AlphaAP: Achieving Weighted Throughput in Virtualised WLANs

Ronan Turner

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2016

Abstract

Mobile data usage is ever increasing. A growing number of users are consuming content on the move, and cellular providers are increasingly looking to newer methods for handling this demand. One such method uses hotspots found nearby users to offload data usage via Wireless Local Area Networks (WLAN). However, to meet this demand network providers must roll out large amounts of infrastructure to achieve suitable coverage, and growing infrastructure in this manner may be costly and impractical.

Many providers are instead looking to virtualisation as the answer. Virtualisation allows providers to use existing network infrastructure - such as routers present in cafes and airports. These techniques suggest a new approach to network devices. Typically, network devices implement functionality at the hardware level; fixed and inaccessible. Network virtualisation encourages exposing this functionality through a clear software interface, allowing programmability of these functions. This separates the operation of network devices into infrastructure providers, and service providers who use this infrastructure. Service providers can either share the cost of new infrastructure or pay a premium to be added to existing infrastructure. This is already being trialed in locations such as airports, where large numbers of users wish to remain connected via their providers.

Recent research has identified unfairness in virtualisation of WLANs, where VAPs with greater numbers of stations (STA) achieve a greater share of throughput. Related research introduces new Medium Access Control (MAC) layer protocols to solve this issue.

In this thesis we work on the foundation of an existing protocol, CVAP, to create AlphaAP, which supports assigning weighted throughput guarantees to the Virtual Access Points (VAP) within a system. In a modern environment where competing service providers wish to share existing hardware through network virtualisation, the ability to prioritise some VAPs over others is a desirable feature. Providers paying a greater share of infrastructure costs may look for greater throughput, or to be able to prioritise between premium-level and advertising supported services. We validate the performance of this protocol through system level simulations that adhere closely to the 802.11 MAC specifications. We show through comprehensive evaluation that AlphaAP achieves arbitrary weighted throughput assignments in scenarios involving a variety of numbers of VAPs and associated STAs, while responding promptly to STAs arriving and leaving the access points.

Table of Contents

Chapter 1

Introduction

Usage of IEEE 802.11 within wireless networking is persistently increasing. A report conducted by Cisco in 2012 [2] demonstrated the growing number of smartphone users looking to access Wireless Local Area Networks (WLAN). More recently, Cisco's Visual Networking Index reported nearly 50% of mobile data traffic was offloaded onto WLANs in 2014 [1]. The report further highlighted the rapid increase in mobile data traffic, in part due to increasing demand for services such as video. A report by Ericsson [3] forecasts this continued rapid growth of mobile data traffic. Hotspot offloading has become an emerging trend as infrastructure providers seek to find new methods to cope with these increasing data demands.

Network infrastructure and network services are very closely linked, with the provision of new network services requiring new infrastructure to operate on. Virtualisation is a well-known networking technique that has been used previously in wired networks and encourages separation of infrastructure from services. Through this technique, infrastructure supplies network functionality through a software interface, which multiple services may then make use of. Hotspot offloading benefits from this technique by offering each service provider an individual virtual access point, while sharing the underlying cost of a single network device. Recent research has focused on Software Defined Wireless Networking (SDWN), Wireless Network Virtualisation (WNV), and NFV (Network Functions Virtualisation) [22]. Boingo, a leading Airport Wireless provider, has announced a rollout of NFV to five of their serviced Airports and are planning further future rollouts [18].

Despite this, recent research [17] indicates that virtualisation fails to provide fairness guarantees to groups of connected devices. Virtualisation enables sharing of infrastructure which introduces requirements of fairness to clients. IEEE 802.11 is a distributed technology with connected stations (STA) contending for access to a shared medium. In 802.11, research has been performed in ensuring fairness in the Downlink (DL) through Access Point (AP) controls [19]. However, ensuring fairness in Uplink (UL) is a more difficult challenge - associated STAs independently choose when to transmit, and any solution must be a distributed protocol operating at each STA. This challenge is central to this thesis. Additionally, it is beneficial for suggested solutions to adhere to the IEEE 802.11 standard. This ensures immediate compatibility with the many

real-world devices currently operating, and eases integration.

This thesis seeks to provide weighted throughput guarantees in the uplink to groups of connected STAs comprising a Virtual Access Point (VAP). With the varying shared costs of infrastructure, many service providers may wish to gain throughput allocations weighted according to their input, or offer a premium service with higher resources. To do so, this thesis introduces the notion of throughput guarantees for VAPs according to assigned weights. This work builds on the work of CVAP [9], an algorithm that uses Control-Theory to adapt the Contention Window used by associated STAs when contending for access. CVAP addresses the unfairness among VAPs by driving the network conditions towards equal throughput between VAPs. Whilst this benefits scenarios where providers may wish equal access to resources, it fails to handle the scenario where providers wish to tune the throughput provided to each VAP. We address this by providing AlphaAP, offering the ability to assign arbitrary throughput allocation in a scenario with multiple VAPs. This provides a configurable solution for sharing providers.

Chapter 2 presents an overview of the background technologies of the project and the related research. Chapter 3 details the core work of the project; outlining the system model constructed and the controller used to drive the system. The chapter concludes by discussing the custom simulator constructed and important implementation decisions. Chapter 4 contains a comparative evaluation of AlphaAP and examines the behaviour of the system in various conditions. Chapter 5 discusses points to consider when implementing AlphaAP on actual hardware, and Chapter 6 concludes the report and suggests areas of further research.

Chapter 2

Background Work

The Institute of Electrical and Electronics Engineers (IEEE) LAN/MAN Standards Committee created the first version of the 802.11 standards in 1997. This contained a set of specifications that allowed devices to operate within Wireless Local Area Networks (WLAN) at certain communication frequencies. Commonly referred to as *Wi-Fi*, this refers to the Wi-Fi Alliance, a body that certifies products as conforming to the IEEE 802.11 standards. Whilst other WLAN mechanisms existed, including HiperLAN and HomeRF, the IEEE 802.11 standards have become the de-facto accepted standard for WLAN communication today.

The IEEE 802.11 standards follow a continuous update timeline, with amendments agreed and published roughly every 5 years [12]. The original standard was published in 1997, with 802.11b and 802.11a amendments shortly thereafter published in 1999, offering higher data rates up to 54 Mb/s. The 802.11g amendment was published in 2003, offering higher data rates for devices operating within the 2.4 GHz spectrum.

These specifications offer details on both the Physical (PHY) layer and the Medium Access Control (MAC) layer. The physical layer deals with the transmission technologies necessary to transmit data within the radio spectrum. This thesis focuses on the MAC layer which documents methods of facilitating distributed access to the medium among competing devices. The 802.11 a/b/g specifications retain usage of the original MAC specifications. The MAC specifications provide two mechanisms for controlling medium access - the Distributed Coordination Function (DCF) and Point Coordination Function (PCF). The PCF offers an optional access mechanism not widely used and we shall not explore this further. Communicating over 802.11 using the DCF is a distributed protocol; STAs wishing to transmit must manage their own access to the medium in a way that attempts to prevent collisions from occurring. Collisions may occur at the PHY layer, through interference and noise on the channel corrupting frames, or at the MAC layer through other STAs transmitting at the same time.

The DCF uses a method of transmission known as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). This employs a slot-based access mechanism that we shall explore in more detail as it forms the foundation of this thesis.

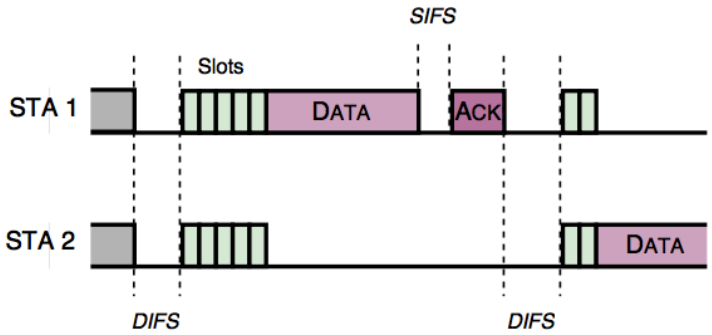

Figure 2.1: Contention Based Access example between two Stations

Fig. 2.1 shows an example slotted contention access that STAs employ. Access is divided into slots that occupy a slot duration as specified by the 802.11 PHY specifications. A STA, wishing to transmit a frame, must choose a slot value at random between 0 and a value known as the Contention Window (CW). Each slot is a fixed duration and thus the choice of a slot value indicates a minimum duration that must be counted down before transmission. STAs waiting to transmit need to sense the channel idle for a minimum period known as Distributed Inter-Frame Spacing (DIFS), then begin to decrease their chosen slot value as each idle slot passes, pausing the countdown whenever the channel is sensed busy again. Once their slot value is decreased to 0, a STA transmits. In Fig. 2.1, both STA 1 and 2 wish to transmit and, upon sensing the medium idle for a DIFS period, begin counting down to their slot. STA 1 reaches 0 after 5 slots, and begins transmitting data. This transmission is sensed by STA 2 and the countdown stops. After a Short Inter-Frame Spacing (SIFS) period, which is used for higher priority frames such as Acknowledgements (ACK), an ACK is sent by STA 1. Both STAs wait a DIFS period again then begin counting down to the next chosen slot. STA 2 reaches 0 after 2 slots and begins transmitting.

The assumption here is that STAs shall choose a slot value at random, different to slot values of other STAs for a suitably large CW value. As the number of STAs increases, the chance of two STAs picking the same slot increases for fixed CW. When this occurs the two STAs form a collision and the data frame transmitted by each may not arrive successfully. To mitigate this and deal with the scenario where larger numbers of STAs exist, a collision prompts a process known as binary exponential backoff. When this occurs, each colliding STA doubles their current CW up to a maximum value of CW_{max} and draws a new slot value between 0 and this new CW. Upon successful transmission a STA shall reset their CW to CW_{min}. Collisions are detected via the lack of an ACK being received from the receiving STA. This is a method known as positive acknowledgement, in that every correctly received data frame must be acknowledged and so the lack of an acknowledgement implies a collision has occurred.

Research has been performed relating to the earlier IEEE 802.11 standards and the DCF in attempting to differentiate service. Aad and Castelluccia [8] explored mechanisms for prioritising between services based on adjusting various parameters relating to DCF mentioned above, which in turn affect the throughput attainable by STAs employing these mechanisms. The authors considered adjusting the binary exponential backoff function, having lower priority STAs draw from a larger CW range. The

authors further considered varying the DIFS period used by STAs, introducing longer DIFS periods for STAs transmitting at a lower priority, a mechanism quite similar to the 802.11e Arbitration Inter-Frame Spacing (AIFS) that we discuss further in this section. Finally, the authors proposed introducing different maximum frame lengths for prioritising STAs, enforcing fragmentation of frames larger than this threshold. The authors encourage the usage of such parameters to achieve a means of differentiating service in the basic 802.11 specifications. Whilst this serves as a beneficial analysis, the authors do not propose a mechanism for assigning these priorities or updating them in a dynamic environment such as 802.11 where STAs may arrive and leave regularly.

A particular driver of mechanisms to achieve service differentiation and balance throughput was the introduction of the IEEE 802.11-2007 Standard. This included the 802.11e amendments which focused on Quality-of-Service (QoS). This introduced, alongside other mechanisms, the Enhanced Distributed Coordination Accesss (EDCA) mechanism that provides the ability to differentiate traffic based on an Access Category (AC) value. Each QoS capable STA contends for the ability to transmit, known as a Transmission Opportunity (TXOP). Each AC has a set of controllable parameters that dictate how STAs contend. These parameters are distributed in Beacon frames, which are well documented management frames from the original standard. These tunable parameters include the minimum contention window, CW_{min}, and the maximum contention window, CW_{max}, limiting the backoff behaviour of STAs. Furthermore, the parameters include a new Inter-Frame spacing, AIFS, which may extend the period at which STAs must detect the channel idle before resuming the countdown. Finally, a TXOP Limit parameter may be assigned that allows a STA, once it has reached the transmission slot, to continue transmitting frames up to the duration allocated without contending for access again.

