure 7.9. Therefore, when a TCP connection is not fully utilizing the bandwidth in the 3G link, a P2P Group Owner typically observes bursts of TCP packets modulated by the bandwidth in the 3G link separated by *inter-burst* times. This characteristic arrival pattern is depicted in Figure 7.9. Thus, an ideal bandwidth estimation algorithm should discard these inter-burst times and estimate the 3G bandwidth using only the interarrival times within a burst. In order to accomplish this, an important observation is that the inter-burst times observed at the P2P Group Owner are strongly correlated with the presence intervals being used by the P2P Group Owner. This fact is also illustrated in Figure 7.9 where, assuming a constant RTT, the time between the first packet of the two bursts of TCP data arriving at the P2P Group Owner is equal to the presence interval being used by the P2P Group Owner. As a result of the previous observations, we design our bandwidth estimation algorithm based on the following assumption : *within a time interval equal to a presence interval, the P2P Group Owner can only observe one or none inter-burst times*. In the Appendix we formalize this assumption and study under which conditions it holds.

Our bandwidth estimation algorithm is described in detail in Algorithm 7.1, and is run by a P2P Group Owner before transmitting the Beacon frame. Thus, before building a Beacon frame a P2P Group Owner estimates the bandwidth available in the 3G link during the last Beacon interval and uses this estimation in order to decide how many presence periods should be scheduled in the upcoming Beacon interval. Notice that, estimating the bandwidth available in the 3G link every Beacon interval (e.g. $100ms$) sets an upper limit on the 3G channel variation rate that AMPP will be able to follow.

[1]Note that we assume that the device's clock granularity is accurate enough to derive meaningful bandwidth estimations from the measured interarrival times.

[2]Notice that the same effect was discussed in detail in Chapter 6.

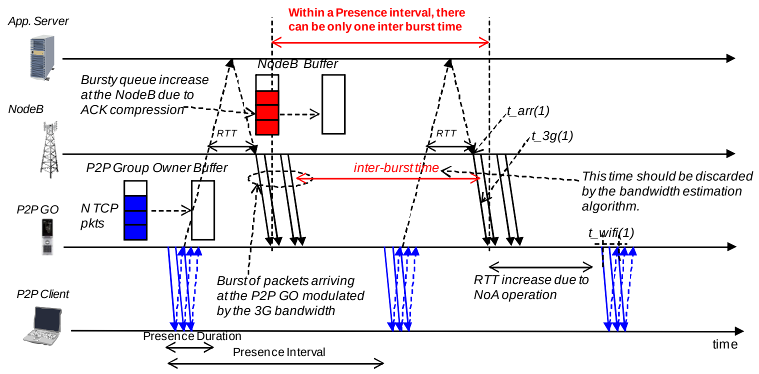

App. Server
NodeB
P2P GO
P2P Client
Within a Presence interval, there can be only one inter burst time
NodeB Buffer
Bursty queue increase at the NodeB due to ACK compression
RTT
RTT
t_arr(1)
t_3g(1)
P2P Group Owner Buffer
inter-burst time
This time should be discarded by the bandwidth estimation algorithm.
N TCP pkts
t_wifi(1)
Burst of packets arriving at the P2P GO modulated by the 3G bandwidth
RTT increase due to NoA operation
Presence Duration
Presence Interval
time

Figure 7.9: Insight on the bandwidth estimation algorithm: From the perspective of the P2P Group Owner, TCP data packets arrive in bursts modulated by the bandwidth of the 3G link, interleaved with inter-burst times that appear when the buffer in the NodeB becomes empty. One can notice in the figure how within a presence interval, there can only be one inter-burst time.

Algorithm 7.1: estimate_3g_bw().

```
– Variables definition
$interarrvs$ ← array containing the interarrivals of the packets received in the last Beacon interval.
$total\_bits$ ← accumulated sizes of the packets received in the last Beacon interval.
$num\_pkts$ ← Number of packets received during the last Beacon interval.
$t_{b2b}$ ← interarrival time below which two packets are considered back to back.
$M$ ← threshold on consecutive back to back packets.
$interval$ ← Current operating interval of the algorithm.
$subintvl \leftarrow \{\}$
– Routine executed when a SP completes
$t_{total}, t_{subint}, m, interarrv_{max} \leftarrow 0, avg_{last} \leftarrow -1$
for $i = 0$ to $num\_pkts - 1$ do
    if $interarrvs(i) > t_{b2b}$ then
        $t_{total} \leftarrow t_{total} + interarrvs(i)$
        $t_{subint} \leftarrow t_{subint} + interarrvs(i)$
        if $interarrvs(i) > interarrv_{max}$ then
            $interarrv_{max} \leftarrow interarrvs(i), i_{max} \leftarrow i, m_{max} \leftarrow m$
        $subintvl \leftarrow \{subintvl, interarrvs(i)\}, m \leftarrow m + 1$
    if $i = num\_pkts - 1$ or $m > 1, t_{subint} + interarrvs(i+1) > interval$ then
        if $m > 1$ and $!interarrvs(i_{max}+1, ..., i_{max}+M) < t_{b2b}$ then
            $avg \leftarrow \frac{t_{subint} - interarrv_{max}}{m-1}, avg_{last} \leftarrow avg$
            $std \leftarrow \sqrt{\frac{1}{m-1}\sum_{j=1, j!=m_{max}}^{m}(subintvl(j) - avg)^2}$
            if $interarrv_{max} > avg + 2std$ then
                $t_{total} \leftarrow t_{total} - interarrv_{max} + avg$
        else if $m = 1$ and $avg_{last}! = -1$ then
            $t_{total} \leftarrow t_{total} - interarrv_{max} + avg_{last}$
        $m \leftarrow 0, t_{subint} \leftarrow 0, interarrv_{max} \leftarrow 0, subintvl \leftarrow \{\}$
if $num\_pkts > 0$ and $t_{total} > 0$ then
    $3g\_bw \leftarrow \alpha \times 3g\_bw + (1 - \alpha) \times \frac{total\_bits}{t_{total}}$
$num\_pkts \leftarrow 0, total\_bits \leftarrow 0$
```

This limitation though, is not intrinsic to AMPP but to the Notice of Absence protocol which can only update the current schedule every Beacon interval. In addition, in order to follow the variations in the 3G bandwidth, our bandwidth estimation algorithm uses an EWMA filter updated with the bandwidth estimated every Beacon interval (line 28 in Algorithm 7.1). Next, we describe the details of our bandwidth estimation algorithm.

A P2P Group Owner records in each Beacon interval the amount of received data, $total\_bits$, the number of packets, num_pkts, and a list of the interarrival times between packets received from the 3G link, $interarrvs(i)$. Note that this list can be fairly small, e.g. assuming a speed of 7.2Mbps in the 3G link and 1.5KB size packets, only 60 packets could arrive within a Beacon interval of 100ms. Then, the list $interarrvs(i)$ is splitted in contiguous subintervals according to the current presence interval ($interval$) used by the P2P Group Owner (line 18).

For each of these subintervals the maximum interarrival time ($interarrv_{max}$) is tagged as a potential *inter-burst* time (line 16), assuming that within a subinterval *there can only be one inter-burst time*. Then it is decided whether $interarrv_{max}$ is indeed an inter-burst time and should therefore be discarded by comparing it against the average (avg) and standard deviation (std) of the other interarrival times contained in $subintvl$. Specifically, if $interarrv_{max} > avg + 2std$ we heuristically consider $interarrv_{max}$ to be an inter-burst time and substitute it by avg (line 23). Notice that the challenge is to distinguish between the long tail in the distribution of interarrival times caused by the 3G link, and the inter-burst times caused by an empty queue in the NodeB. Since the Chebychev inequality (125) establishes that at maximum $\frac{1}{k^2}$ of the distribution values can be separated more than $k\sigma$ from the distribution mean, setting $k = 2$ hints that 75% of the interarrival times due to the 3G bandwidth should be correctly accounted for, while still providing a reasonable protection against inter-burst times. Notice that this heuristic allows us to follow variations in the 3G link, because our *threshold* to detect inter-burst times, i.e. $avg + 2std$, adapts to the statistics of the 3G link. Obviously the presented heuristic can fail, if the time to transmit a packet over a 3G link is higher than our threshold, although the Chebychev inequality should limit the probability of that happening, or if an inter-burst time is actually smaller than the threshold, although in that case the impact of mistaking an inter-burst time with an interarrival time caused by the 3G link should not be significant. In the next section we will evaluate how effective this heuristic is on estimating the bandwidth available in the 3G link.

We set one exception to the previous rule if there are M consecutive *back to back* packets after $interarrv_{max}$ (line 19), where back to back packets are defined as two consecutive packets arriving within t_{b2b}. The reason for this exception is that we empirically observed that a pattern of a long interarrival time followed by several consecutive back to back packets is typically caused by the Hybrid-ARQ protocol in the 3G link (124), which buffers frames to guarantee in order delivery, and not by an empty queue in the NodeB. In our implementation we set $M = 2$ and $t_{b2b} = 2ms$ which is the slot size used in HSDPA. In addition, if there is only one sample in $subintvl$, this value is not trusted, and instead the last recorded average interarrival time is considered, avg_{last} (lines 24-25).

Finally, note that by adjusting the value of the variables t_{b2b} and M Algorithm 7.1 could be tailored to different cellular technologies, like LTE or WiMAX, which operate based on principles similar to the ones outlined in this section.

