Design and Optimization of QoS-based Medium Access Control Protocols for Next-Generation Wireless LANs

Dionysios Skordoulis

School of Engineering and Design

Brunel University

A thesis submitted for the degree of

Doctor of Philosophy

February 2013

I dedicate this thesis to the memory of my mother

Carol Constance Kemp (1945 – 2004).

From my earliest days, she taught me to believe in myself and to strive for the highest goals. I owe every bit of my existence to her.

Acknowledgements

I am indebted to many individuals for their care and support given to me during my doctoral studies. First and foremost, I would like to express my deep gratitude to my PhD supervisor Dr. Qiang Ni, who has provided me with constant encouragement, insightful advice, and invaluable suggestions. This work would not have been possible without him and his tremendous willingness to anticipate.

I am also grateful to Brunel University, the School's Department of Electronic & Computer Engineering (ECE), and the Engineering & Physical Sciences Research Council (EPSRC) for trusting me and providing me with financial support during the years of my doctoral study.

My special thanks should also go to my father, sister, relatives (especially to my aunt and cousins) and friends since they always have loved me, believed in me, and encouraged me throughout my studies. Last but not least, a final acknowledgement goes to my future beloved wife Ioanna and her family for their support, understanding, and encouragement.

Abstract

In recent years, there have been tremendous advances in wireless & mobile communications, including wireless radio techniques, networking protocols, and mobile devices. It is expected that different broadband wireless access technologies, e.g., WiFi (IEEE 802.11) and WiMAX (IEEE 802.16) will coexist in the future. In the meantime, multimedia applications have experienced an explosive growth with increasing user demands. Nowadays, people expect to receive high-speed video, audio, voice and web services even when being mobile. The key question that needs to be answered, then, is how do we ensure that users always have the "best" network performance with the "lowest" costs in such complicated situations?

The latest IEEE 802.11n standards attains rates of more than 100 Mbps by introducing innovative enhancements at the PHY and MAC layer, e.g. MIMO and Frame Aggregation, respectively. However, in this thesis we demonstrate that frame aggregation's performance adheres due to the EDCA scheduler's priority mechanism and consequently resulting in the network's poor overall performance. Short waiting times for high priority flows into the aggregation queue resolves to poor channel utilization. A Delayed Channel Access algorithm was designed to intentionally postpone the channel access procedure so that the number of packets in a formed frame can be increased and so will the network's overall performance. However, in some cases, the DCA algorithm has a negative impact on the applications that utilize the TCP protocol, especially the when small TCP window sizes are engaged. So, the TCP process starts to refrain from sending data due to delayed acknowledgements and the overall throughput drops.

In this thesis, we address the above issues by firstly demonstrating the potential performance benefits of frame aggregation over the next-generation wireless networks. The efficiency and behaviour of frame aggregation within a single queue, are mathematically analysed with the aid of a $M/G^{[a,b]}/1/K$ model. Results show that a trade-off choice has to be taken into account over minimizing the waiting time or maximizing utilization. We also point out that there isn't an optimum batch collection rule which can be assumed as generally valid but individual cases have to be considered separately. Secondly, we demonstrate through extensive simulations that by introducing a method, the DCA algorithm, which dynamically determines and adapts batch collections based upon the traffic's characteristics, QoS requirements and server's maximum capacity, also improves efficiency. Thirdly, it is important to understand the behaviour of the TCP flows over the WLAN and the influence that DCA has over the degrading performance of the TCP protocol. We investigate the cause of the problem and provide the foundations of designing and implementing possible solutions. Fourthly, we introduce two innovative proposals, one amendment and one extension to the original DCA algorithm, called Adaptive DCA and Selective DCA, respectively. Both solutions have been implemented in OPNET and extensive simulation runs over a wide set of scenarios show their effectiveness over the network's overall performance, each in its own way.

Supporting Publications

Journal and Magazines

1. **D. Skordoulis**, Q. Ni, H. Chen, A. 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, 2008. [1]

Conference Papers

1. **D. Skordoulis**, Q. Ni, and C. Zarakovitis, "A Selective Delayed Channel Access (SDCA) for the High-Throughput IEEE 802.11n," *in Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE.* IEEE, 2009, pp. 1–6.