The addition of these parameters and a mechanism for dynamically broadcasting them to associated STAs opened research opportunities into service differentiation, fairness, and throughput guarantees in 802.11. By allowing these parameters to be broadcast, proposed solutions that adhere to this broadcasting mechanism can achieve standards compliance, attaining seamless compatibility with QoS capable STAs in existence. Jeong, Choi, and Kim [15] indicated how the contention window limits and AIFS of EDCA could be used to provide weighted fairness between uplink and downlink in WLANs. This mitigates the imbalance present when only the AP contends for access in the downlink and all other STAs contend for access in the uplink. Similarly addressing the unfairness between uplink and downlink, Lee, Liao, and Chen [21] provide weighted fairness between uplink and downlink in WLANs by modifying only the minimum contention window size. The authors proposed a feedback control mechanism that adjusts the minimum contention window value based on monitoring the throughput and bandwidth requirements of the downlink and uplink flows. However, the authors require associated STAs to broadcast their bandwidth requirements to the AP (suggested alongside ACK frames), which would require modification of the behaviour of STAs, limiting the ease of implementing this mechanism.

Further research has attempted to address fairness concerns within 802.11 through other means. Yao et al. [23] investigate a modified packet scheduling algorithm for APs that balances fairness and efficiency. These solutions prove suitable for downlink

goals but fail when considering uplink, as STAs are distributed and must independently choose when to transmit. This is particularly evident in the work of Kyasanur and Vaidya [16] where the authors analysed the performance of 802.11 in the presence of selfish STAs choosing not to adhere to the standard MAC protocols. Thus, fairness goals and throughput guarantees must be provided through other means. Zou et al. [24] investigate achieving optimal throughput through setting a constant contention window based on the number of STAs associated to an AP. Whilst results indicate a constant contention window may attain an optimal throughput, this relies on all STAs knowing the number of contending STAs at a given moment, which would require broadcasting from the AP or modifications to each STA.

Research on throughput allocation in virtualised WLANs is sparse. SplitAP [10] provides a solution that offers uplink traffic fairness within virtualised WLANs by proposing a new architecture. This method requires the installation of a client-side plugin on STAs to augment this architecture, which moves away from 802.11 standards compliance to provide these guarantees and provides a barrier for rolling out this implementation.

CVAP [9] is a proposed algorithm that conforms to the 802.11 specification and provides equal uplink fairness to contending stations within virtual WLAN scenarios. This algorithm runs a controller for each virtual access point that adapts the contention window used by STAs associated to each virtual access point. These contention windows are broadcast in beacon frames as specified by the 802.11e standard. This guarantees equal throughput to VAPs and adapts to dynamic situations whilst conforming to standards, allowing existing devices to benefit from this configuration. Whilst CVAP guarantees equal throughput among VAPs, the algorithm offers no means to perform weighted allocation of throughput.

This thesis addresses an empty space in the research on virtualised WLANs. We construct a new algorithm, AlphaAP, using the techniques from CVAP, that allows for arbitrary allocation of throughput to each VAP within a group. This is configured through weights for VAPs which may be assigned on monetary, traffic quality, or similar quality of service differentiation requirements.

Chapter 3

AlphaAP

Central to this thesis is the introduction and usage of a system model that we leverage to generate weighted throughput for the Virtual Access Points (VAP) within the system. We adapt the system model provided for CVAP [9] and use the same structure to validate the model and construct a suitable controller that drives the system to a desired point of operation.

As in CVAP, we make use of a Proportional-Integral (PI) controller [20] to drive the system, which remains particularly applicable to AlphaAP. PI controllers operate by continuously adjusting a control parameter affecting the system in response to an error signal and past error history. By deriving an appropriate error signal and controller parameter, PI controllers may operate without particular knowldge of the underlying process. We can tune the responsiveness and stability of the PI controller through suitable choice of proportional and integral constants. Of particular benefit, PI controllers are simplistic with minimal implementation footprint - an important criteria for AlphaAP as this protocol would be implemented on the firmware of a network device.

We introduce constant weights, denoted $\alpha_1, \alpha_2, \ldots, \alpha_N$, ranging between 0 and 1, for each VAP in the system. These weights allocate a percentage of the total throughput to each VAP. For example, a value of $\alpha_1 = 0.1$ would indicate in our system that VAP_1 should receive 10% of the total throughput available.

We make various assumptions with our system model to simplify the analysis. Firstly, we assume perfect channel conditions with no losses due to noise. We make the simplifying assumption that all stations (STA) are within range of all others, removing the Hidden Terminal problem where some STAs may not be able to receive the transmissions of others. The combination of these mean that in our model collisions only occur due to the choice of two or more STAs transmitting at the same time.

In 802.11, Quality of Service amendments allow for the addition of multiple transmission queues - one for each traffic priority. We make the simplifying abstraction that each STA operates only a single transmission queue. We also assume this queue is never empty - that is, a STA always has a packet ready to transmit. Finally, in our configuration we use the Contention Window to tune the behaviour of a VAP and

as such, we fix the contention window by setting $CW_{min} = CW_{max}$, thus eliminating the binary exponential backoff described in Chapter 2. The weights, α_i, that allocate throughput, are constants associated with each VAP. These are considered fixed and do not change over the duration of a scenario.

After introducing the system model and deriving the addition of weights to the system we use control theory to introduce a mechanism for driving the contention window to achieve the optimal point of operation with a suitably small delay. We show through analysis that our configuration for the controller remains stable, and validate this later using simulations.

3.1 System Model

In this section we define our system model and clearly introduce weights to bias the throughput allocation of AlphaAP. Our system consists of N Virtual Access Points, VAP_1, VAP_2, ..., VAP_N, each with $n_1, n_2, \ldots, n_N$ associated stations. Fig. 3.1 presents a scenario with 3 VAPs and varying numbers of associated STAs.

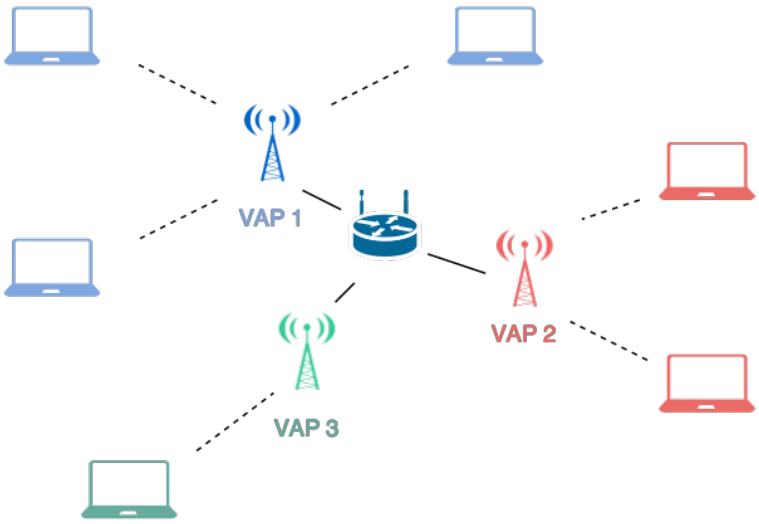

Figure 3.1: An example VAP scenario with 3 VAPs and varying associated STAs

We work with the constant weights, $\alpha_1, \alpha_2, \ldots, \alpha_N$, that range between 0 and 1, as introduced above.

In the slotted access mechanism as specified by 802.11 standards, we consider a given slot as empty or occupied. A slot is empty if no STA has attempted a transmission within this slot, and occupied if a transmission is attempted. We note an occupied slot may contain either a successful transmission or a colliding transmission.

We denote the probability of a station in VAP_i transmitting on some randomly chosen slot, as derived by Bianchi [11],

$$\tau_i = \frac{2}{1 + CW_i}, \tag{3.1}$$

where CW_i is the Contention Window value broadcast to stations associated with VAP_i. This equation takes a simple form due to the fixing of $CW_{min} = CW_{max}$.

Next, we introduce an expression defining the total throughput achieved by VAP_i,

$$R_i = \frac{\mathbb{E}[\text{data from VAP}_i \text{ in slot}]}{\mathbb{E}[\text{slot length}]} = \frac{S_i L}{P_e T_e + (1 - P_e) T_o}. \tag{3.2}$$

In Eq. 3.2, L denotes the average frame size of a transmission. T_e denotes the duration of an empty slot which is the slot duration as specified by the 802.11 PHY layer specifications. T_o denotes the average duration of an occupied slot. An occupied slot may contain either a successful or colliding transmission. We make the simplifying assumption of fixed frame sizes in this analysis. We consider the average duration of an occupied slot, T_o, to be a suitable approximation given that the duration of a collision and a successful transmission are almost equal under these assumptions in 802.11.

Next, we introduce P_e as the probability of an empty slot in the system. That is, the probability no STA amongst all VAPs decides to transmit in the slot, defined as,

$$P_e = \prod_{k=1}^{N} (1 - \tau_k)^{n_k}. \tag{3.3}$$

Next, we define the probability of a slot containing a successful transmission from VAP_i, which is the probability a single STA from VAP_i decides to transmit, and all remaining STAs in VAP_i and other VAPs decide not to transmit. This can be expressed as,

$$S_i = n_i \tau_i (1 - \tau_i)^{n_i - 1} \prod_{\substack{k=1 \\ k \neq i}}^{N} (1 - \tau_k)^{n_k} = \frac{n_i \tau_i}{1 - \tau_i} \prod_{k=1}^{N} (1 - \tau_k)^{n_k} = \frac{n_i \tau_i}{1 - \tau_i} P_e. \tag{3.4}$$

The weights introduced above, α_i, represent the percentage of total throughput in the system assigned to VAP_i. To this effect, we require full assignment of the throughput in the system, denoted,

$$\sum_{k=1}^{N} \alpha_k = 1. \tag{3.5}$$

Having introduced the parameters, we set out two goal conditions.

1. The achieved throughput for VAP_i equals α_i fraction of the total throughput

$$R_i = \alpha_i \sum_{k=1}^{N} R_k$$

2. The total throughput in the system is the maximum possible.

$$\max \sum_{k=1}^{N} R_k$$

Addressing the first, we see that the following holds,

$$\frac{R_i}{\alpha_i} = \frac{R_j}{\alpha_j}$$
$$\frac{\frac{n_i\tau_i}{1-\tau_i}P_e}{\alpha_i(P_eT_e+(1-P_e)T_o)} = \frac{\frac{n_j\tau_j}{1-\tau_j}P_e}{\alpha_j(P_eT_e+(1-P_e)T_o)} \tag{3.6}$$

We next note that we can use $\tau_i \ll 1 \; \forall i$. This follows as τ_i is the probability of a single STA choosing to transmit in some slot. If this were a large value the collision probability amongst all VAPs would be overly large.

Given Eq. 3.6, and using that $\tau_i \ll 1 \; \forall i$, we have that the following is a suitable condition to ensure the first goal.

$$\frac{n_i\tau_i}{\alpha_i} = \frac{n_j\tau_j}{\alpha_j} \tag{3.7}$$

Achieving the second goal involves maximizing the total throughput in the system. We note that goal one ensures that the throughput of an individual VAP$_i$ is a constant fraction, α_i, of the total throughput and so maximizing R_i is equivalent to maximizing the total throughput.