7.3.1.2 Adapting the Number of Presence Intervals per Beacon

In this section, we describe how AMPP uses the input provided by our bandwidth estimation algorithm in order to decide how many presence periods should be scheduled every Beacon interval. For this purpose what AMPP does is to adjust the value of the presence interval advertised in the Notice of Absence schedule being broadcasted in the Beacon, i.e. if $interval = 25ms$ then there will be four presence periods within a Beacon interval. Our interval adaptation algorithm shares the same design phylosophy than the BA-TA algorithm presented in Chapter 6, and its main ideas are as follows:

- *Underutilization Detection*: AMPP maintains two estimates: i) an estimate of the bandwidth available in the 3G link obtained with our bandwidth estimation algorithm, $3g\_bw$, and ii) an estimate of the traffic flowing between the P2P Group and the 3G link, thr. In order to detect if the 3G link is being underutilized, AMPP computes the ratio between its two estimates, i.e. $ratio = \frac{thr}{3g\_bw}$.
- *Reaching Utilization Target*: AMPP takes as input parameter a desired utilization target of the 3G link, i.e. $ratio_{min}$. Based on this, AMPP checks if the desired utilization is satisfied and adjusts the selected presence interval in the following way: i) if the 3G link is underutilized, i.e. $ratio < ratio_{min}$, AMPP will decrease its operating interval (increase the number of presence periods) within a Beacon interval in an attempt of reducing the RTT experienced by a TCP connection and improve throughput, ii) if utilization is sufficient, i.e. $ratio > ratio_{min}$, AMPP will increase its operating interval to check whether a higher presence interval still delivers the desired utilization. Note that higher presence intervals are preferred since they allow for longer sleep times and thus, higher power saving.

In addition to the previous main algorithm principles more issues need to be considered in practice to achieve the desired functionality. Our first design decision is that, in order to be able to refresh the used presence interval every Beacon interval, AMPP will only select presence intervals that are sub-multiple of the Beacon interval, i.e. $interval = \frac{BI}{k_{curr}}$, where $k_{min} \leq k_{curr} \leq k_{max}$, $k_{curr}, k_{min}, k_{max} \in \mathbb{Z}$. Notice that if an absence period would overlap with a Beacon frame, a P2P Client might skip the Beacon, missing the updated schedule. A detailed description of our interval adaptation algorithm is provided in Algorithm 7.2 and summarized next.

We start discussing the case where $ratio < ratio_{min}$ (lines 9 to 29). In this case AMPP considers the link to be underutilized and thus, tries to decrease the current presence interval. However, in order to reduce spurious updates, we introduce a memory, $count_{down_{max}}$, which establishes a number of consecutive Beacon intervals before AMPP updates the current presence interval. We can see between lines 20 and 29 how AMPP decreases the presence interval by increasing the parameter k_{curr}, where $interval = \frac{BI}{k_{curr}}$. In addition, under special circumstances AMPP may as well decide to keep the P2P Group Owner awake during the upcoming Beacon interval. These circumstances are: i) if the current presence duration, L_P, is above the selected interval (line 24), or ii) if the desired utilization, $ratio_{min}$, is not achieved even when operating at the minimum allowed interval, k_{max}, (line 28). Notice that when staying awake the P2P Group Owner does not publish any Notice of Absence schedule in the Beacon frame.

The P2P Group Owner should be in active mode (awake) only if the Wi-Fi interface is busy transmitting packets. Hence, before deciding to keep the P2P Group Owner awake, Algorithm 7.2 records in the variable $\Delta_{r_{slp}}$ the speed of increase of $ratio$. Thus, if the 3G link continues to be underutilized, i.e. $ratio < ratio_{min}$, but the P2P Group Owner is in active mode (lines 13-19), the algorithm checks whether the speed of increase of $ratio$, $\Delta_{r_{act}}$, is above $\Delta_{r_{slp}}$. If that is not the case, keeping the P2P Group Owner awake is not being effective and the P2P Group Owner goes back to normal power saving operation. In order to effectively compare $\Delta_{r_{act}}$ and $\Delta_{r_{slp}}$, a hysteresis is introduced controlled by the γ parameter.

Now we consider the case where the 3G link is being sufficiently utilized, i.e. $ratio \geq ratio_{min}$ (lines 30 to 37). In this case AMPP increases the operating interval by means of decreasing the variable k_{curr}, where $interval = \frac{BI}{k_{curr}}$. Note that, again, a memory has been introduced to reduce spurious updates, $count_{up_{max}}$. In addition, if the P2P Group Owner happened to be in active mode at that moment, the last presence interval and presence duration used before switching to active mode are restored (line 35).

One more practical aspect to be considered is the case where applications, e.g. Web, do not offer enough load to saturate the 3G link. In this case the previous logic would drive the P2P Group Owner towards using small presence intervals, which would not be energy efficient. In order to counter

Algorithm 7.2: adjust_interval().

```
– Variables definition
$ratio_{min} \leftarrow$ 3G link utilization threshold.
$count_{up_{max}}/count_{down_{max}} \leftarrow$ Variables to control the speed of interval increase/decrease.
$k_{min}/k_{max} \leftarrow$ Variables that control the maximum/minimum presence intervals.
$n\_int_{max} \leftarrow$ Threshold on the maximum number of presence periods without data before updating the used interval.
– Routine executed every Beacon interval $3g\_bw \leftarrow estimate\_3g\_bw(), thr \leftarrow estimate\_thr()$
if $3g\_bw, thr > 0$, $n\_int\_no\_data < n\_int_{max}$ then
    $ratio \leftarrow \frac{thr}{3g\_bw}$
    if $ratio < ratio_{min}$ then
        $count_{up} \leftarrow 0, count_{down} \leftarrow count_{down} + 1$
        if $count_{down} = count_{down_{max}}$ then
            $count_{down} \leftarrow 0$
            if $active$ is $1$ then
                $\Delta_{r_{act}} \leftarrow \frac{ratio - ratio_{last}}{count_{down_{max}}}$
                if $\Delta_{r_{act}} < \max\{0, (1+\gamma)\Delta_{r_{slp}}\}$ then
                    $count_{act} \leftarrow count_{act} + 1$
                    if $count_{act} = n\_int_{max}$ then
                        $active \leftarrow 0, L_P \leftarrow L_{P_{last}}, k_{curr} \leftarrow k_{last}, last \leftarrow up, count_{act} \leftarrow 0$
                $ratio_{last} \leftarrow ratio$
            else
                $go\_to\_active \leftarrow$ ($last = down$ and $ratio > (1+\gamma)ratio_{last}$) or ($last = up$ and
                $ratio < (1-\gamma)ratio_{last}$)
                if $k_{curr} < k_{max}$ then
                    $k_{curr} \leftarrow \min\{k_{max}, k_{curr} + 1\}, ratio_{last} \leftarrow ratio, last \leftarrow down$
                    if $\frac{BI}{k_{curr}} < L_P$ then
                        $active \leftarrow 1, \Delta_{r_{slp}} \leftarrow \frac{ratio - ratio_{last}}{count_{down_{max}}}, count_{act} \leftarrow 0$
                    else
                        $active \leftarrow 0, L_{P_{last}} \leftarrow L_P, k_{last} \leftarrow k_{curr}$
                else if $go\_to\_active$ then
                    $active \leftarrow 1, \Delta_{r_{slp}} \leftarrow \frac{ratio - ratio_{last}}{count_{down_{max}}}, ratio_{last} \leftarrow ratio, last \leftarrow down, count_{act} \leftarrow 0$
    else
        $count_{down} \leftarrow 0, count_{up} \leftarrow count_{up} + 1$
        if $count_{up} = count_{up_{max}}$ then
            $count_{up} \leftarrow 0, last \leftarrow up, ratio_{last} \leftarrow ratio$
            if $active$ is $1$ then
                $active \leftarrow 0, L_P \leftarrow L_{P_{last}}, k_{curr} \leftarrow k_{last}$
            else
                $k_{curr} \leftarrow \max\{k_{min}, k_{curr} - 1\}$
else if $n\_int\_no\_data \geq n\_int_{max}$ then
    $k_{curr} \leftarrow \max\{k_{min}, k_{curr} - 1\}, n\_int\_no\_data \leftarrow 0, last \leftarrow up, ratio_{last} \leftarrow ratio, count_{up} \leftarrow 0,$
    $count_{down} \leftarrow 0$
$interval \leftarrow \frac{BI}{k_{curr}}$
```

this effect, the variable $n\_int\_no\_data$ is defined in Algorithm 7.2, which accounts for the number of scheduled presence periods during the last Beacon interval where no data was transmitted. Thus, if $n\_int\_no\_data > n\_int_{max}$ the algorithm quickly increases the operating interval (lines 38-39).

Finally, the design of AMPP allows to trade-off energy and performance by configuring the parameters $ratio_{min}$, k_{min}/k_{max} (maximum/minimum presence intervals), and $count_{down_{max}}/count_{up_{max}}$ (speed of decrease/increase). For instance applications with strict QoS requirements, e.g. VoIP, can be easily accomodated within the presented framework, by configuring the presence interval, i.e. k_{min}/k_{max}, to operate within limits that fulfill their delay constraints. These requirements can be conveyed by a P2P Client to a P2P Group Owner using the Notice of Absence protocol by sending a P2P Presence Request (see Section 7.1). In the next section we will study the effect of the previous parameters.

7.3.2 AMPP: Algorithm evaluation

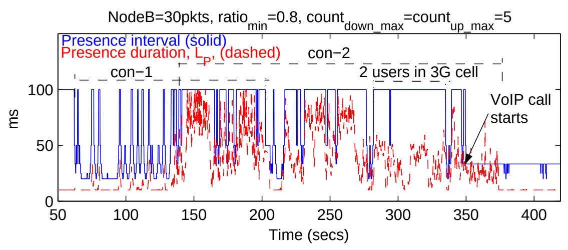
In this section we will evaluate the performance of AMPP. Like in the case of ASPP, we will start analyzing the algorithm dynamics, to then present a steady state evaluation with popular applications.

7.3.2.1 Algorithm dynamics

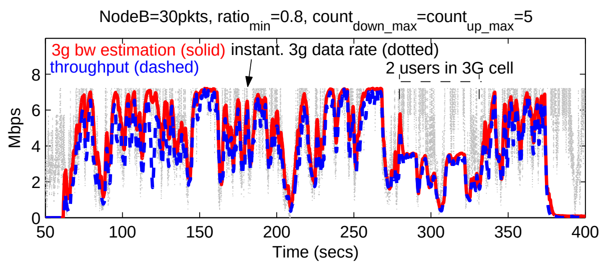
We start presenting the results of an experiment where two P2P Clients connected to a mobile phone acting as P2P Group Owner download a 50MB file through the 3G network. The Typical Urban channel model is considered in this experiment. The two File Transfers from each P2P Client experience different connection delays, $RTT_{base_1} = 40ms$ and $RTT_{base_2} = 20ms$, and a 30 packets per flow buffer is considered in the NodeB.