2. **D. Skordoulis**, Q. Ni, G. Min, and K. Borg, "Adaptive Delayed Channel Access for IEEE 802.11n WLANs," *in Circuits and Systems for Communications, 2008. ICCSC 2008. 4th IEEE International Conference on.* IEEE, 2008, pp. 167–171.

3. **D. Skordoulis**, Q. Ni, U. Ali, and M. Hadjinicolaou, "Analysis of Concatenation and Packing Mechanisms in IEEE 802.11n," *in Proceedings of the 6th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNET07)*, 2007.

Posters

1. **D. Skordoulis**, Q. Ni and C. Zarakovitis, "Various Delay Channel Access algorithms for the IEEE 802.11n", *Brunel University's Graduate School Research Student Poster Conference*, 6-7 May 2009 [2]

[1]This article was cited over 100 times at Google Scholar in 2012
[2]Awarded with the "Best Poster Award" (ranked in Top-6)

Contents

List of Figures

List of Tables

List of Symbols

AC_i	access category with index i.
$B(t)$	service time distribution function.
D	deterministic distribution.
D_i	the saturation delay for the priority i class.
$E(B)$	mean service time.
$E(I)$	expected server's "idle" period.
$E(L_q)$	mean queue length.
$E(O)$	expected server's "busy" period.
$E(S)$	mean number of packets in the server.
$E(S^A)$	mean number of packets per start.
$E(W)$	mean waiting time.
$E(X)$	expected value of r.v. X.
$FS\text{-}to\text{-}FS\ Interval$	the time (in μs) to transmit one data frame with ACK.
$Frame\ Rate$	the number of frames per second that can be transmitted across.
G	general distribution of the service times.
H_2	hyperexponential distribution.
I	channel idle period in slot time.
IAT	stands for inter-arrival time but within this text it is used to denote a DCA condition ($t - T_L < T_B$).
K	size of capacity queue (waiting space).
KB	Kilobyte (1 KB is equivalent to 1 000 bytes).
$L_{i,retry}$	the retry limit for the priority i class.
L_{packet}	MSDU length in bytes (octets).
L_{pad}	the PPDU pad bit field (size in bits).
L_{tail}	the PPDU tail bit field (size in bits).
M	exponential distribution of the interarrival times.
Mb/s	Megabit per second (1 Mb is equivalent to 1 000 000 bits).
N	total number of ACs.

N_{MSDU}	the number of packets from upper layer that are still in the aggregation buffer.
N_{dbps}	number of data bits per symbol (function of PHY rate).
$N_{i,retry}$	the r.v. representing the total number of retries for the priority i class.
P_B	blocking probability.
R_x	receiver of the station or node.
S_i	the normalized saturation throughput for the priority i class.
T_B	the inter-arrival time of the traffic burst under review.
T_F	the arrival time of the first packet from upper layer in the current aggregation buffer.
T_H	the time to transmit the MAC & PHY headers plus any pads or tails.
T_L	the arrival time of the last packet from upper layer.
T_c	collision period in slot time.
T_o	the time of the sender's time-out.
T_s	successful transmission period in slot time.
T_x	transmitter of the station or node.
T_{ACK}	time (in μs) to transmit an acknowledgement.
T_{CA}	the channel-access starting time for an aggregate.
$T_{E(L_{packet})}$	the time to transmit the average payload.
$T_{E(L_{packet^*})}$	the time to transmit the payload of the longest frame that has collide.
T_{TX}	the transmission starting time for an aggregate.
$T_{preamble}$	time (in μs) to transmit a PLCP preamble.
T_{signal}	time (in μs) to transmit a Signal Field.
T_{slot}	the duration of a time slot set by the physical layer encoding method in-use.
T_{symbol}	time (in μs) to transmit a Service Field (define symbol clock and code).
$T_{ACK_{Timeout}}$	time (in μs) to receive an ACK.
$TxTime$	time (in ms) to transmit one data frame.
$W_{i,j}$	the current contention window size of AC_i in stage j.
X^2	chi-square distribution.
X_i	the r.v. representing the total number of slots when the counter freezes for the priority i class.