To this end, through some operations, we can express R_i as,

$$\begin{aligned} R_i &= \frac{P_e n_i \tau_i L}{P_eT_e+(1-P_e)T_o} = \frac{n_i\tau_i L}{\frac{T_o}{P_e}+(T_e-T_o)} \\ &= \frac{n_i\tau_i L}{T_o e^{\frac{n_i\tau_i}{\alpha_i}\sum\alpha_k}-(T_o-T_e)} = \frac{n_i\tau_i L}{T_o e^{\frac{n_i\tau_i}{\alpha_i}}-(T_o-T_e)}. \end{aligned} \tag{3.8}$$

Following this, we can compute the optimal τ_i that achieves this throughput, which we shall denote as τ_i^*. We do so by setting,

$$\frac{dR_i}{d\tau_i} = 0 \tag{3.9}$$

Through some operations, we arrive at,

$$T_o e^{\frac{n_i\tau_i}{\alpha_i}}\left(1-\frac{n_i\tau_i}{\alpha_i}\right) - T_o + T_e = 0 \tag{3.10}$$

We use a Taylor-series expansion and show,

$$e^{\frac{n_i\tau_i}{\alpha_i}} \approx 1 + \frac{n_i\tau_i}{\alpha_i} + \frac{n_i^2\tau_i^2}{2\alpha_i^2} + \ldots, \tag{3.11}$$

and ignore the terms following second power as $\tau_i \ll 1$. Thus, substituting in Eq. 3.10, we reach that,

$$\tau_i^* = \frac{\alpha_i}{n_i}\sqrt{\frac{2T_e}{T_o}}, \tag{3.12}$$

and we can clearly show that Eq. 3.7 holds for this optimal τ_i^*, i.e.

$$\frac{n_i\tau_i^*}{\alpha_i} = \frac{n_j\tau_j^*}{\alpha_j}$$
$$\frac{n_i}{\alpha_i}\frac{\alpha_i}{n_i}\sqrt{\frac{2T_e}{T_o}} = \frac{n_j}{\alpha_j}\frac{\alpha_j}{n_j}\sqrt{\frac{2T_e}{T_o}}.$$

The corresponding contention window is then defined as,

$$CW_i = \frac{2}{\tau_i^*} - 1. \tag{3.13}$$

We similarly compute the *point of operation* of the WLAN, which is the probability of an empty slot when all VAPs are configured with the above optimal τ_i^*, as,

$$P_e^* = \prod_{k=1}^{N}(1-\tau_i^*)^{n_k}$$
$$P_e^* = \prod_{k=1}^{N} e^{-\alpha_k\sqrt{\frac{2T_e}{T_o}}}$$
$$P_e^* = e^{-\sqrt{\frac{2T_e}{T_o}}\sum_{k=1}^{N}\alpha_k}$$
$$P_e^* = e^{-\sqrt{\frac{2T_e}{T_o}}}, \tag{3.14}$$

which achieves the result. P_e^* is a fixed constant irrespective of the number of VAPs or STAs, which represents a target state of operation for the WLAN. This provides a desirable target that we may use in the next section. This concludes our system model and the introduction of weights.

3.2 Controller

We begin by defining the error signal for the controller. We desire two objectives - the first is to maximize the overall throughput in the system, and the second is for each VAP_i to achieve a share of throughput according to the weight, α_i.

To achieve the first, we use the error signal, e_{opt},

$$e_{opt} = P_e^* - P_e. \tag{3.15}$$

Using this, if the system achieves a probability of empty slot, P_e, that is out-with the optimal point of operation, this will trigger an adjustment of the CW_i in the appropriate direction.

To achieve the second objective, we introduce the following error signal,

$$e_{fair,i} = \frac{S_i}{\alpha_i} - \sum_{k=1}^{N} S_k. \tag{3.16}$$

This signal ensures that if VAP_i is achieving a larger throughput than α_i fraction of the sum of all throughputs, the error signal will be positive resulting in an increase of CW_i thereby lessening the channel utilization for VAP_i.

We can see in the goal condition of Eq. 3.7, this error signal shall reach 0, via,

$$\sum_{k=1}^{N} S_k \approx P_e \sum_{k=1}^{N} n_k \tau_k = n_i \tau_i P_e \sum_{k=1}^{N} \frac{\alpha_k}{\alpha_i} = \frac{n_i \tau_i}{\alpha_i} P_e \approx \frac{S_i}{\alpha_i},$$

where above we use that $n_k \tau_k = \frac{\alpha_k}{\alpha_i} n_i \tau_i$ and $\tau_i \ll 1 \; \forall i$.

This results in the following complete error signal for VAP_i:

$$e_i = e_{opt} + e_{fair,i} = P_e^* - P_e + \frac{S_i}{\alpha_i} - \sum_{k=1}^{N} S_k. \tag{3.17}$$

We can show that there exists a unique solution where $e_i = 0 \; \forall i$ with $e_{opt} = 0$ and $e_{fair,i} = 0 \; \forall i$. [1]

Using these results, we outline the new control system with these additional weights,

$$E = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{pmatrix} = \begin{pmatrix} P_e^* - P_e + S_1 - \alpha_1 \sum_j S_j \\ P_e^* - P_e + S_2 - \alpha_2 \sum_j S_j \\ \vdots \\ P_e^* - P_e + S_N - \alpha_N \sum_j S_j \end{pmatrix} \tag{3.18}$$

$$O = \begin{pmatrix} o_1 \\ o_2 \\ \vdots \\ o_N \end{pmatrix}, \; W = \begin{pmatrix} CW_1 \\ CW_2 \\ \vdots \\ CW_N \end{pmatrix} \tag{3.19}$$

Fig. 3.2 presents the system containing a distinct PI controller for each VAP_i that will take as input the error signal e_i and output o_i. Each VAP_i will take the output o_i and multiply it by $\frac{n_i}{\alpha_i}$ to obtain the CW value to be broadcast to the associated stations.

[1] See Appendix A.

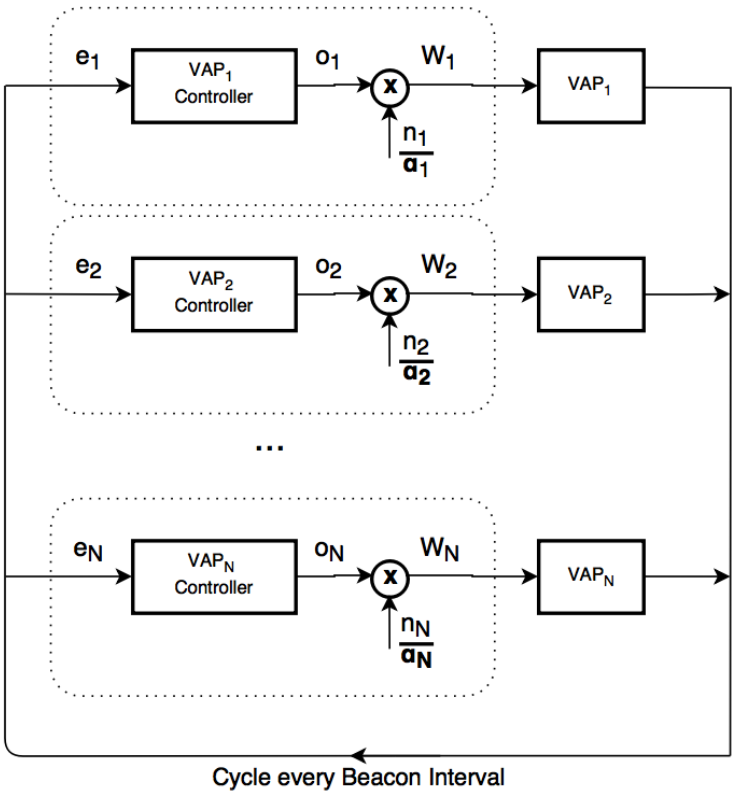

Figure 3.2: Our system containing a distinct PI controller for each VAP

Given this, we express the relationship:

$$W(z) = \mathfrak{N}_\alpha \cdot O = \mathfrak{N}_\alpha \cdot C \cdot E(z) \tag{3.20}$$

where we define

$$\mathfrak{N}_\alpha = \begin{pmatrix} \frac{n_1}{\alpha_1} & 0 & \dots & 0 \\ 0 & \frac{n_2}{\alpha_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{n_N}{\alpha_N} \end{pmatrix}, \; C = \begin{pmatrix} C_{PI}(z) & 0 & \dots & 0 \\ 0 & C_{PI}(z) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C_{PI}(z) \end{pmatrix} \tag{3.21}$$

Thus, our system can now be collapsed into a single relationship involving a single controller and the external collection of VAPs, as indicated in Fig. 3.3. The system takes as input the collection of errors, E, produces the output, O, which, in combination with N_α, produces W, the collection of Contention Windows to be used by the VAPs. Z_{-1} represents the direct relation between the output signal of the PI controller and the input error signal of the next iteration.

To analyse this system further, we follow the same process as in CVAP [9], by defining the transfer function, H, that takes as input the PI controller output signal, o_i, and

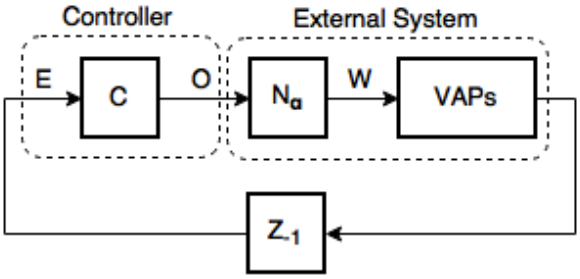

Figure 3.3: Our system represented as a single controller and external system, with a transfer function representing the loop.

produces the next error signal, e_i. Taking the output signals, O, we can obtain the contention windows to be used by each VAP by multiplying by N_α. Given the contention window used by a VAP over a beacon interval, we can obtain τ_i through Eq. 3.1, P_e through Eq. 3.3, and S_i through Eq. 3.4, producing all necessary parameters to compute the new error signal, e_i. Thus, we observe a non-linear relationship between E and O.

In order to use this relationship to define the transfer function H, we must linearize this around the stable point of operation presented in Section 3.2, P_e^*, when the system expresses only small variation. Thus, we can express this variation using the output signal, o_i, as follows,

$$o_i = o_{i,opt} + \delta o_i, \tag{3.22}$$

with $o_{i,opt}$ the output signal that produces the ideal τ_i^* yielding the point of operation.