Figure 7.10(a) illustrates the dynamics of the presence interval (solid line) and presence duration (dashed line) advertised by the P2P Group Owner in this experiment. As it can be observed, before 60 seconds (no traffic present) the P2P Group Owner advertises a big interval of 100ms and a small duration of 10ms. This is the default operation when there is no traffic in the network which allows a P2P Group Owner to operate in a very power efficient way. After 60 seconds, when the first File Transfer starts (signaled as con-1 in the figure), the algorithm starts adjusting the advertised presence intervals and durations. In this case AMPP needs to schedule small intervals in order to keep the throughput of the connection above the configured ratio ($ratio_{min} = 0.8$). In order to realize why this is needed recall how ASPP's throughput degraded with a buffer of 30 packets in the NodeB, shown in Figure 7.5(a).

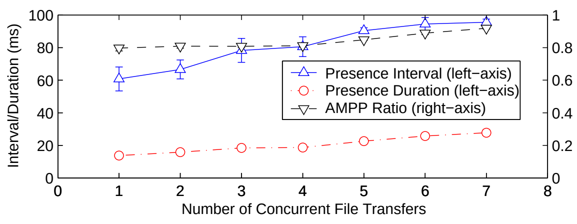
At 140 seconds the second connection starts (con-2) and the aggregate load at the NodeB increases which, together with a slight improvement in the 3G channel condition (that can be observed in Figure 7.10(b)), allows the P2P Group Owner to achieve the required throughput using bigger intervals and



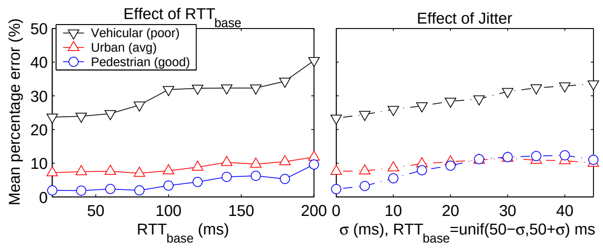
(a) Duration and Interval dynamics. Solid/Dashed lines depict the variation of the presence interval/duration.



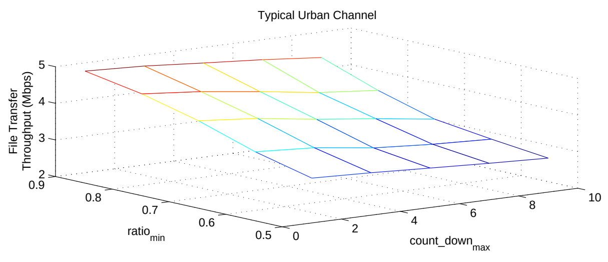
(b) $3g\_bw$ and achieved throughput. Notice how the $3g\_bw$ estimation and the delivered throughput follow the 3G available bandwidth.



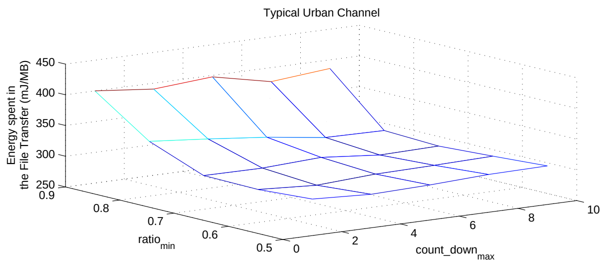
(c) Dynamics when increasing the number of concurrent file transfers. Notice that $AMPP\ Ratio$ is plotted against the right axis.



(d) 3G Bandwidth Estimation Accuracy. The solid lines are plotted against the lower x-axis, and the dashed ones against the upper x-axis.



(e) AMPP Throughput as a function of $ratio_{min}$, $count_{down_{max}}$ and $count_{up_{max}}$.



(f) AMPP Energy as a function of $ratio_{min}$, $count_{down_{max}}$ and $count_{up_{max}}$.

Figure 7.10: Dynamics of the *AMPP* algorithm.

presence durations. After 200 seconds the first file transfer completes and the P2P Group Owner is forced to use again smaller intervals to maintain throughput. However, it is interesting to notice in this case that since $RTT_{base_2} < RTT_{base_1}$, the P2P Group Owner can now operate with bigger intervals than with the first File Transfer.

Between 280 and 330 seconds, a second user enters the 3G cell that was serving the P2P Group Owner halving the bandwidth available to the mobile phone (P2P Group Owner) in the 3G link[1]. AMPP's bandwidth estimation algorithm detects this bandwidth reduction (see Figure 7.10(b)) and, in order to save energy, the P2P Group Owner starts operating with a bigger interval and a smaller presence duration.

Finally, at 350 seconds a VoIP call enters the system. The VoIP client sends a P2P Presence Request frame to the P2P Group Owner requesting the P2P Group Owner to be present at least every 40ms in order to maintain the QoS needed in the Voice call. AMPP can easily accomodate the incoming call by configuring $k_{min} = 3$ upon receiving a P2P Presence Request from the P2P Client, which establishes a maximum presence interval of $33ms$.

Figure 7.10(b) depicts for the same experiment the instantaneous rate in the 3G link, the $3g\_bw$ estimation and the throughput delivered to the P2P Client. Notice how by means of adapting interval and duration, AMPP is able to deliver to the P2P Client all the bandwidth available in the 3G channel.

We study now how the AMPP algorithm behaves when not one but several TCP connections are established concurrently through the P2P Group Owner. Figure 7.10(c) depicts the result of an experiment where we analyze the average presence interval and duration selected by AMPP, together with the ratio between the throughput delivered to the P2P Clients and the bandwidth available in the 3G link. In the experiment we incrementally increase the number of P2P Clients connected to the P2P Group Owner from 1 to 8. Each of the P2P Clients in our experiment retrieves a 50MB file using a TCP connection that experiences a different path delay (RTT_{base}) between $20ms$ and $80ms$. In the figure the average presence interval and duration are plotted against the left y-axis while the AMPP ratio is plotted against the right y-axis. Notice that from the perspective of AMPP it does not matter if packets come from the same or different connections.

From the results in Figure 7.10(c) we can observe that in all cases AMPP is able to deliver the required ratio of the 3G bandwidth ($ratio_{min} = 0.8$). However, if only one client is present, AMPP needs to use smaller presence intervals (schedule more presence periods within a Beacon frame). The reason is that, as the number of concurrent TCP connections increases, the buffer in the NodeB becomes empty

[1] In our simulation the 3G cell operates using a Round Robin scheduler

less often. Thus, AMPP benefits from this phenomenon by operating with higher presence intervals which is more energy efficient.

In order to evaluate the accuracy of our bandwidth estimation algorithm, Figure 7.10(d) presents the mean percentage error achieved by our bandwidth estimation algorithm in the following experiments. A first experiment (left sub-graph) where we increase the value of RTT_{base} while keeping the same buffer size in the NodeB. Notice that a higher path delay increases the bandwidth delay product and hence the likelihood of having an empty buffer in the NodeB, which is the case that should be detected by our bandwidth estimation algorithm. A second experiment (right sub-graph), where we introduce an increasing jitter around an average path delay of 50ms.

Notice that, as shown in the Appendix, jitter is one of the conditions that challenge the fundamental assumption of our bandwidth estimation algorithm, i.e. the existence of a single inter-burst time within a presence interval. We repeated the previous two experiments for our different 3G channel models. As seen in Figure 7.10(d) the error of our 3G bandwidth estimation algorithm is kept below 10% for the whole parameter range in the Pedestrian-A and Typical Urban channels. Only in the fastly varying Vehicular-A channel, the estimation error reaches values around 30% because the TCP connection often stalls in this case not providing enough packets for AMPP to have a reliable estimation.

In order to gain a deeper understanding on the influence of the $ratio_{min}$, $count_{down_{max}}$ and $count_{up_{max}}$ AMPP parameters, Figures 7.10(e) and 7.10(f) depict respectively the throughput and energy spent in the P2P Group Owner in the Typical Urban channel, when a P2P Client downloads a 50MB File and a 30 packets buffer is considered in the P2P Group Owner. In this experiment, $ratio_{min}$ is varied between 0.5 and 0.9 and $count_{down_{max}}$ is varied between 1 and 9, having $count_{up_{max}} = 10 - count_{down_{max}}$. We can see looking at the previous figures how increasing $ratio_{min}$ or decreasing $count_{down_{max}}$ and increasing $count_{up_{max}}$ increases both throughput and energy consumption, being $ratio_{min}$ the strongest parameter, since it results in the P2P Group Owner offering more presence intervals within a Beacon interval[1].

Finally, note that AMPP could be applied also to uplink File Transfers. In this case though, AMPP should estimate the 3G uplink available bandwidth from the returning TCP ACKs. We have not further studied this approach because our experiments reveal that given the usually limited uplink 3G bandwidth, ASPP (no interval adaptation) suffices to provide an adequate performance in this case.

[1]Notice though that the energy profile depicted in Figure 7.10(f) may slightly vary depending on the power characteristics of the considered Wi-Fi chipset.

7.3.2.2 Steady State Evaluation

In this section we evaluate the performance of our *AMPP* algorithm with popular applications like File Transfers and Web traffic. The same experiments defined in section 7.2.2.3 for the ASPP evaluation are used here to study AMPP's steady state performance. In this case we will focus in the results obtained with the AC_VI settings since this was the Wi-Fi configuration providing the best energy efficiency in ASPP but exhibiting the highest performance degradation. It is therefore the goal of AMPP to overcome the performance problems of ASPP while achieving similar energy efficiencies.

Two different configurations for AMPP that illustrate the effect of its configurations parameters will be considered:

- *QoS configuration*: $ratio_{min}$, $count_{down_{max}}$ and $count_{up_{max}}$ set to 0.9, 5 and 10 respectively.
- *Energy configuration*: $ratio_{min}$, $count_{down_{max}}$ and $count_{up_{max}}$ set to 0.8, 10 and 5 respectively.