Having defined this variation we can approximate this by,

$$\delta E = H \cdot \delta O, \tag{3.23}$$

where we define H using the partial derivatives,

$$H = \begin{pmatrix} \frac{\partial e_1}{\partial o_1} & \frac{\partial e_1}{\partial o_2} & \cdots & \frac{\partial e_1}{\partial o_N} \\ \frac{\partial e_2}{\partial o_1} & \frac{\partial e_2}{\partial o_2} & \cdots & \frac{\partial e_2}{\partial o_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_N}{\partial o_1} & \frac{\partial e_N}{\partial o_2} & \cdots & \frac{\partial e_N}{\partial o_N} \end{pmatrix} \tag{3.24}$$

Using that,

$$\frac{\partial e_i}{\partial o_j} = \frac{\partial e_i}{\partial \tau_j} \frac{\partial \tau_j}{\partial CW_j} \frac{\partial CW_j}{\partial o_j}, \tag{3.25}$$

we have, due to the addition of weights in the system,

$$\frac{\partial CW_j}{\partial o_j} = \frac{n_j}{\alpha_j}, \tag{3.26}$$

and,

$$\frac{\partial \tau_j}{\partial CW_j} = -\frac{1}{2}\tau_j^2. \tag{3.27}$$

We next consider $\frac{\partial e_i}{\partial \tau_j}$. We separate cases and consider the case where $i \neq j$ first. With some work, we obtain

$$\frac{\partial e_i}{\partial \tau_j} = \frac{P_e n_j}{1-\tau_j}\left(1 - \frac{1}{\alpha_i}\left(\frac{n_i\tau_i}{1-\tau_i}\right) - \frac{1}{1-\tau_j} + \sum_k \frac{n_k\tau_k}{1-\tau_k}\right). \tag{3.28}$$

Evaluating this at the stable point of operation, with τ_i^* as in Eq. 3.12, and using $n_k\tau_k = \frac{n_i\tau_i}{\alpha_i}\alpha_k$,

$$\frac{\partial e_i}{\partial \tau_j} = 0. \tag{3.29}$$

In a similar manner, we show that $\frac{\partial e_i}{\partial \tau_i}$ can be shown as

$$\frac{\partial e_i}{\partial \tau_i} = \frac{P_e n_i}{1-\tau_i}\left(1 + \frac{1}{\alpha_i}\left(\frac{1}{1-\tau_i} - \frac{n_i\tau_i}{1-\tau_i}\right) - \frac{1}{1-\tau_i} + \sum_k \frac{n_k\tau_k}{1-\tau_k}\right), \tag{3.30}$$

which, when evaluated at the stable point of operation, results in,

$$\frac{\partial e_i}{\partial \tau_i} = \frac{P_e n_i}{\alpha_i}. \tag{3.31}$$

With some rearranging, we have that,

$$\frac{\partial e_i}{\partial o_i} = \frac{P_e n_i}{\alpha_i} \cdot -\frac{1}{2}\tau_i^2 \cdot \frac{n_i}{\alpha_i} = -\frac{P_e T_e}{T_o}, \tag{3.32}$$

and

$$\frac{\partial e_i}{\partial o_j} = 0, \tag{3.33}$$

which results in the transfer function matrix, H, being defined as,

$$H = K_H I \tag{3.34}$$

$$K_H = -\frac{P_e T_e}{T_o}. \tag{3.35}$$

We finish by choosing suitable constants K_P, K_I for the controller. These must be chosen such that they offer suitable responsiveness to changes, yet allow the controller to settle at the target value without extreme fluctuations. Due to the direct parallel

between AlphaAP and CVAP , we adopt the same constants [9] and evaluate the suitability of these constants in the Evaluation phase. As such, we have constants,

$$K_P = 0.4\frac{T_o}{P_e^* T_e},$$
$$K_I = \frac{0.2}{0.85}\frac{T_o}{P_e^* T_e}.$$

This characterizes our system based on H and C and completes our system model and controller description.

3.3 Simulation

A system-level simulator was constructed as part of the evaluation phase of the project. The simulator focuses on the MAC layer components of the IEEE 802.11 specification and makes many simplifying assumptions to limit the work to a feasible scope. The simulator was constructed using the C programming language and progresses time using discrete events registered by the Medium, VAP, and STA entities in the system. A fundamental goal in construction was to allow for flexibility in running simulations; offering easy operation through configuration files that are used as input. This allows for the organisation of scenarios into separate configuration files which facilitate repeated simulations. This greatly assisted the evaluation phase of this project.

In the following section we discuss the simulator in greater detail. Section 3.3.1 discusses various implementation choices in the simulator, Section 3.3.2 discusses the assumptions made and limitations of the simulator, and Section 3.3.3 discusses the rationale behind building a simulator rather than using an existing framework.

3.3.1 Implementation

3.3.1.1 Transmission State Tracking

Given the shared nature of the wireless medium, stations wishing to transmit must undergo the set of operations as specified by a coordination function [7]. The state diagram in Fig. 3.4 indicates the behaviour as implemented by STAs in our simulation, regardless of the MAC protocol employed. We note this implicitly confirms it is not necessary for modifications to be made to STAs to benefit from AlphaAP.

STAs are considered `Inactive` if they have not associated with a Virtual Access Point. This may occur when simulation scenarios include STAs associating or disassociating at a specific time within the simulation, which is used to test the adaptation of a protocol to changing numbers of STAs. When `Inactive`, a STA performs no actions until an association occurs.

In the default state, STAs are either in the `Empty` or `Ready` state, indicating the presence (or lack of) a packet ready to be transmitted. When in the `Ready` state, STAs

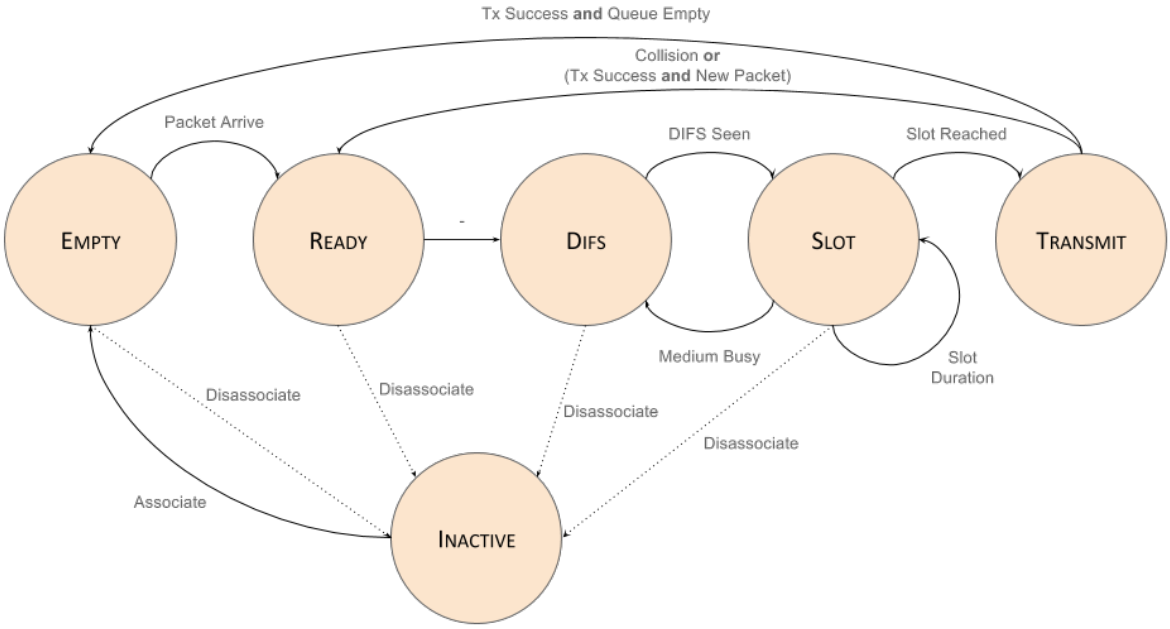

Figure 3.4: Station State Diagram

immediately move to the `DIFS` state. Following this, transitions occur based on events representing those conditions defined in the IEEE Standard [7]; with events occurring for DIFS intervals, slot boundaries, and slot backoff reached. If the `Medium` becomes busy during slot operations, the STA resets the state to `DIFS` and repeats the process while maintaining the random slot generated previously.

Once the slot event is activated that corresponds to the chosen slot, a STA transmits the packet in the simulation by passing the packet to the `Medium` entity. Section 3.3.1.2 discusses further the behaviour within the `Medium`. STAs receive a callback when the packet has finished transmitting in the `Medium`, and the packet has the `collided` flag set to `true` or `false` to indicate if a collision occurred. The STA now enters the `Empty` state if a successful transmission occurred and no further packets exist in the queue. Alternatively, the STA enters the `Ready` state if a successful transmission occurred but further packets are available to transmit, or, if a collision occurred which needs to retransmit.

3.3.1.2 Medium Access and Sensing

To simulate the notion of a shared medium, all STAs wishing to transmit submit packets to a single `Medium` entity. The `Medium` is queried by STAs to determine if the channel is busy or idle, simulating the physical carrier sensing that occurs in 802.11. The simulated `Medium` employs the notion of a Network Allocation Vector (NAV) and abstracts the physical carrier sensing.

Fig. 3.5 shows the behaviour of the `Medium` when two packets overlap in transmissions.

Figure 3.5: Shared Medium - Behaviour on Collision

The first packet is initially accepted by the `Medium`. Upon insertion of the second packet, the `Medium` arbitrates these and indicates a collision by setting the `collided` flag on the packets. When the time is reached to emit these packets, the sender is notified of a completed transmission and the collision is indicated by the presence of the flag.

This process abstracts the STAs awaiting an `ACK` response from the receiver of the packet and simplifies the behaviour of the simulator, allowing focus on the `MAC` protocol in use by VAPs.

3.3.1.3 Simulator Event Loop

The simulator design choice employs the use of discrete events to progress the simulation. Rather than increment in short steps of time, the simulator requests the next event to occur from each entity (`Medium`, `VAP`, `STA`) in the simulation and chooses the earliest to activate. In this manner, the simulator may skip large durations when no further events will occur in the intermediate time period. Fig. 3.6 presents the process that occurs on each simulation loop; this continues until the simulation time has reached the end time specified in the configuration file.

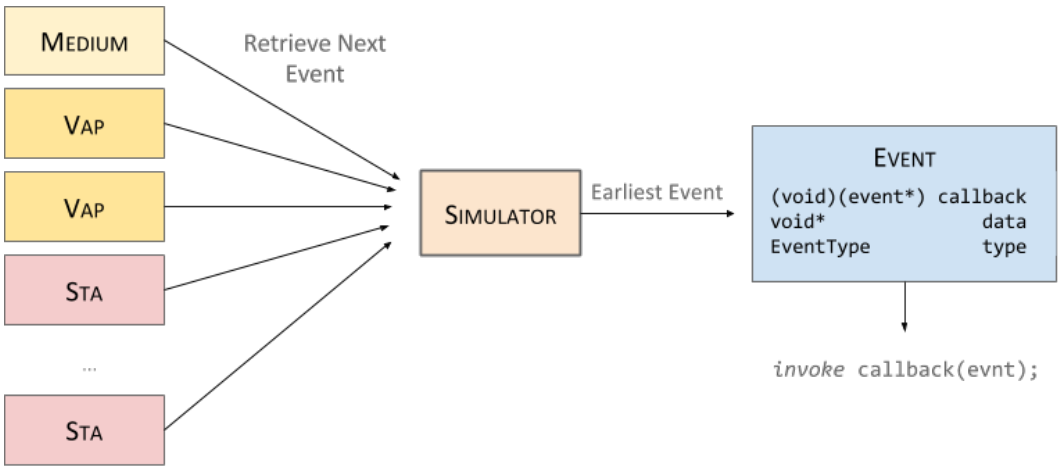

Figure 3.6: Simulator Progression - Choice of Events

As shown in Fig. 3.6, each event is provided from an entity and contains a callback function and a data instance. This allows for each event to be activated immediately on selection, and unique callback functions may be assigned for each event type.

3.3.1.4 Configuration Files

A fundamental goal in the construction of the simulator was to facilitate the evaluation phase of this research. To achieve this, the simulator accepts information on the scenario to simulate via configuration files which are easily modified.

Users can adjust through configuration files the number of VAPs present in the scenario, and the weights assigned to each of these. The MAC protocol is selected ranging from DCF, CVAP, and AlphaAP. Varying numbers of STAs may be assigned to each VAP and each STA may operate in either *saturated* or *unsaturated* traffic modes. Association and Disassociation Events may be specified; indicating a STA should associate or disassociate from the supplied VAP at a given time. Finally, the total duration of the simulation may be specified.