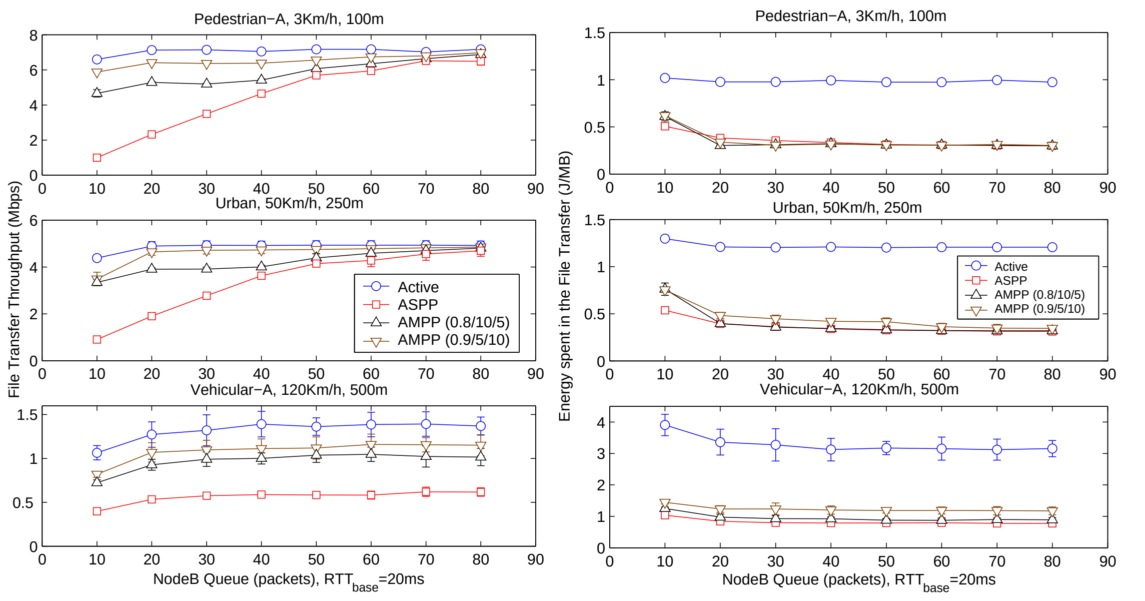
The maximum and minimum allowed presence intervals k_{min}/k_{max} are set to $20ms$ and $100ms$ respectively. Finally, the parameters n_int_{max} and γ are empirically set to 3 and 0.1 respectively.

NodeB Buffering Variation Experiment

Figures 7.11(a) and 7.11(b) depict the performance of the considered algorithms during a File Transfer when varying the buffering available in the NodeB and setting the path delay equal to $RTT_{base} = 20ms$. In the figure it can be clearly observed how for all of our considered 3G channel models, i.e. Pedestrian-A, Typical Urban and Vehicular-A, the AMPP algorithm significantly outperforms ASPP introducing only a marginal power consumption increase. In particular, we note that the AMPP QoS configuration delivers in all cases a File Transfer throughput very close to the one delivered when the P2P Group Owner is always active and at the same time an energy consumption very close to the ASPP one. The reason why AMPP is able to provide this improved performance is that it successfully identifies when the 3G link is being underutilized and in such cases schedules additional presence periods within Beacon intervals.

Path Delay Variation Experiment

Figures 7.12(a) and 7.12(b) depict the performance of the algorithms under study when varying the path delay, RTT_{base}, and fixing the buffer size in the NodeB to 30 packets per flow. Like in the previous experiment, AMPP significantly outperforms ASPP for all the considered 3G chanel models operating close to the performance of an always Active solution while providing energy efficiencies similar to



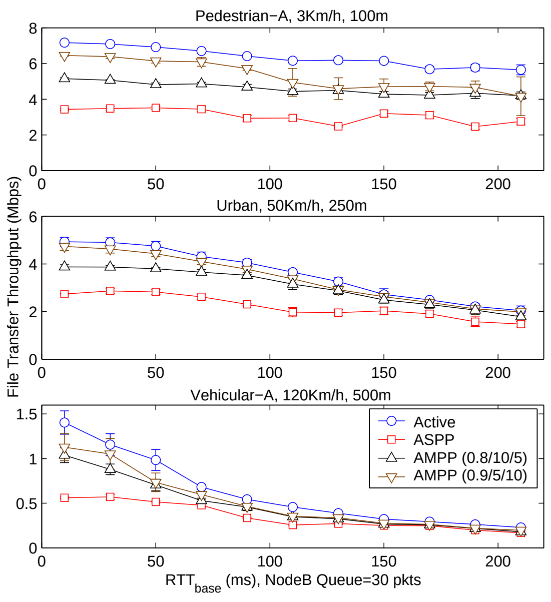
(a) Average File transfer throughput. (b) Average Energy spent by the P2P Group Owner in the file transfer.

Figure 7.11: File transfer performance when varying the buffering in the NodeB.

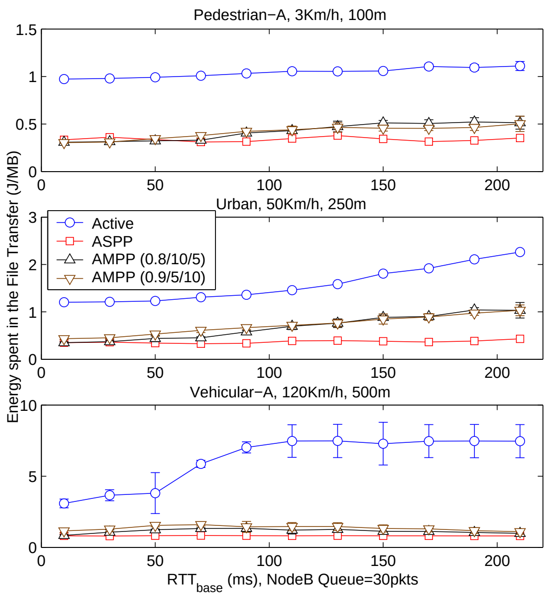
those of ASPP. The only exception is the case when RTT_{base} is large for the Pedestrian-A (good) and Typical Urban (average) channels. In these cases AMPP results in a higher energy consumption than ASPP, although still much lower than the one of the Active algorithm. The reason for a higher energy consumption in these cases is that TCP can not fill the path's bandwidth delay product and AMPP ends up operating with reduced intervals, i.e. $20ms$ in our experiments, with a small portion of the TCP congestion window being transmitted in each presence period.

Web Traffic Experiment

Finally, we complete our evaluation by studying in Figure 7.13 how AMPP performs with Web traffic. In the figure it can be observed how AMPP reduces the time to transfer a Web page compared to ASPP, specially when $RTT_{base} < 50ms$. The reason is that since Web traffic usually can not fill up the 3G link, $ratio$ falls below $ratio_{min}$ and AMPP decreases the used presence interval. However, when an interval smaller than RTT_{base}, i.e, the delay between the NodeB and the Web server, is selected, the P2P Group Owner schedules some presence periods where no data is transmitted and hence, AMPP increases the used presence interval because $n\_int\_no\_data > n\_int_{max}$. Therefore, when $RTT_{base} < 50ms$, the P2P Group Owner ends up scheduling a presence interval that ocillates around RTT_{base}, which



(a) Average File transfer throughput.



(b) Average Energy spent by the P2P Group Owner in the file transfer.

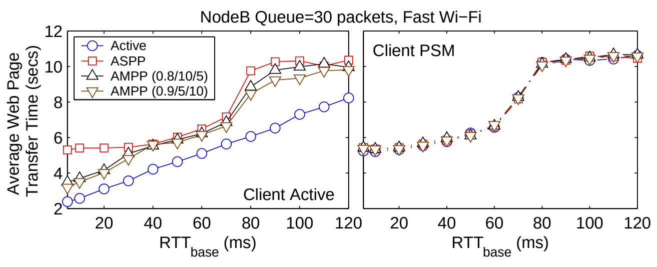
Figure 7.12: File transfer performance when varying the path delay (RTT_{base}).

is obviously a desirable behavior in the case of Web. If $RTT_{base} > 50ms$ AMPP often schedules its immediately higher interval which is $100ms$ resulting in a behaviour similar to ASPP. Regarding energy, we can see in Figure 7.13(b) how adapting the presence interval does not significantly increase the energy consumption of AMPP with respect to the one of ASPP.

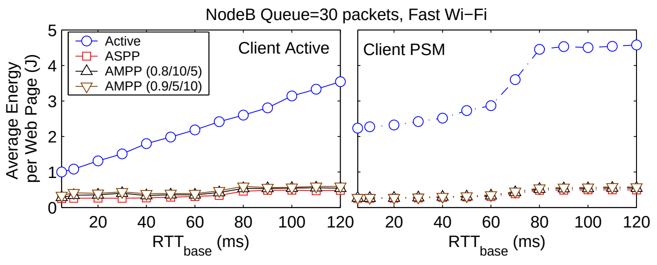
7.4 Summary and conclusions

Bringing device to device connectivity to a mass market is a key milestone in Wi-Fi's evolution roadmap. For this purpose, the Wi-Fi Alliance has recently developed the Wi-Fi Direct technology which should become the key device to device communication enabler. Among the different requirements to be fulfilled by this technology, battery usage efficiency is a central one due to the high penetration of Wi-Fi in mobile devices.

In this chapter we have analysed the two power saving protocols defined in Wi-Fi Direct allowing APs to save power, Opportunistic Power Save (OPS) and Notice of Absence (NoA), and designed two algorithms to efficiently use them: Adaptive Single Presence Period (ASPP) and Adaptive Multiple Presence Periods (AMPP). These algorithms allow a portable device implementing Wi-Fi Direct (e.g.



(a) Average Web page download time.



(b) Average Energy spent by the P2P Group Owner per Web page.

Figure 7.13: Web performance.

a mobile phone) to offer access to an external network (e.g. a cellular network) while addressing the trade-off between performance and energy consumption in a configurable manner. ASPP and AMPP performance has been thoroughly analyzed considering both their dynamic and steady state behaviour.