The configuration file also allows for separation of output based on the type of messages written by the simulator. These are separated into four categories: Statistics, Events, Errors, and Values. The user may configure these to be output to separate files, standard output, or not at all; allowing for flexibility in which messages are output. Finally, the user may customise exactly which value information is output, such as contention windows, slot choices, and probabilities. This allows each scenario to output only the information required for further analysis and increases the efficiency of the simulation.

The combination of these configuration options allows for many different scenarios to be run through the simulator without recompilation. The presence of redirecting output in configuration files, and selecting only values of interest, allows for many configuration files to be constructed tailored to evaluation requirements.

3.3.2 Limitations

Various limitations have been imposed and assumptions made when constructing this simulator. These abstract unnecessary details and make practical the workload involved. Here, we briefly describe some of these restrictions placed on the simulator.

3.3.2.1 Fixed Packet Size

Each STA simulated has packets arrive into a transmission queue at intervals which vary based on the delivery mode chosen. While the configuration file allows varying the delivery mode (between `uniform` and `poisson`), the packet size transmitted remains fixed at 1500 bytes, as specified in a constant in `sta.c`.

3.3.2.2 Fixed Data Rate

In addition to the packet size, the transmission data rates are fixed for each STA simulated. The data rate of each STA has been fixed at 54 Mbps and `ACK` rates are fixed at 24 Mbps. Whilst these are fixed, these rates are represented by constants in `sta.c` and can be easily altered during recompilation.

3.3.2.3 Hidden Node Problem

The simulator does not simulate the physical positioning of STAs. For simplicity, we assume each STA is within range of all other STAs thus removing the Hidden Node problem. As a consequence, we have chosen not to implement the well-known CTS / RTS mechanism of 802.11.

3.3.3 Justification

Many other system-level simulators exist for IEEE 802.11, including NS3 [4], OMNeT++ [5], or Pamvotis [6]. The decision was made to implement a custom simulator rather than choose to modify one of these for the reasons discussed below.

CVAP and AlphaAP are unusual algorithms. They run a PI controller once every beacon interval which modifies a contention window parameter and broadcasts this to STAs. This broadcasting conforms to the IEEE 802.11 specifications by altering the EDCA parameter set present in beacon frames. Whilst this mechanism could be implemented in the frameworks mentioned above, this would require modifying the internal structure of these frameworks; a process that is error-prone. Performing this modification successfully would require allocating time to learning the internal structure of these frameworks. The decision was instead made to spend this time on building a custom simulator, focusing on important areas and abstracting unnecessary details.

Many of the limitations mentioned in Section 3.3.2 are purposeful; they limit the research undertaken to a reasonable scope. In the simulators above, enforcing these limitations would require extra time to disable the necessary features or document their effect on our system. In a custom-built simulator these limitations are easy to abstract and make adhere to our system model.

Chapter 4

Evaluation

Simulations are performed to evaluate the proposed system. Through certain system scenarios we explore the responsiveness, correctness, and suitability of AlphaAP.

We begin by examining the total throughput achieved across three scenarios. Each STA is *saturated*, that is, always having a packet to transmit, with packets of length 1500 bytes. Each STA operates using a fixed PHY data rate of 54 Mbps. We consider overhead for `MAC`, `LLC`, `IP`, and `UDP` headers. Fig. 4.1 shows, for each scenario, this total throughput achieved with three selected MAC protocols: `DCF`, `CVAP`, and `α-AP` (AlphaAP). Throughput is given per VAP. Each scenario is repeated across five simulations and the mean of these attempts is displayed. The error bars are not displayed as they are too small to be visible on the graph.

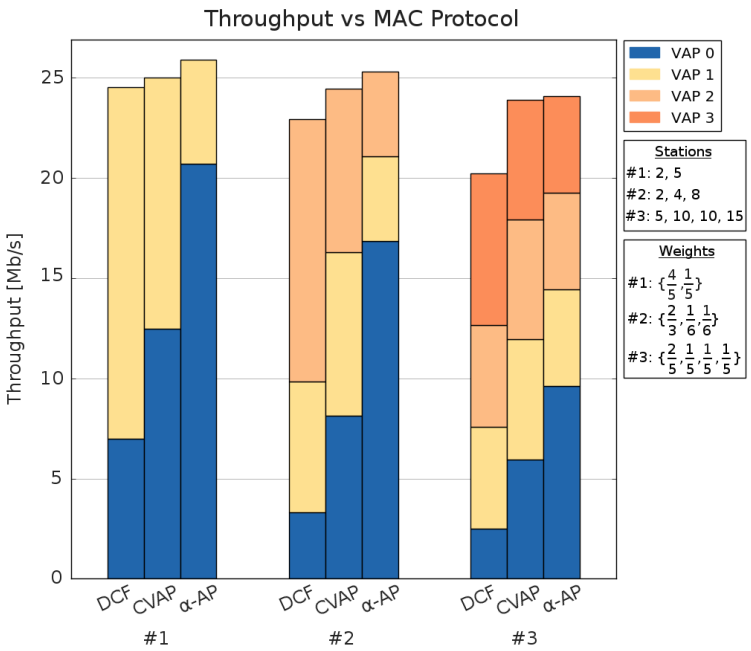

Figure 4.1: Throughput attained in varying VAP scenarios with MAC protocols `DCF`, `CVAP`, and `α-AP`. Weights, α, listed beside the plot and number of associated STAs.

We can see from these results that AlphaAP is able to achieve a total throughput greater than DCF in all scenarios. In scenario three, AlphaAP achieves a total throughput of 24.07 Mbps, with CVAP and DCF achieving 23.88 Mbps and 20.20 Mbps respectively. The difference between the total throughput attained by DCF and AlphaAP increases as we look at scenarios with larger numbers of STAs. In all scenarios, AlphaAP attains a Jain Fairness Index [14] value of 1 when considered using our weighted throughput measure detailed further below in Eq. 4.1, whilst CVAP and DCF achieve lower values.

We note that the scenarios are set with VAP 0 assigned the largest weight in each scenario, yet having the lowest numbers of associated STAs. The lower number of STAs is reflected by the decreasing throughput VAP 0 attains in each scenario under DCF. DCF provides an unfair allocation of throughput to VAPs 1, 2, and 3 in all scenarios, given the larger numbers of STAs associated. However, AlphaAP is able to correctly assign VAP 0 the allocated share of total throughput according to the weight specified in each scenario. We also observe that in Scenarios 2 and 3, AlphaAP has weights set to equalise the throughput allocated to VAPs 1, 2, and 3 while VAP 0 obtains a larger share. AlphaAP is able to correctly achieve this throughput share in both scenarios, indicating AlphaAP achieves the goal of assigning weighted throughput shares amongst competing numbers of VAPs and STAs.

Next, we briefly examine the individual throughput attained by each STA in Scenario 1; the scenario with 2 VAPs, one with 2 STAs and the other with 5 STAs. These are the same results as in Scenario 1 of Fig. 4.1 with STA-level granularity. Fig. 4.2 shows for this scenario the throughput attained per STA, in varying MAC protocol modes.

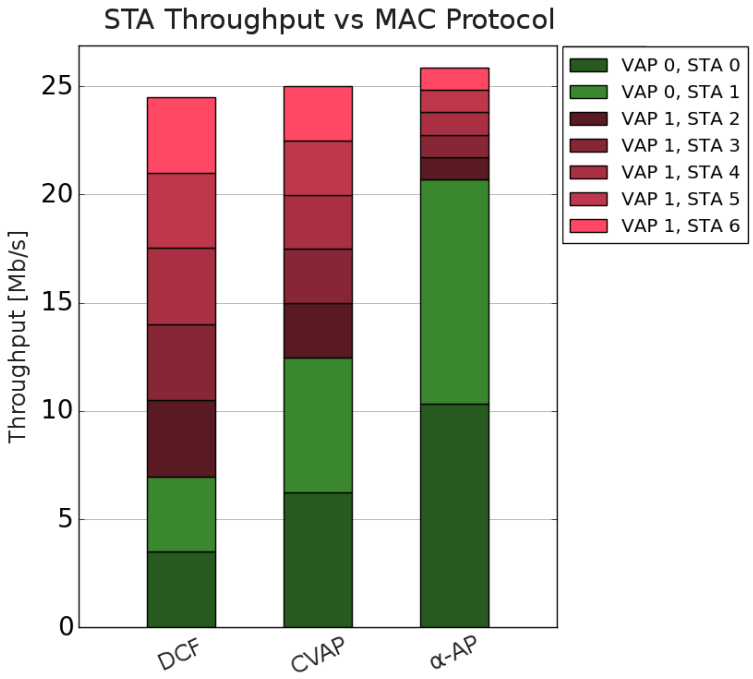

Figure 4.2: Throughput attained by each STA, grouped by VAP, in MAC modes `DCF`, `CVAP`, and `α-AP`. VAP 0 has 2 STAs associated, and VAP 1 has 5 STAs. The weights are $\alpha_0 : \frac{4}{5}$, $\alpha_1 : \frac{1}{5}$.

We first consider the individual STA throughput attained in each mode. In DCF

operation, the mean throughput obtained by each STA is 3.5 Mbps, with a standard deviation of 0.02. As expected, each STA in DCF obtains a fair share of the available throughput. In CVAP operation, VAP 0 obtains a throughput of 12.47 Mbps, with standard deviation 0.01, and VAP 1 obtains 12.52 Mbps, with standard deviation 0.01. CVAP equally shares the total throughput available, even with VAP 0 only containing 2 STAs while VAP 1 contains 5. Throughput per VAP is equally shared among the associated STAs with VAP 0 and VAP 1 STAs obtaining mean throughput of 6.24 Mbps and 2.5 Mbps respectively.

Focusing on AlphaAP, we first note the total throughput attained by VAP 0 and VAP 1 is 20.69 Mbps and 5.19 Mbps respectively, with standard deviation less than 0.04 in both cases. This corresponds to the weight assignments of $\frac{4}{5}$ to VAP 0 and $\frac{1}{5}$ to VAP 1. We note that the STAs associated with VAP 0 obtain an equal share of the throughput achieving a mean individual throughput of 10.35 Mbps, and VAP 1 STAs attaining a throughput of 1.04 Mbps. This confirms individual STAs continue to obtain an equal share of the throughput assigned to a VAP in AlphaAP.

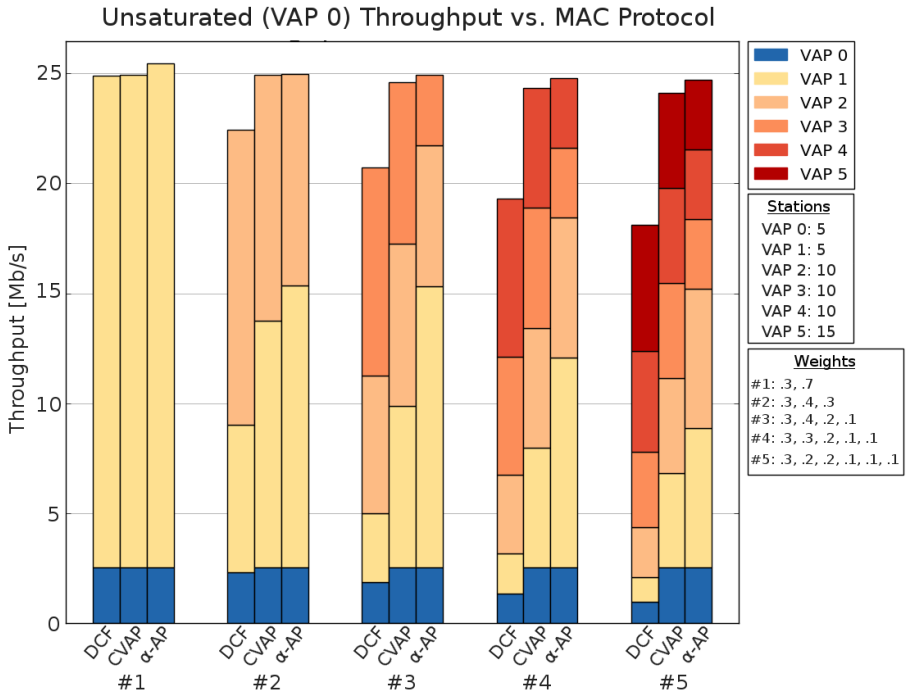

Figure 4.3: Throughput Achieved, per-VAP, in modes `DCF`, `CVAP`, `α-AP` with VAP 0 containing unsaturated STAs for increasing numbers of VAPs. Weights and STA numbers are listed alongside the plot.