From our results the following conclusions can be drawn: i) ASPP and AMPP successfully manage to significantly reduce the power consumption of Wi-Fi Direct devices acting as Access Points (50-90%) without introducing a major user experience degradation, ii) the NoA protocol in combination with the AMPP algorithm delivers a close to optimal user experience and energy efficiency, and iii) AMPP's tuning parameters can be configured to prioritize either energy saving or user experience according to the device manufaturer preferences.

Appendix 7.A Revisiting the assumptions of the 3g bandwidth estimation algorithm

Consider in Figure 7.9 a presence period at time t_P with N packets in the P2P GO belonging to $M \geq 1$ TCP connections. Each packet, $i = 1...N$, will result in one or two (if this packet is the last of the current congestion window) new packets arriving at the NodeB at time $t^i_{arr} = t_P + t^i_{wifi} + RTT^i$, where RTT^i is the RTT experienced by packet i, excluding the time spent in the Wi-Fi network, and t^i_{wifi} is the time to transmit the TCP Data and TCP Ack in the Wi-Fi network. Consider $\{t^k_{arr}\}$ to be the ordered set of arrivals at the NodeB. Algorithm 7.1 assumes that the buffer at the NodeB never gets empty while processing these packets, which is true unless $\exists k$ such that:

$$t^k_{arr} - t^1_{arr} = \Delta^k_{wifi} + \Delta^k_{RTT} > \sum_{j=1}^{k-1} t^j_{3g}$$

Where t^j_{3g} is the time of transmitting packet j over the 3G link, $\Delta^k_{wifi} = t^k_{wifi} - t^1_{wifi}$ and $\Delta^k_{RTT} = RTT^k - RTT^1$. Given the fact that typically the Wi-Fi bandwidth is much higher than the 3G bandwidth, in general Δ^k_{wifi} is significantly smaller than $\sum_{j=1}^{k-1} t^j_{3g}$. In addition, Δ^k_{RTT} contains the RTT difference between two packets separated k positions in the set of arrivals at the NodeB. This difference can be assumed to be small if these packets belong to the same TCP connection and potentially high otherwise. However, it is interesting to notice that even packets belonging to different connections, as shown in (126), tend to get clustered in a bottleneck queue, which would favor the bursty behavior assumed in Figure 7.6. We have extensively evaluated whether this condition holds in practice in Figure 7.10(d), where a random RTT^i was introduced for each packet. Finally, notice in Figure 7.6 that the next burst of packets at the P2P GO will not arrive before $t_P + interval$, therefore in general only one or no inter-burst times will be present within a presence interval, which is the basic design assumption of Algorithm 7.1.

8

Conclusions and Future Work

Energy efficiency appears as one of the biggest challenges ahead to be solved by the Wi-Fi technology in order to continue with its successful development, and foster even more the advent of the mobile computing paradigm. Therefore, the goal of this thesis has been to develop a set of contributions that address the fundamental trade-off between QoS and energy efficiency in Wi-Fi networks. This trade-off has been addressed from a plurality of perspectives, namely real-time traffic, data traffic, distributed Wi-Fi protocols, centralized Wi-Fi protocols and even energy efficient Access Points.

In particular, our major contributions can be summarized as follows:

- Our first and second contributions in Chapters 3 and 4, target energy efficiency for real-time applications over distributed QoS and power saving protocols in Wi-Fi. In this context, we have proposed algorithms that decide how a station in power save mode should trigger the Access Point to recover its buffered data. By means of extensive performance evaluations we have shown that our proposed solutions improve upon existing solutions in the state of the art.
- Our third contribution in Chapter 5 targets real-time applications but considers centralized Wi-Fi QoS and power saving protocols. In this context we have proposed a novel scheduler running in the Access Point that spreads in time the service periods of the stations. We have shown that our proposed scheduler has a small complexity, and have analysed the benefits of this approach in terms of QoS, energy efficiency and admission capacity.
- Our forth contribution in Chapter 6 focuses again on distributed Wi-Fi QoS and power saving protocols, but this time from the perspective of data traffic. We have studied by means of analysis and simulations the detailed interactions between TCP and the power saving protocols in Wi-Fi, and have proposed a novel algorithm that configures the operation of the power saving protocol

according to the bottleneck bandwidth experienced by TCP connections. Our proposed algorithm significantly improves upon existing algorithms in the state of the art.

- Finally our fifth contribution in Chapter 7 has focused on the energy efficiency of Access Points in the context of the Wi-Fi Direct technology. In this context we have designed and evaluated two novel algorithms that balance the energy consumed by an Access Point with the QoS experienced by its associated clients.

Therefore, we conclude that the work in this thesis proves that significant improvements in energy efficiency are possible by designing smarter algorithms that only affect the devices with stringent power saving requirements. Thus, by following the design guidelines stated in Chapter 2: i) focus on MAC layer solutions, ii) design within the limits of current standards, and iii) favor client side solutions, the algorithms proposed throughout this thesis represent a realistic path forward to improve energy efficiency in Wi-Fi networks.

We would like to finalize this thesis by outlining possible lines of future work. In particular each of the individual algorithms contributed in the different chapters of this thesis naturally leads to further investigations which we detail next:

- The adaptive triggering algorithm presented in Chapter 3 and the CA-DFA algorithm introduced in Chapter 4 should be combined and evaluated together, providing a solution where a Wi-Fi station in power save mode adapts its polling interval to both the characteristics of the applications and the amount of congestion in the network.

- An interesting line for future work would be to study the interactions between the CA-DFA algorithm introduced in Chapter 4 and the PSMP protocol defined in 802.11n or the Multi User MIMO capabilities that will be included in 802.11ac. PSMP or Multi User MIMO benefit from having many users to be served at a given point in time, which is less likely to happen if stations use large aggregation intervals.

- The DRA scheduler introduced in Chapter 5 could be extended in the following ways. First, the initial work on admission control introduced in Chapter 5 should be extended to further assess the impact of DRA in the admission region of a network with QoS and power consumption guarantees. Appropriate admission control algorithms should be designed for that purpose. Second, algorithms could be devised that, by appropriately pre-sorting the flows to be processed by DRA, improve even further the achieved flow separation. Third, DRA could be extended to allow a set of overlapping co-channel APs to provide QoS and power saving guarantees in a distributed way[1]. Finally, it would be interesting to study how a new protocol like PSMP can enhance the performance of a grouping scheduler by advertising an appropriate schedule at the beginning of each group allocation period.

- A path for future work regarding the study presented in Chapter 6 is to further understand the fairness interactions between TCP flows in active mode and TCP flows in power saving, and how BA-TA can be used to mitigate potential unfairness.
- The power saving algorithms introduced in Chapter 7 could be improved in the following way. For instance, algorithms could be devised to adaptively tune the multiple parameters involved in AMPP based on remaining battery capacity. In addition, other algorithms could be devised that control the fraction of bottleneck bandwidth obtained by the applications running directly in the P2P Group Owner and the fraction delivered to the associated P2P Clients.

Finally, a clear line of future work is the generalization of the methods and algorithms proposed in this thesis for technologies other than Wi-Fi. For instance, cellular technologies like WiMAX and LTE, or even sensor networks like 802.15, all have power saving mechanisms that resemble those available in Wi-Fi. It should therefore be possible to adapt the algorithms and insights presented in this thesis, e.g. the TCP model developed in Chapter 6, for these technologies.

[1]Notice that DRA can already be directly applied to the Overlapping-BSS mechanisms being proposed in 802.11aa (117).

List of Figures

1.1 Widening Energy Gap in Mobile Computing 2

1.2 Wi-Fi IC Shipments by Device Type, In Millions 3

1.3 Impact of Wi-Fi on battery life 4

1.4 Power Consumption Figures in current Wi-Fi chipsets. Data from (23), (24), (25). . . . 5

1.5 Power vs Time during a VoIP call over Wi-Fi 6

2.1 Wi-Fi Architecture 10

2.2 Concurrent Backoff Entities in EDCA 11

2.3 Contention Parameters of each AC 12

2.4 Sample of HCCA operation during the CFP and the CP 13

2.5 Overview of the A-MSDU and the A-MPDU aggregation schemes in 802.11n, ... 14

2.6 Example of Standard PSM operation. The station awakes to listen to every Beacon, Listen Interval=1, and discovers with the TIM bit if there is buffered data for it in the AP. If that is the case the station triggers the transmission of buffered frames using the PS-Poll frame. The station can awake at any time to perform uplink transmissions. . . 15

2.7 Example of U-APSD operation. U-APSD configuration: AC_VO and AC_VI both trigger- and delivery-enabled. AC_BE and AC_BK neither delivery- nor trigger-enabled (i.e., use legacy 802.11 power save mode). 17

2.8 S-APSD example of operation. S-APSD configuration: VoIP traffic stream in the downlink configured to use S-APSD with HCCA access mode, AC_VI traffic in the downlink uses S-APSD with EDCA access mode, AC_BE and AC_BK configured to use legacy 802.11 power save mode. 18

2.9 Example of PSMP operation 19

3.1 DL stream's delay/jitter under a constant polling interval, Δ_{trig}. 34

3.2 Data and signaling trigger generation employing re-scheduling. 34
3.3 Evolution of $\epsilon(n)$ along the error function. 43
3.4 Timing between consecutive events. 44
3.5 Δ_{trig} dynamics. 46
3.6 Behavior of the algorithm when relaxing the assumptions of the model. 51
3.7 Dynamics of the adaptive algorithm with a VoIP codec with silence suppression (upper graph) and with a Video codec (lower graph). 55
3.8 Dynamics of the adaptive algorithm with TCP traffic. 57
3.9 Dynamics of the adaptive algorithm with multiple simultaneous applications. 58
3.10 Overall Network Performance. 62
3.11 Effect of congestion on the Adaptive Algorithm. 65

4.1 QoS sensitive Applications. Performance of Voice and Video stations when increasing the number of station clusters in the network, in terms of Power Consumption (upper graph), CDF95 Downlink Delay (middle graph) and Average Downlink Throughput after the de-jitter buffer (lower graph). 72
4.2 Overhead incurred by the power saving protocols under study. The upper graph depicts signaling overhead, and the lower one channel access overhead (collisions). 74
4.3 Non-QoS sensitive Applications. Performance of Web and FTP stations when increasing the number of Clusters in the network, in terms of Power Consumption (upper graph) and Average Web Page Delay/Average FTP Connection Throughput (lower graph). . . 75
4.4 CA-DFA Dynamics-1. 87
4.5 CA-DFA Dynamics-2. 90
4.6 QoS sensitive Applications. Effect of CA-DFA on average power consumption (upper graph) and worst case delay (lower graph) for Voice and Video applications. 92
4.7 CA-DFA dynamics. Variation of the used SI as congestion increases, and performance gains with different deployment scenarios. 93
4.8 Non-QoS sensitive Applications. Effect of CA-DFA on average power consumption (upper graph) and performance (lower graph) of the Web and FTP applications in a multi-service Wi-Fi network. 95

5.1 Example of HCCA allocations with a *Grouping* (upper graph) and a *Distribution* (lower graph) approach. 100
5.2 Distances between resource allocations. 101

5.3 Distances between release times. 104
5.4 Minimum *left* and *right* effective distances. 106
5.5 Minimum *left* and *right* distances. 107
5.6 Resulting distributions with a max-min criteria. 109
5.7 Performance of DRA. 111
5.8 Performance of HCCA stations. 116
5.9 Performance of EDCA Voice stations. 118
5.10 Performance of EDCA FTP stations. 120
5.11 F function implementation 122
5.12 Definition of a flow. Notice that since its release time, a flow can be scheduled within a time D_j in order to avoid deadline violations. 125
5.13 Considered HCCA architecture. 127
5.14 Admission region using an EDF scheduler and a perfect Admission Control. 129

6.1 Scenario under study. 133
6.2 Buffer Experiment. For each of the algorithms and scenarios under study, throughput is depicted on the left graph and energy on the right one. Notice that the same legend applies to all sub-figures. 136
6.3 TCP dynamics. The upper graph depcits the *Slow DSL* scenario, and the lower one the *Fast DSL* scenario. 137
6.4 ACK compression in a TCP connection, caused by Wi-Fi power saving protocols. . . . 139
6.5 RTT Experiment. For each of the algorithms and scenarios under study, throughput is depicted on the left graph and energy on the right one. Notice that the same legend applies to all sub-figures. 142
6.6 TCP model. The two graphs depict the data arrivals from the DSLAM to the AP, the Beacon transmissions from the AP to the Wi-Fi station, and the delivery of the buffered data after the Beacon. 144
6.7 TCP dynamics. 145
6.8 Intuition behind the Peak Bandwidth Estimation Algorithm. 148
6.9 Basic BA-TA Dynamics. The figures represent the variation over time of BA-TA used interval (black line), the $instant\_thr$ estimation (blue line), and the $peak\_rate$ estimation (red line). 154

6.10 Basic BA-TA Dynamics. The figures represent the variation over time of BA-TA used interval (black line), the $instant\_thr$ estimation (blue line), and the $peak\_rate$ estimation (red line). 155

6.11 Effect of BA-TA parameters. The upper part of each figure depicts the *Fast DSL* scenario, and the lower part of each figure depicts the *Wi-Fi Bottleneck* scenario. In addition, notice that the depicted throughput and energy metrics are normalized. 157

6.12 BA-TA dynamics with multiple flows. 158

6.13 BA-TA Extensive Evaluation. The upper part of each graph depicts the results for the Buffer Experiment, and the lower part of each graph depicts the results for the RTT Experiment. In addition, notice that the depicted throughput and energy metrics are normalized. 159

6.14 BA-TA performance with Web Traffic. 160

7.1 Our target scenario. A mobile phone sharing access to a 3G network with a set of connected devices using Wi-Fi Direct. 166

7.2 Example operation of the Wi-Fi Direct power saving protocols for a P2P Group Owner. 168

7.3 Sample of PHY rate variation in the HSDPA Channels considered in our evaluation. . . 173

7.4 Dynamics of the *ASPP* algorithm. 175

7.5 File transfer performance when varying the buffering in the NodeB. For each figure the left column represents the results when using the Wi-Fi AC_VI configuration, and the right column the results when using the Wi-Fi AC_BK configuration. 179

7.6 Dynamics of a TCP transfer when the P2P Group Owner runs a power saving protocol. Notice that the two P2P Clients illustrate a different case: the upper one a case where the file is transferred over AC_VI, and hence the P2P Group Owner can use smaller presence periods, and the lower one a case where the file is transferred over AC_BK and so the P2P Group Owner uses larger presence periods. 180

7.7 File transfer performance when varying the path delay (RTT_{base}). For each figure the left column represents the results when using the Wi-Fi AC_VI configuration, and the right column the results when using the Wi-Fi AC_BK configuration. 182

7.8 Web traffic performance. 183

7.9 Insight on the bandwidth estimation algorithm: From the perspective of the P2P Group Owner, TCP data packets arrive in bursts modulated by the bandwidth of the 3G link, interleaved with inter-burst times that appear when the buffer in the NodeB becomes empty. One can notice in the figure how within a presence interval, there can only be one inter-burst time. 186
7.10 Dynamics of the *AMPP* algorithm. 192
7.11 File transfer performance when varying the buffering in the NodeB. 196
7.12 File transfer performance when varying the path delay (RTT_{base}). 197
7.13 Web performance. 198

References

[1] B. Chalamala, *Portable electronics and the widening energy gap.* Proceedings of the IEEE, Vol.95, No. 11, November 2007.[1] 1, 2

[2] Intel, *The Evolution of a Revolution*, available at *http://download.intel.com/pressroom/kits/IntelProcessorHistory.pdf*. 2

[3] *Hard drive capacity over time*, available at *http://commons.wikimedia.org*. 2

[4] R. Wang, *Enabling Mobile Broadband Internet*, Codio Link White Paper, July 2009. 2

[5] R. Van Nee, *Breaking the Gigabit-per-second barrier with 802.11ac*, Wireless Communications, IEEE , vol.18, no.2, pp.4, April 2011. 2

[6] M. Broussely, G. Archdale, *Li-ion batteries and portable power source prospects for the next 5-10 years*, Journal of Power Sources, Volume 136, Issue 2, Selected papers presented at the International Power Sources Symposium, 1 October 2004, Pages 386-394. 2

[7] T.E. Starner. *Powerful change part 1: batteries and possible alternatives for the mobile market.* Pervasive Computing, IEEE, 2(4):8688, Oct.-Dec. 2003. 1

[8] Wi-Fi Alliance, *Thirsty for bandwidth, consumers and carriers embrace Wi-Fi phones*, March 23rd 2010, available at *http://www.wi-fi.org*. 2

[9] IEEE 802.11 WG, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11, June 1999. 2, 10, 14

[10] Wi-Fi Alliance, http://www.wi-fi.org. 2, 11

[1]Notice that the numbers appearing at the end of each reference are links, in the electronic version of this document, to the pages where this reference is cited.

[11] Wi-Fi Alliance, P2P Technical Group, *Wi-Fi Peer-to-Peer (P2P) Technical Specification v1.0*, December 2009. 2, 19, 166, 172

[12] Wi-Fi Alliance, *Thirsty for bandwidth, consumers and carriers embrace Wi-Fi phones*, March 23rd 2010, available at *http://www.wi-fi.org*. 2

[13] IEEE 802.11 WG, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band*, IEEE Standard 802.11ad/D4.0, July 2011. 3

[14] IEEE 802.11 WG, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 5: Enhancements for Very High Throughput for Operation in Bands below 6 GHz*, IEEE Standard 802.11ac/D1.0, May 2011. 3

[15] IEEE 802.11 WG, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 3: TV White Spaces Operation*, IEEE Standard 802.11af/D1.02, June 2011. 3

[16] IEEE 802.11 TGah group, *http://www.ieee802.org/11/Reports/tgah_update.htm*. 3

[17] HTC Hero, *http://www.htc.com/de/product/hero/specification.html* 4

[18] R. Helgason, E. A. Yavuz, S. T. Kouyoumdjieva, L. Pajevic, and G. Karlsson. *A mobile peer-to-peer system for opportunistic content-centric networking*. In Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds (MobiHeld '10). ACM, New York, NY, USA, 21-26. 3

[19] E. Nightingale, M. Anand and J. Flinn. *Self-tuning wireless network power management.* Wireless Networks 11(4),451469, July 2005. 3, 25, 26

[20] V. Raghunathan C. Schurgers and M. Srivastava. *Power management of energy-aware communication systems.* ACM Trans. Embedded Comp. Systems 2(3),431447, August 2003. 3

[21] Atheros Communications, *Power Consumption and Energy Efficiency Comparisons of WLAN Products*, 2003. 4

[22] B. McFarland, *Power-reduction details determine a chips overall power consumption (part 2 of 2).* http://www.wirelessnetdesignline.com, March 2008. 5, 6

[23] Atheros AR6001, available at *http://www.atheros.com*. 5, 174, 205

[24] *Broadcom 4311AG 802.11a/b/g*, available at *http://www.broadcom.com*. 5, 70, 134, 205

[25] *Intel 3945 a/b/g*, data-sheet available at *http://h18000.www1.hp.com*. 5, 205

[26] D. Camps-Mur, X. Perez-Costa, S. Sallent Ribes, *An adaptive solution for Wireless LAN distributed power saving modes*, Computer Networks, Volume 53, Issue 18, 24 December 2009, Pages 3011-3030. 7, 30

[27] D. Camps Mur, M. Gomony, X. Perez-Costa and S.Sallent, *Leveraging 802.11n Frame Aggregation to enhance QoS and Energy Efficiency in Wi-Fi Networks*, submitted to Elsevier Computer Networks. 7, 68

[28] D. Camps-Mur, X. Pérez-Costa, V. Marchenko and S. Sallent-Ribes, *On centralized schedulers for 802.11e WLANs distribution versus grouping of resources allocation*. Wireless Communications and Mobile Computing, 11: n/a. doi: 10.1002/wcm.1046, 2011. 8, 98