Fig. 4.1 displays the throughput achieved when all STAs are saturated. Next, we examine the throughput achieved when a single VAP, VAP 0, contains unsaturated STAs, with packets arriving following a poisson distribution at a rate of 500 Kb/s. VAPs are configured to have a different number of STAs associated, as indicated beside the plot. The weights assigned for AlphaAP operation are also listed. As above, each simulation was performed five times and the mean of these results was taken. The error range for these results is too small to be visible on the plot and so error bars are not

shown. Fig. 4.3 presents these results.

Focusing first on DCF, we note that the total throughput steadily decreases as we increase the number of VAPs. As the number of contending STAs increase, more collisions occur and each colliding STA performs binary exponential backoff, resulting in a greater waiting time to access the medium and thus decreased total throughput. Focusing on VAP 0, the unsaturated VAP, we note the obtained throughput decreases steadily as more STAs are present. This is clearly an undesirable result in a shared situation; unsaturated STAs should have their transmissions accepted and not be prevented by the presence of other saturated STAs. Modes CVAP and AlphaAP are able to continually accept all transmissions from the unsaturated STAs of VAP 0, as indicated by the constant throughput obtained by the VAP, whilst meeting the goal criteria of each protocol. CVAP is able to achieve equal throughput among the remaining VAPs, and AlphaAP achieves the weight distributions specified for each remaining VAP. We note this holds for the varying weight distributions of each scenario and number of STAs, indicating AlphaAP operates successfully in the presence of unsaturated STAs. We also note AlphaAP is able to continually achieve an optimal total throughput, even in the presence of unsaturated STAs.

IEEE 802.11 operates by adapting the contention window (CW) of associated stations in response to transmission conditions. Next, we explore the CW changes over time in a system with 2 VAPs, with 2 and 5 associated stations respectively. It is important the system adapts the CW rapidly and that this CW remains stable once it has reached the target value. Fig. 4.4 presents the CW changes over time in a scenario with CVAP and AlphaAP, where the AlphaAP mode attempts to satisfy weights of $\alpha_0 = \frac{4}{5}$, $\alpha_1 = \frac{1}{5}$.

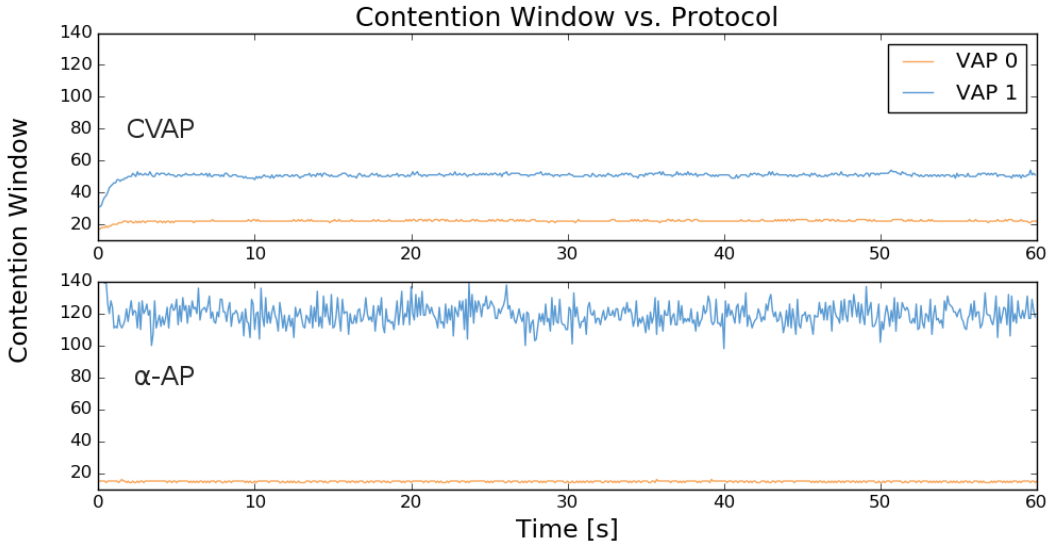

Figure 4.4: Contention Window by Time for MAC protocols `CVAP` and `α-AP` with 2 VAPs with 2, 5 stations respectively, and weights $\{\alpha_0 : \frac{4}{5}, \alpha_1 : \frac{1}{5}\}$

We clearly see both CVAP and AlphaAP achieve a rapid adaptation of the CW from simulation start to the optimal CW in both VAP 0 and VAP 1. VAP 0 achieves the optimal CW very promptly in both protocols while VAP 1, containing 5 associated

STAs rather than 2, adapts at a slightly slower rate; still achieving the optimal value in less than 2 seconds, a suitably short duration. We see no significant deviations in the CW values over the 60 second time period in either CVAP or AlphaAP. However, we do note that AlphaAP shows far greater variation in CW responses for VAP 1 than that of CVAP. This shall be examined further below.

Focusing on AlphaAP, we note that the CW range between VAP 0 and VAP 1 is larger than that of CVAP. VAP 1 has a mean CW value of 119 (compared to a value of 50 in CVAP), and VAP 0 has a mean CW value of 14 (compared to 22 in CVAP). The large range is due to the values of weights allocated to VAPs and the number of STAs connected to each. Assigning a lower weight to VAP 1, with more STAs associated than VAP 0, requires a larger CW value to achieve the desired weighted throughput. We observe the CVAP scenario is roughly equivalent to assigning AlphaAP weights of $\frac{1}{2}$ to each VAP. In scenarios where small weights are assigned to a VAP with large numbers of associated STAs, AlphaAP may assign very large CW values to satisfy the conditions. Large CW values increase the delay in accessing the medium for STAs associated to these VAPs.

Next, we shall examine further how the weight distributions affect the CW target. We compare the change in CW produced by the protocol in the system as in the scenario in Fig. 4.4, but with two different weight distributions for these VAPs.

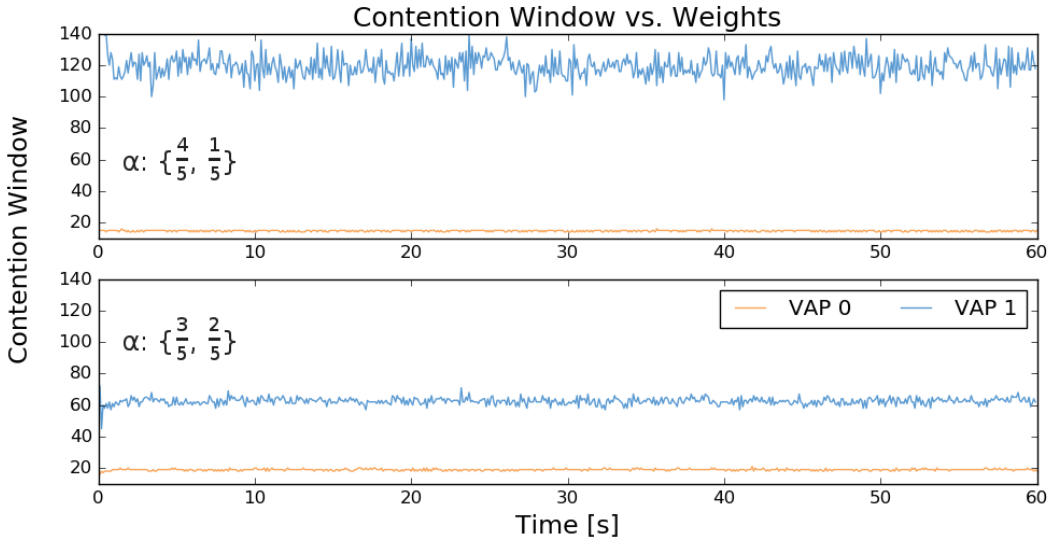

Figure 4.5: Contention Window variation in scenario with 2 VAPs, each with 2 and 5 stations. (Above) has weights $\frac{4}{5}, \frac{1}{5}$ and (below) has weights $\frac{3}{5}, \frac{2}{5}$.

In both scenarios, we see a rapid adjustment of the CW value for each VAP, with the CW reaching optimum value within 1 second for each weight scenario. We note that assigning weights with smaller differences (as seen in $\alpha_0 = \frac{3}{5}, \alpha_1 = \frac{2}{5}$), results in a smaller difference between the optimal CW of VAP 0 and VAP 1. Additionally, we see less variation over time in the CW in this distribution when compared to the larger optimal CW of VAP 1 in the scenario of $\alpha_0 = \frac{4}{5}, \alpha_1 = \frac{1}{5}$.

The PI controller driving the CW makes use of two constants, K_P and K_I, to configure

the responsiveness of the controller. K_P, the proportional constant, may be modified to control the magnitude of the response to a given error signal. K_I, the integral constant, is frequently seen to control this response over time, allowing the system to settle at a stable point. To examine the suitability of our choices of K_P, K_I, Fig. 4.6 presents the CW of a VAP over time with these constants of varying magnitudes.

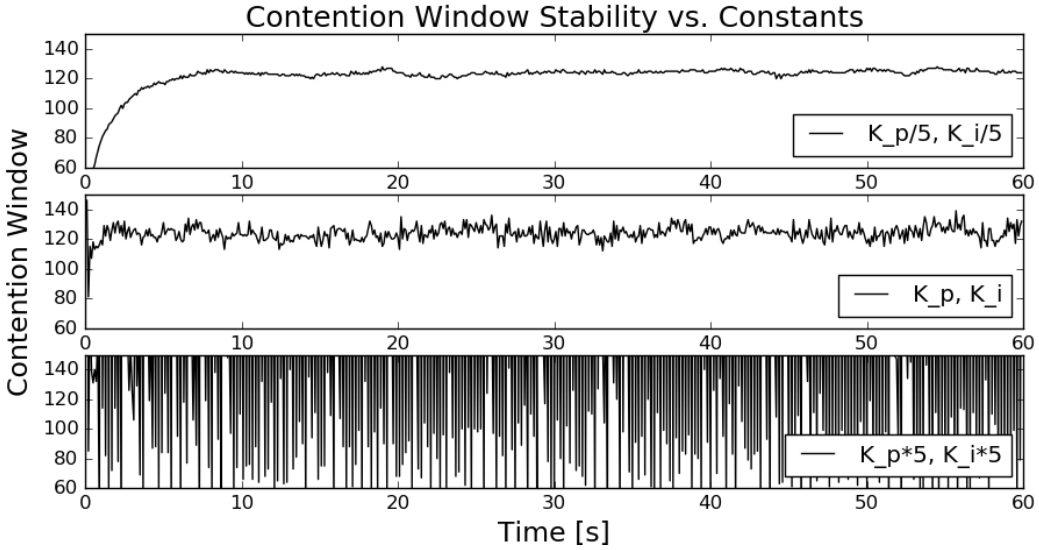

Figure 4.6: Contention Window by Time with $\{K_P/5, K_I/5\}$, $\{K_P, K_I\}$, and $\{K_P * 5, K_I * 5\}$

Fig. 4.6 indicates the constants K_P, K_I chosen for the PI controller remain optimal, as in CVAP. As seen in the upper plot, assigning a smaller constant greatly increases the settling time for the CW to reach optimum value, decreasing the reaction speed to dynamic situations. In the lower plot, a larger constant results in drastic fluctuations that would prevent reasonable functioning of the system. The central plot indicates our chosen constants and remains a fairly stable choice.