[29] D. Camps Mur and S. Sallent, *Enhancing the performance of TCP over Wi-Fi Power Saving Mechanisms*, submitted to IEEE Transactions on Mobile Computing. 8, 132

[30] D. Camps-Mur, X. Perez-Costa, S. Sallent-Ribes, *Designing energy efficient access points with Wi-Fi Direct*, Computer Networks, available online 22 June 2011, ISSN 1389-1286, DOI: 10.1016/j.comnet.2011.06.012. 8, 165

[31] IEEE 802.11 group, *http://www.ieee802.org/11/* 9

[32] IEEE 802.11 WG, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements*, IEEE Standard 802.11e, November 2005. 10, 14, 54, 97, 98, 112

[33] IEEE 802.11 WG, *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Std 802.11-2007*. 9, 18, 20, 70, 102

[34] IEEE Computer Society, *802.3: IEEE Standard for Local and Metropolitan Area Networks Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*, 2005. 9

[35] Wi-Fi Alliance, Quality of Service (QoS) Task Group, *Wi-Fi Multimedia (including WMM Power-Save) Specification v1.1*, 2005. 11, 17, 97, 172

[36] IEEE 802.11 WG, *IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput.* 13, 177

[37] T.K. Paul, T. Ogunfunmi, *Wireless LAN Comes of Age: Understanding the IEEE 802.11n Amendment*, Circuits and Systems Magazine, IEEE , vol.8, no.1, pp.28-54, First Quarter 2008. 14

[38] E. Tan, L. Guo, S. Chen and X. Zhang, *PSM-throttling: Minimizing Energy Consumption for Bulk Data Communications in WLANs*, Network Protocols, 2007. ICNP 2007. IEEE International Conference on , vol., no., pp.123-132, 16-19 Oct. 2007. 25, 26

[39] R.Krashinsky and H.Balakrishnan, *Minimizing energy for wireless web access with bounded slowdown*, Wireless Networks archive Volume 11 , Issue 1-2 (January 2005). 24, 26, 56, 57, 59, 76, 120, 132, 180, 181

[40] D.Qiao and K.G.Shin, *Smart Power Saving Mode for IEEE 802.11 Wireless LANs*, Proceedings of IEEE INFOCOM, March 2005. 24, 26

[41] G. Anastasi, M. Conti, E. Gregori, A. Passarella *802.11 powersaving mode for mobile computing in WiFi hotspots: limitations, enhancements and open issues*, Wireless Networks Volume 14 , Issue 6 (December 2008). 25, 26

[42] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck and X. Zhang. *Delving into internet streaming media delivery: a quality and resource utilization perspective.* In IMC 06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, pages 217230, New York, NY, USA, 2006. ACM. 25

[43] S. Chandra, *Wireless network interface energy consumption, implications for popular streaming formats.* ACM Multimedia Systems Journal, Springer-Verlag, 9(2):185201, August 2003. 23, 26

[44] Y. Wei, S.M. Bhandarkar and S. Chandra, *A client-side statistical prediction scheme for energy aware multimedia data streaming*, Multimedia, IEEE Transactions on , vol.8, no.4, pp.866-874, Aug. 2006. 23, 26

[45] Y.Chen and S.Emeott, *Investigation into Packet Delivery Options for WLAN Acess Points Implementing Unscheduled Power Save Delivery*, Proceedings of IEEE Globecom, November 2005. 23, 26

[46] S. Takeuchi, K. Sezaki and Y. Yasuda, *Quick Data-Retrieving for U-APSD in IEEE802.11e WLAN Networks*. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E89-A, 7 (July 2006), 1919-1929. 23, 26

[47] X.Pérez-Costa, D.Camps-Mur and A. Vidal. *SU-APSD: Static IEEE 802.11e Unscheduled Automatic Power Save Delivery*. Computer Networks Volume 51, Issue 9, 20 June 2007, Pages 2326-2344. 23, 26, 33, 70, 71, 114, 134, 153

[48] G. Ciccarese, G. Convertino, M. De Blasi, S. Elia, C. Palazzo, and L. Patrono. *A novel apsd scheduler for wlan ieee 802.11e.* Communication Systems, Networks and Digital Signal Processing, Symposium (CSNDSP), July 2006. 23, 26

[49] G. Boggia, P. Camarda, F. A. Favia, L. A. Grieco, and S. Mascolo. *Providing delay guarantees and power saving in ieee 802.11e network.* Lecture Notes in Computer Science, Springer, pages 323 - 332, May 2005. 24, 26

[50] B. Otal and J. Habetha, *Power saving efficiency of a novel packet aggregation scheme for high-throughput WLAN stations at different data rates*, Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st , vol.3, no., pp. 2041- 2045 Vol. 3, 30 May. 22, 26

[51] *Google wants to fix its Android fragmentation problem*, available at *http://www.digitaltrends.com*. 27

[52] H. Schwarz, D. Marpe and T. Wiegand, *Overview of the Scalable Video Coding Extension of the H.264/AVC Standard*, Circuits and Systems for Video Technology, IEEE Transactions on , vol.17, no.9, pp.1103-1120, Sept. 2007. 30

[53] L.D. Cicco, S. Mascolo, V. Palmisano, *Skype Video Congestion Control: an Experimental Investigation*, Computer Networks (2010). 30

[54] Y. Li, T.D. Todd and D. Zhao, *Access Point Power Saving in Solar/Battery Powered IEEE 802.11 ESS Mesh Networks*, Second International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine), 2005. 25, 26, 170

[55] T.D. Todd, A.A. Sayegh, M.N. Smadi and D. Zhao, *The need for access point power saving in solar powered WLAN mesh networks*, IEEE Network, May-June 2008. 25, 26

[56] K. Miyuki, *Radio Communication Systems*, Patent from Sony Corp., JP 2004336401. 25, 26

[57] S.Mangold and S.Choi and G.R.Hiertz and O.Klein and B.Walke, *Analysis of IEEE 802.11e for QoS Support in Wireless LANs*, IEEE Wireless Communications Magazine, December 2003. 11

[58] X.Pérez-Costa, D.Camps-Mur and T.Sashihara. *Analysis of the Integration of IEEE 802.11e Capabilities in Battery Limited Mobile Devices*. IEEE Wireless Communications Magazine (WirComMag), special issue on Internetworking Wireless LAN and Cellular Networks, Volume 12, Issue 6, December 2005. 23, 26

[59] UPnP Forum, *UPnP QoS Architecture V2.0*, October 2006. 20, 30

[60] BonJour, available at *http://www.apple.com/support/bonjour/*. 20

[61] X. Pérez-Costa, and D. Camps-Mur. *A protocol Enhancement for IEEE 802.11 Distributed Power Saving Mechanisms. No Data Acknowledgement*. IST summit 2007, Budapest, July 2007. 37

[62] A. Hyvarinen, J. Karhunen and E. Oja, *Independent Component Analysis*, Whiley & Sons, inc. 38

[63] ITU-T Recommendation Y.1541. 52, 53

[64] OPNET Technologies, *http://www.opnet.com*. 53, 69, 113, 133, 172

[65] F.H.P. Fitzek and M.Reisslein, *MPEG-4 and H.263 Video Traces for Network Performance Evaluation*, IEEE Network, Vol. 15, No. 6, pages 40-54, November/December 2001. 53, 69, 100, 113

[66] Cisco Aironet 802.11a/b/g Wireless CardBus Adapter, *http://www.cisco.com/*. 61, 114

[67] Wi-Fi Alliance, *Wi-Fi Alliance releases Smart Grid report; creates new task group to address energy management issues*, November 11th, 2009, available at *http://www.wi-fi.org*. 3

[68] X.Pérez-Costa and D.Camps-Mur. *IEEE 802.11e QoS and Power Saving Features: Overview and Analysis of Combined Performance*. IEEE Wireless Communications Magazine (WirComMag), Volume 17, Issue 4, August 2010. 24, 26

[69] Wi-Fi Alliance, *Wi-Fi Alliance releases new Wi-Fi Certified n product data*, September 30th, 2010, available at *http://www.wi-fi.org*. 13

[70] D. Skordoulis, N. Qiang, C. Hsiao-Hwa, A.P. Stephens, C. Liu and A. Jamalipour, *IEEE 802.11n MAC frame aggregation mechanisms for next-generation high-throughput WLANs*, Wireless Communications, IEEE , Vol.15, No.1, pp.40-47, February 2008. 13

[71] IEEE 802.11 WG, *IEEE Standard for Information Technology Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks Specific Requirements Part 11: Wireless Lan Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Radio Resource Measurement of Wireless Lans," IEEE Std 802.11k-2008*. 82, 83

[72] V. Namboodiri and L. Gao, *Energy-Efficient VoIP over Wireless LANs*, Mobile Computing, IEEE Transactions on , Vol.9, No.4, pp.566-581, April 2010. 23, 26

[73] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin and R. Gupta, *Wireless wakeups revisited: energy management for voip over wi-fi smartphones*. In Proceedings of the 5th international conference on Mobile systems, applications and services (MobiSys '07). ACM, New York, NY, USA, 179-191. 23, 26

[74] D. Bertozzi, A. Raghunathan, L. Benini and S. Ravi, *Transport protocol optimization for energy efficient wireless embedded systems*. Design, Automation and Test in Europe Conference and Exhibition, 2003 , vol., no., pp. 706- 711, 2003. 25, 26

[75] V. Baiamonte and C.F. Chiasserini, *Saving energy during channel contention in 802.11 WLANs*, Mobile Networks and Applications 11 (2) (2006), pp. 287296. 22, 26

[76] L. Bononi, M. Conti and L. Donatiello, *A distributed mechanism for power saving in IEEE 802.11 wireless LANs*, Mobile Networks and Applications 6 (3) (2001), pp. 211 - 222 22, 26

[77] R. Bruno, M. Conti and E. Gregori, *Optimization of efficiency and energy consumption in P-persistent CSMA-based wireless LANs*, IEEE Transactions on Mobile Computing 1 (1) (2002), pp. 10-31. 21, 26

[78] J.R. Hsieh, T.H. Lee and Y.W. Kuo, *Energy-efficient multi-polling scheme for wireless LANs*, IEEE Transactions on Wireless Communications 8 (3) (2009), pp. 1532 - 1541. 22, 26

[79] J.A. Stine and G. De Veciana, *Improving energy efficiency of centrally controlled wireless data networks*, Wireless Networks 8 (6) (2002). 22, 26

[80] Y. He and R. Yuan, *A novel scheduled power saving mechanism for 802.11 wireless LANs*, IEEE Transactions on Mobile Computing 8 (10) (2009). 22, 26

[81] J.R. Lee, S.W. Kwon and D.H. Cho, *Adaptive beacon listening protocol for a TCP connection in slow-start phase in WLAN*, IEEE Communications Letters 9 (9) (2005), pp. 853 - 855. 25, 26

[82] S.L. Tsao and C.H. Huang, *Energy-efficient transmission mechanism for VoIP over IEEE 802.11 WLAN*, in: Wireless Communications and Mobile Computing, 10.1002/wcm.747. 23, 26

[83] C. Zhu, H. Yu, Xinbing Wang and H. Chen, *Improvement of Capacity and Energy Saving of VoIP over IEEE 802.11 WLANs by A Dynamic Sleep Strategy*, IEEE GLOBECOM09 (2009). 23, 26

[84] Y. Lin and V.W.S. Wong; *WSN01-1: Frame Aggregation and Optimal Frame Size Adaptation for IEEE 802.11n WLANs*, Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE , pp.1-6, Nov. 27 2006-Dec. 1 2006. 13

[85] Ralink Technology, *Ralink Unveils Next Generation Dual-Band 2x2 802.11n/Bluetooth 3.0+HS Combo Module*, *http://www.ralinktech.com*. 69

[86] A. Corlett, D.I. Pullin and S. Sargood, *Statistics of One-Way Internet Packet Delays*, 53rd IETF, Minneapolis, March 2002. 54, 69, 135

[87] S.Yun, H.Kim, H.Lee and I.Kang, *100+ VoIP streams on 802.11b: The power of combining voice frame aggregation and uplink-downlink bandwidth control in wireless LANs*, IEEE Journals on Selected Areas in Communications (JSAC), Vol. 25, Issue 4, pp. 689-698, May 2007. 67, 76, 77, 78

[88] D. Akhmetov, *802.11n OPNET model*, available at *http://www.opnet.com*. 69

[89] IEEE 802.11 WG, *Wireless LAN's Usage Models*, doc.: IEEE 802.11-03/802r23, May 2004. 69

[90] V. Erceg, et.al., *TGn Channel Models*, IEEE P802.11 Wireless LANs, Tech. Rep., May 2004. 69

[91] *Average Web Page Size Triples Since 2003* available at *http://www.websiteoptimization.com/*. 69, 133, 154, 175, 181

[92] X. Li and Q.A. Zeng, *Capture Effect in the IEEE 802.11 WLANs with Rayleigh Fading, Shadowing and Path Loss*, Wireless and Mobile Computing, Networking and Communications, 2006 (WiMobapos 2006). IEEE International Conference on Volume, Issue, 19-21 June 2006 Page(s):110 - 115. 74

[93] Wi-Fi Alliance, *Delivering the Best User Experience with Voice over Wi-Fi*, 2008. 85

[94] L. Changwen, A.P. Stephens, *Delayed Channel Access for 802.11e Based WLAN*, IEEE International Conference on Communications'06, vol.10, pp.4811-4817, June 2006. 77

[95] D. Skordoulis, Q. Ni, G. Min, K. Borg, *Adaptive Delayed Channel Access for IEEE 802.11n WLANs*, IEEE International Conference on Circuits and Systems for Communications (ICCSC) 2008, Shangai, China, May 2008. 77

[96] D. Skordoulis, Q. Ni, C. Zarakovitis, *A Selective Delayed Channel Access (SDCA) for the high-throughput IEEE 802.11n*, IEEE Wireless Communications and Networking Conference (WCNC), pp.1-6, 5-8, April 2009. 77

[97] Alberto Medina, Mark Allman, and Sally Floyd. *Measuring the evolution of transport protocols in the internet*. SIGCOMM Comput. Commun. Rev. 35, 2 (April 2005), 37-52. 133

[98] Sangtae Ha, Injong Rhee, and Lisong Xu. *CUBIC: a new TCP-friendly high-speed TCP variant*. SIGOPS Oper. Syst. Rev. 42, 5 (July 2008), 64-74. 133

[99] Akamai, *The State of the Internet. 3rd Quarter 2010*, available at *http://www.akamai.com/stateoftheinternet*, 2010. 133

[100] P. Acharya, A. Sharma, E.M. Belding, K.C. Almeroth, K. Papagiannaki, *Rate Adaptation in Congested Wireless Networks through Real-Time Measurements*, Mobile Computing, IEEE Transactions on , Vol.9, No.11, pp.1535-1550, Nov. 2010. 152

[101] T. Li, D. Leith and D. Malone, *Buffer sizing for 802.11-based networks*, IEEE/ACM Trans. Netw. 19, 1 (February 2011), 156 - 169. 135

[102] G.F. Franklin, J.D. Powell and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, Adison-Wesley. 149

[103] S. Floyd, T. Henderson and A. Gurtov, *The NewReno Modification to TCP's Fast Recovery Algorithm*, RFC 3782, 1999. 140

[104] T. Ott, J. Kemperman and M. Mathis, *Window Size Behavior in TCP/IP with Constant Loss Probability*, DIMACS Workshop on Performance of Realtime Applications on the Internet, Plainfield NJ, November 6-8, 1996. 131

[105] Wi-Fi Alliance, *Enthusiasm for Wi-Fi in consumer electronics continues to grow*, available at *http://www.wi-fi.org*. 97

[106] X.Pérez-Costa, D.Camps-Mur, J.Palau, D.Rebolleda and S.Akbarzadeh, *Overlapping Aware Scheduled Automatic Power Save Delivery Algorithm*, European Wireless Conference, Paris, April 2007. 99, 101, 110

[107] G. Bianchi, A. Di Stefano, C. Giaconia, L. Scalia, G. Terrazzino and I. Tinnirello, *Experimental Assessment of the Backoff Behavior of Commercial IEEE 802.11b Network Cards*, INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, pp.1181-1189, 6-12 May 2007. 97

[108] P. Ansel, Q. Ni and T. Turletti, *FHCF: a simple and efficient scheduling scheme for IEEE 802.11e*, Springer/Kluwer Journal on Mobile Networks and Applications (MONET), Vol. 11, 2006. 99

[109] Q.Zhao and D.Tsang, *An Equal-Spacing-Based Design for QoS Guarantee in IEEE 802.11e HCCA Wireless Networks*, IEEE Transactions on Mobile Computing, Issue 99, 2008. 99

[110] A.Grilo, M.Macedo and M.Nunes, *A Scheduling Algorithm for QoS Support in 802.11e Networks*, IEEE Wireless Communications, June 2003. 99

[111] S. Shenker, C. Partridge, and R. Guerin, *Specification of guaranteed quality of service*, RFC 2212, Internet Engineering Task Force, Sept. 1997. 12

[112] Sha et.al., *Real Time Scheduling Theory: A Historical Perspective*, Real-Time Systems Volume 28, Issue 2-3 (November-December 2004). 103

[113] L. Georges, P. Mhlethaler and N. Rivierre, *A Few Results on Non-Preemptive Real time Scheduling*, INRIA Research Report nRR3926, 2000. 103, 125

[114] Y.Higuchi et.al., *Delay Guarantee and Service Interval Optimization for HCCA in IEEE 802.11e WLANs*, Wireless Communications and Networking Conference, WCNC 2007. 114

[115] L. Berlemann, *Distributed Quality-of-Service Support in Cognitive Radio Networks*, PhD Thesis, February 2006. 99

[116] J. Goossens, *Scheduling of Offset Free Systems*, Real-Time Systems, 5, 1-26, 1997. 110, 127

[117] IEEE 802.11 WG, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 2: MAC Enhancements for Robust Audio Video Streaming*, IEEE Standard 802.11aa/D6.0, July 2011. 97, 203

[118] S. Baruah, L. Rosier and R. Howell, 1990b. *Algorithms and complexity concerning the preemptive scheduling of periodic real-time tasks on one processor*, The Journal of Real-Time Systems 2. 126

[119] R. Pellizzoni and G. Lipari, *Feasibility Analysis of Real-Time Periodic Tasks with Offsets*, Real-Time Systems Journal, 2005, Vol. 30(1-2): 105-128. 126

[120] IEEE 802.11-2007 Standard, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2007. 177

[121] J. Lorchat and T. Noel, *Energy saving in IEEE 802.11 communications using frame aggregation*, in: IEEE Global Telecommunications Conference, vol. 3, 2003, pp. 1296 - 1300. 22, 26

[122] *Enhanced UMTS Radio Access Network Extensions for NS2 (EURANE)*, available at *http://eurane.ti-wmc.nl/eurane*. 172

[123] J.C. Mogul, *Observing TCP Dynamics in Real Networks*, ACM Sigcomm Computer Communication Review, Volume 22 Issue 4, Oct. 1992. 185

[124] *3GPP TS 25.321 V6.5.0*, Technical Specification Group Radio Access Network; Medium Access Control (MAC) protocol specification (Release 6), 2006. 188

[125] Grimmett and Stirzaker, *Probability and Random Processes*, Oxford Science Publications, ISBN 0 19 853665 8, Problem 7.11, proof available at: http://www.mcdowella.demon.co.uk/Chebyshev.html. 187

[126] S. Shenker, L. Zhang and D. Clark, *Some Observations on the Dynamics of a Congestion Control Algorithm*, Computer Communication Review 20(5):30-39, October,1990. 199

[127] List of deployed HSDPA networks worldwide, available at *http://en.wikipedia.org/wiki/List_of_HSDPA_networks*. 169