As a measure of indicating the suitability of the proposed AlphaAP protocol we modify the Jain Fairness Index, as produced by Jain et al. [14], for the weighted throughput achieved by each VAP. Considering a scenario with N VAPs, and R_i, α_i, the throughput achieved and weight given for VAP$_i$ respectively, we calculate this as,

$$J_\alpha = \frac{(\sum_{i=1}^{N} \frac{R_i}{\alpha_i})^2}{N \cdot \sum_{i=1}^{N} (\frac{R_i}{\alpha_i})^2}. \tag{4.1}$$

Fig. 4.7 presents the weighted JFI for the same scenarios as in Fig. 4.1. The error bar for each of these results is too small to be visible on the plot and so is not displayed. Note the weight distributions given beside the plot, and associated numbers of stations.

The results shows that AlphaAP consistently achieves a weighted JFI of 1 in each scenario. This indicates AlphaAP successfully assigns weighted throughput to VAPs based on assigned weights. AlphaAP performs successfully in the presence of varying numbers of VAPs and associated STAs. The results also confirm the motivation for

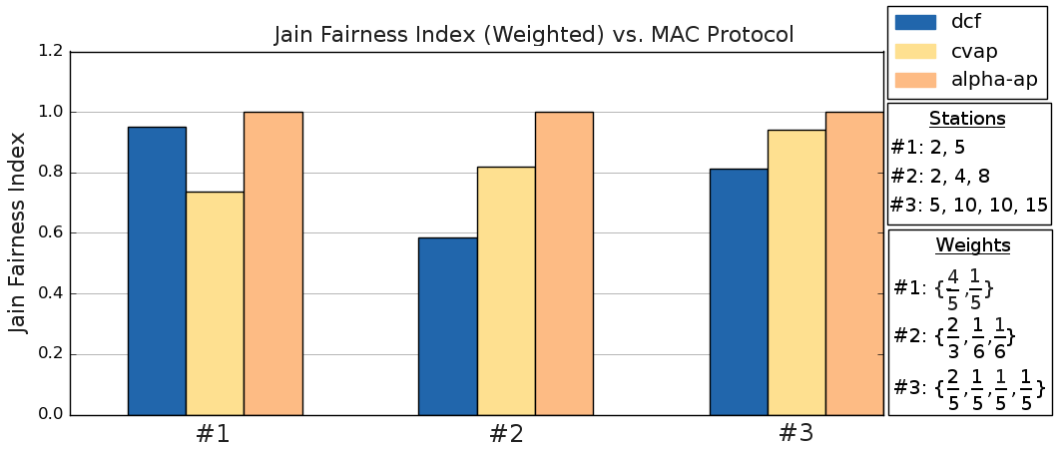

Figure 4.7: Weighted Jain Fairness Index measurements for each MAC protocol and varying numbers of associated stations and VAP counts.

AlphaAP; with the results of DCF and CVAP failing to obtain the weighted throughput allocations configured in each scenario.

We now move to exploring the responsiveness of the system. WLANs are variable systems with STAs associating or disassociating over time. Each VAP should adapt the CW as STAs associate or disassociate, in a timely manner, to continue to achieve the desired weighted throughput. Fig. 4.8 explores the responsiveness of two VAPs beginning with 2 and 5 STAs respectively. One STA disassociates from VAP 0 at 15s, two STAs associate to VAP 1 at 30s, and two STAs associate to VAP 1 at 45s. Red lines indicate the point of these occurrences. Note that weights for this scenario are $\alpha_0 : \frac{3}{5}, \alpha_1 : \frac{2}{5}$, as in the lower case of Fig. 4.5.

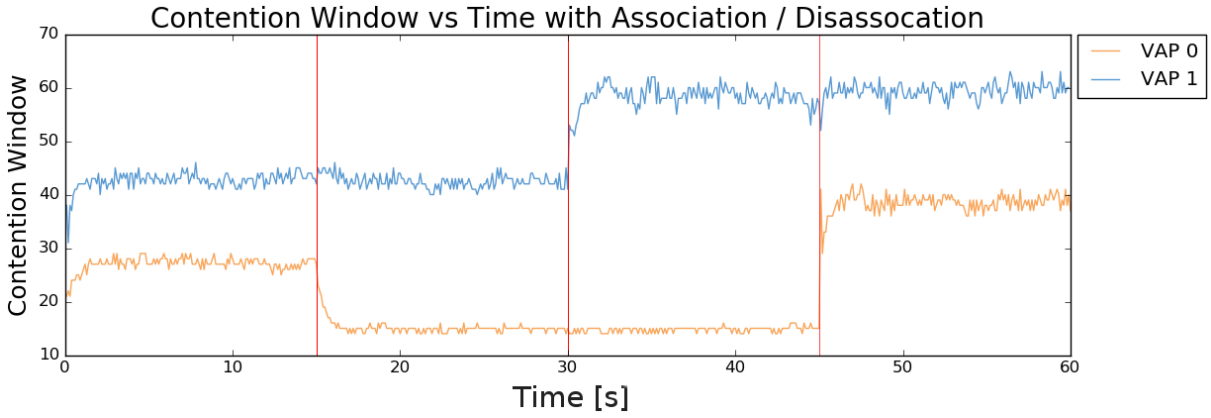

Figure 4.8: Contention Window responsiveness with 2 VAPs with 2 and 5 Stations respectively, with a STA disassociating at 15s, and 2 STAs associating at 30s, followed by 2 more at 45s.

This plot presents a snapshot of a dynamic situation commonly observed within a WLAN. VAP 0 drops from two associated STAs to one at 15 seconds into the simulation, and promptly drops the optimal CW to continue meeting the target weighted throughput with only one associated STA. At 30 seconds in, VAP 1 has two more

STAs associate, resulting in seven associated STAs, whilst VAP 0 still has only one associated. We note VAP 0 experiences no fluctuation in CW with this change, with the only effect being VAP 1 increasing the CW to a larger value. At 45 seconds, two new STAs associate to VAP 0, increasing its CW, while VAP 1 retains the current target CW. After each new batch of associations or disassociations, AlphaAP drives the CW to a new optimal value within 1.5 seconds, indicating suitable response times to changes in WLAN conditions.

Through previous evaluation, we have seen CW assignments include larger values if the weight distributions and STA numbers are unbalanced. As the CW of a VAP increases, the range of transmission delays that STAs experience is increased. Given the variation in CW observed in Fig. 4.5, we can observe the Medium Access Delay in these two weight distributions via a cumulative distribution function, as given in Fig.4.9.

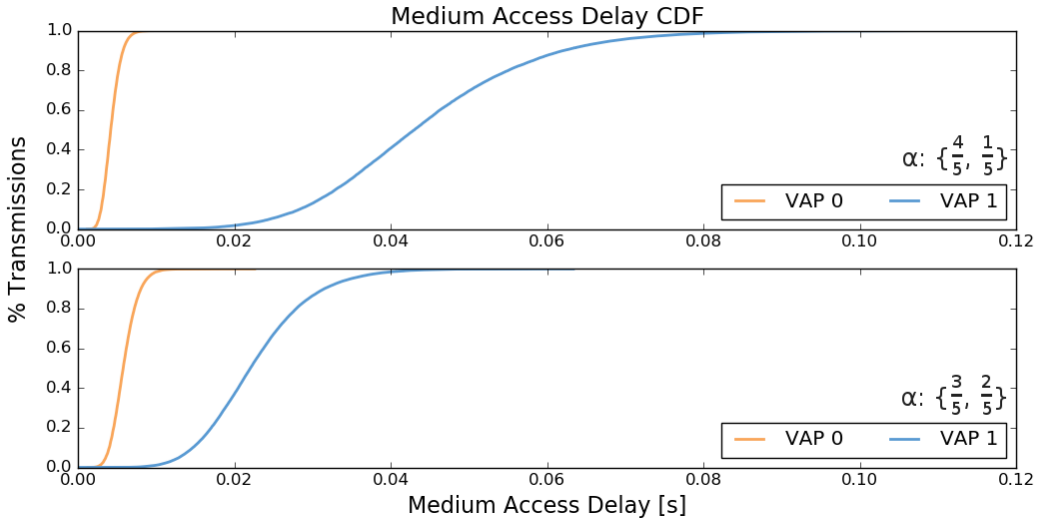

Figure 4.9: Cumulative Distribution Function for the Medium Access Delays in scenario with 2 VAPs, with 2 and 5 associated stations, and weights given as indicated within each plot.

It is clear from the cumulative distribution functions that the unbalanced weights in the upper scenario result in VAP 1 having larger medium access delays overall, and a wider range of delays. VAP 1 contains a larger number of associated STAs (five of the seven present in the scenario), and AlphaAP drives the contention window larger to assign $\frac{1}{5}$ of the resources. Compared to the more balanced weight distribution of the second scenario, where VAP 1 is assigned a weight of $\frac{2}{5}$, the larger delays are clear. VAP 0 remains fairly similar in both scenarios, with a slight increase in delays in the lower scenario as the weight assigned decreases to $\frac{3}{5}$ from $\frac{4}{5}$.

We note this confirms previous intuition from observing the contention window variability in Fig. 4.5. Larger contention window values not only cause increased medium access delays, but also increase variation in the delay, imposing greater jitter on STAs connected to these VAPs. This could prove a threat to the quality of service provided to clients. For example, a popular VAP with large numbers of associated STAs, that is

configured to have a small share of the throughput will drive the contention window higher, resulting in increased delays. ITU Recommendations [13] indicate a one-way delay of less than 100ms is suitable for interactivity and up to 400ms is acceptable for real-time applications. Fig. 4.5 indicates STAs in each of these scenarios fall within these recommendations.

Chapter 5

Implementation Consideration

In the following section, we describe certain considerations that must be met when implementing AlphaAP on physical hardware. This serves as a discussion on the practicality of the algorithm and possible avenues for future research.

5.1 Beaconing

We first consider the impact of beacon frames. AlphaAP controls a set of N Virtual Access Points (VAPs). It does this by transmitting new contention window values every beacon interval. In our simulation, we abstracted this concept and updated the contention window for all associated stations (STAs) directly. However, in the presence of real access points, this would need to be transmitted in the EDCA parameter set of a beacon management frame, as set out in 802.11 Std Section 7.3.2.29 [7].

Beacon frames are a well established concept in 802.11 and are used for many purposes including timing, access point awareness, and configuration. AlphaAP only adjusts the parameters sent within beacon frames and does not require any extra transmissions to operate. However, as these beacon frames are used by AlphaAP, we give brief consideration to the channel occupancy overhead of them when N VAPs operate simultaneously.

With N VAPs present in the system we must transmit N beacon frames every beacon interval, which is typically set to 100ms. A beacon frame must be sent using the minimal data rate so as to be understood by all possible associated STAs. Typically, this will be 1 Mb/s.

Given this, we can consider the overhead of beacon frames for the VAPs within each beacon interval. Each beacon frame must be preceded by a SIFS interval; the period awaited by an AP before transmitting a management frame. Beacon frames have variable header sizes, and as such, no fixed length.

With an average beacon frame size of L bytes, and beacons transmitted at a basic rate

of 1 Mb/s, each beacon frame takes duration, in milliseconds,

$$T_d = \frac{SIFS}{1000} + 1000 \cdot \frac{8 \cdot L}{1024^2}. \tag{5.1}$$

Given N VAPs, the total time allocated to beacon transmissions is,

$$T_b = N \cdot T_d. \tag{5.2}$$

With T, the beacon interval in milliseconds, the percentage overhead of beacon frame transmissions per beacon interval is,

$$Overhead = \frac{T_b}{T} = \frac{N}{T} \cdot T_d = \frac{N}{T}(\frac{SIFS}{1000} + 1000 \cdot \frac{8 \cdot L}{1024^2}). \tag{5.3}$$

In a capture of 953 beacon frames in an environment with 8 distinct Access Points, the average beacon frame length, including frame headers, was 291 bytes. Given this, we can observe the percentage overhead per beacon interval for varying numbers of VAPs below.

Number of VAPs	Overhead (%)
1	2.236
2	4.472
3	6.708
4	8.945
5	11.181

Table 5.1: Percentage of the beacon interval occupied by beacon frame transmissions.

As the number of VAPs increase, a larger portion of transmission time is occupied by beacon frames. This imposes a natural limit on the number of VAPs that may be situated on a single device. An option to alleviate this as the number of VAPs increase is to increase the beacon interval duration. Further research could consider the adaptiveness of AlphaAP when the beacon interval is increased for a collection of VAPs.

5.2 Information Access

Further considerations lie in the implementation of the algorithm on physical devices. No modifications are required to associated stations (STA); each simply takes the contention window parameters broadcast in beacon frames and applies these to subsequent transmission attempts.

However, for each virtual access point, the physical device implementing the algorithm must run a PI controller driven by certain information close to the MAC layer. Specifically, the PI controller at each VAP makes use of the success probability of the associated STAs, that is, the probability of a given slot containing a successful transmission from any STA associated with the VAP. Each VAP must also have access to

this information for other VAPs running on the same device. Finally, each VAP must have access to the number of idle and busy slots.

Typically, this lower level information is not accessible to the network drivers for devices. To access this information and run a suitable controller to implement AlphaAP, it may be required to modify the firmware of a device to either supply this information to the driver, or implement the controller on the firmware itself.

Additionally, it should be noted that the PI controller runs with minimal overhead for each VAP. At each beacon interval, each VAP must run a computation with $O(N)$ complexity, where N is the number of VAPs. Given the discussion in the previous section on beaconing, N is sufficiently small.

Chapter 6

Conclusion

In this thesis, we successfully introduced AlphaAP, a new MAC protocol that allows for arbitrary weighted throughput allocation among Virtual Access Points (VAP). AlphaAP resolves the issue of unfairness in virtualisation techniques for WLANs.

After reviewing the existing research, we created a system model for AlphaAP and through theoretical analysis validated the usage of a Proportional-Integral (PI) controller to drive the system to a suitable point of operation. A system-level simulator was constructed that focused on the MAC protocol of 802.11. This was used to run a variety of scenarios including differing numbers of VAPs and associated stations (STA), and varying weight allocations.

These scenarios were used to evaluate the performance of AlphaAP against the Distributed Coordination Function (DCF) and CVAP. DCF is the basic MAC protocol provided by the 802.11 standards, and CVAP acted as the foundation for AlphaAP as it achieves equal throughput among VAPs. Results showed AlphaAP successfully allocated throughput to VAPs based on specified weights, while continuing to achieve a total throughput greater than DCF and comparable to CVAP. A modified Jain Fairness Index measure was used to confirm the weighted fairness of the throughput allocations.

Having satisfied the original objective, we investigated the impact of AlphaAP on contention window stability, medium access delays, and responsiveness to STA associations or disassociations. We established that AlphaAP drives the contention window appropriately quickly and remains stable once the optimal value is reached. However, for unabalanced weight distributions AlphaAP can drive the contention window to a large value, which may cause jitter and larger medium access delays.

Further areas of research were highlighted, including investigating the performance of AlphaAP in the presence of real-time traffic requirements, TCP traffic, or multi-rate STAs. We also presented considerations for the physical implementation of AlphaAP.

AlphaAP offers a solution to throughput allocation in virtualisation, with real-world applications for negotiating infrastructure sharing for hotspot data offloading. As AlphaAP conforms to the IEEE 802.11 standards, it could be implemented with minimal footprint and immediate compatibility with existing physical infrastructure.

Appendices

Appendix A: PI Controller Error Solution

We follow the same proof as in CVAP [9]. Using the definition of e_i as in Eq. 3.17, we wish to show there exists a unique solution to $e_i = 0 \ \forall i$, such that $e_{opt} = 0$ and $e_{fair,i} = 0 \ \forall i$. That is, there exists a condition for the system of PI controllers where the error can be set to 0 and both our error criteria are met.

We begin by subtracting e_j from e_i giving,

$$e_i - e_j = \frac{S_i}{\alpha_i} - \frac{S_j}{\alpha_j}.$$

As $e_i = 0 \ \forall i, j$ we have that $\frac{S_i}{\alpha_i} = \frac{S_j}{\alpha_j} \ \forall i, j$. Using this, we show $e_{fair,i} = 0 \ \forall i$ as follows,

$$\begin{aligned} e_{fair,i} &= \frac{S_i}{\alpha_i} - \sum_{k=1}^{N} S_k \\ &= \frac{S_i}{\alpha_i} - \sum_{k=1}^{N} \frac{\alpha_k}{\alpha_i} S_i \\ &= \frac{S_i}{\alpha_i} - \frac{S_i}{\alpha_i} \sum_{k=1}^{N} \alpha_k \\ &= 0, \end{aligned}$$

as $\sum_{k=1}^{N} \alpha_k = 1$ by Eq. 3.5. Using the goal condition defined in Eq. 3.7, $\frac{n_i \tau_i}{\alpha_i} = \frac{n_j \tau_j}{\alpha_j}$, and observing that α_i is a fixed constant $\forall i$, we have for each (n_i, n_j) pairing a direct relationship between τ_i and τ_j, as observed in [9]. Thus, representing $e_{opt} = 0$ as,

$$\begin{aligned} P_e &= P_e^* \\ \prod_{k=1}^{N} (1 - \tau_k)^{n_k} &= P_e^*, \end{aligned}$$

we observe the same relation where P_e^* is a fixed constant, the point of operation, between 0 and 1, and the left hand side of the equation is a decreasing function from 1 to 0. Thus, there exists a unique solution where this equation is satisfied and $e_{opt} = 0$, fulfilling the objective.

Bibliography

[1] *Cisco - Visual Networking Index (VNI): Global Mobile Data Traffic Forecast Update, 2014-2019*. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf.

[2] *Cisco - What Do Consumers Want From Wi-Fi?* http://www.cisco.com/web/about/ac79/docs/sp/SP_Wi-Fi_Consumers.pdf.

[3] *Ericsson Mobility Report, November 2015*. http://www.ericsson.com/res/docs/2015/mobility-report/ericsson-mobility-report-nov-2015.pdf.

[4] *NS-3 Simulator*. https://www.nsnam.org/.

[5] *OMNeT++*. https://omnetpp.org/.

[6] *Pamvotis*. http://pamvotis.org/.

[7] 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 (Revision of IEEE Std 802.11-1999)*, pages 1–1076, June 2007.

[8] Imad Aad and Claude Castelluccia. Differentiation mechanisms for IEEE 802.11. In *Proceedings IEEE INFOCOM 2001, The Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Twenty years into the communications odyssey, Anchorage, Alaska, USA, April 22-26, 2001*, pages 209–218. IEEE, 2001.

[9] Albert Banchs, Pablo Serrano, Paul Patras, and Marek Natkaniec. Providing Throughput and Fairness Guarantees in Virtualized WLANs Through Control Theory. *MONET*, 17(4):435–446, 2012.

[10] Gautam D. Bhanage, Dipti Vete, Ivan Seskar, and Dipankar Raychaudhuri. SplitAP: Leveraging Wireless Network Virtualization for Flexible Sharing of WLANs. In *Proceedings of the Global Communications Conference, 2010. GLOBECOM 2010, 6-10 December 2010, Miami, Florida, USA*, pages 1–6. IEEE, 2010.

[11] Giuseppe Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *Selected Areas in Communications, IEEE Journal on*, 18(3):535–547, 2000.

[12] IEEE. Official IEEE 802.11 Working Group Project Timelines. `http://www.ieee802.org/11/Reports/802.11_Timelines.htm`.

[13] International Telecommunication Union (ITU-T). Network Performance Objectives for IP-based Services (Y.1541). 2011.

[14] Raj Jain, Arjan Durresi, and Gojko Babic. Throughput fairness index: An explanation. Technical report, Tech. rep., Department of CIS, The Ohio State University, 1999.

[15] Jiwoong Jeong, Sunghyun Choi, and Chong-kwon Kim. Achieving Weighted Fairness between Uplink and Downlink in IEEE 802.11 DCF-Based WLANs. In *Second International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QSHINE 2005), 22-24 August 2005, Lake Buena Vista, FL, USA*, page 22. IEEE Computer Society, 2005.

[16] Pradeep Kyasanur and Nitin H. Vaidya. Selfish MAC Layer Misbehavior in Wireless Networks. *IEEE Trans. Mob. Comput.*, 4(5):502–516, 2005.

[17] Chengchao Liang and F. Richard Yu. Wireless Network Virtualization: A Survey, Some Research Issues and Challenges. *IEEE Communications Surveys and Tutorials*, 17(1):358–380, 2015.

[18] 2015) Marc Patterson (May 8th. *Transforming Airport Networks with Virtualization (Web Log Post)*. `http://www.boingo.com/blog/2015/05/transforming-airport-networks-with-virtualization/`.

[19] Marc Portoles-Comeras, Zhun Zhong, and Sunghyun Choi. IEEE 802.11 Downlink Traffic Shaping Scheme for Multi-User Service Enhancement. In *Proceedings of the IEEE 14th International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2003, 7-10 September 2003, Beijing, China*, pages 1712–1716. IEEE, 2003.

[20] Antonio Visioli. *Practical PID Control*. Springer, 2006.

[21] C. Wang, H. K. Lo, K. Liang, and C. Hsu. A Cross-layer Design for per-flow and Weighted Fairness between Uplink and Downlink in WLANs. In *Communication Systems (ICCS), 2010 IEEE International Conference on*, pages 421–425, Nov 2010.

[22] Mao Yang, Yong Li, Depeng Jin, Lieguang Zeng, Xin Wu, and Athanasios V. Vasilakos. Software-Defined and Virtualized Future Mobile and Wireless Networks: A Survey. *MONET*, 20(1):4–18, 2015.

[23] Yi Yao, Bo Sheng, and Ningfang Mi. DAT: An AP Scheduler using Dynamically Adjusted Time Windows for Crowded WLANs. In Sheng Zhong, Dejing Dou, and Yu Wang, editors, *30th IEEE International Performance Computing and*

Communications Conference, IPCCC 2011, Orlando, Florida, USA, November 17-19, 2011, pages 1–6. IEEE, 2011.

[24] Xiangyi Zou, Lianying Wang, Liqiang Zhao, and Hailin Zhang. Using a Constant Contention Window to Maximize the Throughput of IEEE 802.11 WLANs. In *2007 IET Conference on Wireless, Mobile and Sensor Networks (CCWMSN07)*, 2007.