X_i	the r.v. representing the total number of backoff slots for the priority i class.
δ	the time of the propagation delay.
σ_{incr}	the increment value of σ each time it decreases or increases.
σ_{max}	the maximum value that σ can be assigned to.
σ_{min}	the minimum value that σ can be assigned to.
a	threshold of the server starting rule.
b	server capacity.
$b(i,t)$	a random process representing the value of the backoff counter at time t.
$b_{i,j,l}$	the stationary distribution of the Markov chain.
c_B	coefficient of variation of service time.
c_X	coefficient of variation of r.v. X.
i	index of the priority class.
j	backoff stage.
l	backoff delay in time slots.
n_i	number of STAs in a given priority class i.
p_b	the probability that the channel is busy.
p_i	the probability that at a AC_i a transmitted frame collides and the correlated STA senses the medium busy.
p_s	the probability that a successful transmission occurs in a slot time.
$p_{s,i}$	the probability that a successful transmission occurs in a slot time for the priority i class.
$p_{t,i}$	the probability that a station in the AC_i priority class transmits during a generic slot time.
$s(i,t)$	a random process representing the backoff stage j.
t	timer.
γ	a ratio of the inter-arrival time to the channel access delay.
λ	average(mean) arrival rate.
μs	microsecond is an SI unit of time equal to one millionth of a second.
μ	average(mean) service rate.
ϕ	an oscillator factor that controls consecutive σ triggers.
ψ	an oscillator factor that controls consecutive IAT triggers.
ρ	server utilization factor.

σ	the maximum number of packets in the aggregation buffer before aggregation is triggered.
τ	the maximal waiting time for a packet in the aggregation buffer.

List of Acronyms

A-MPDU Aggregated MAC Protocol Data Unit.
A-MSDU Aggregated MAC Service Data Unit.
AC Access Category.
ACK Acknowledgement.
ADCA Adaptive DCA.
AIFS Arbitrary Inter-Frame Space.
AIFSN AIFS-number.
AP Access Point.

BAR Block Acknowledgement Request.
BDP Bandwidth-Delay Product.
BE Best Effort.
BER Bit-Error-Rate.
BK Background.
BlockAck Block Acknowledgement.

CCA Clear Channel Assessment.
CFB Controlled Frame-Bursting.
CPR Constant Packet Rate.
CRC Cyclic Redundancy Check.
CSMA/CA Carrier Sense Multiple Access with Collision avoidance.
CTS Clear To Send.
CW Contention Window.
cwnd Congestion Window.

D-ITG Distributed Internet Traffic Generator.
d.f. distribution function.
DA Destination Address.
DCA Delayed Channel Access.
DCF Distributed Coordination Function.
DES Discrete Event Simulation.
DIFS DCF Inter-Frame Space.

EDCA Enhanced Distributed Channel Access.
ETSI European Telecommunications Standards Institute.

FIFO First-In-First-Out.
FIN Finalize.
FSM Finite State Machine.
FTP File Transfer Protocol.

HCCA HCF Controlled Channel Access.
HCF Hybrid Coordination Function.
HDTV High-Definition television.
HT High Throughput.
HTSG High-Throughput Study Group.

i.i.d. independent identically distributed.
IANA Internet Assigned Numbers Authority.
IBSS Independent Basic Service Set.
ICI Interface Control Information.
IDT Inter-Departure Time.
IEEE Institute of Electrical and Electronics Engineers.
IETF Internet Engineering Task Force.
IF Internet File.
IFS Inter-Frame Space.
IP Internet Protocol.

KP Kernel Procedure.

L.S.T. Laplace-Stieltjes transform.
LLC Logical Link Control.
LoS Line of Sight.

MAC Media Access Control.
MCS Modulation and Coding Scheme.
MIMO Multiple-Input / Multiple-Output.
MMPP Markov-Modulated Poisson Process.
MPDU MAC Protocol Data Unit.
MS Microsoft.
MSDU MAC Service Data Unit.
MSS Maximum Segment Size.
MTU Maximum Transmission Unit.

NP Non-deterministic Polynomial-time.

OFDM Orthogonal Frequency Division Multiplexing.

OL Offered Load.
OPNET Optimized Network Engineering Tool.
OS Operating System.
OSI Open Systems Interconnection.

P2P peer-to-peer.
PASTA Poisson Arrivals See Time Averages.
PCF Point Coordination Function.
PHY Physical.
PLCP Physical Layer Convergence Protocol.
PLR Packet Loss Rate.
PPDU PLCP Protocol Data Unit.
PSDU PHY Service Data Unit.
PSMP Power Save Multi-Poll.

QAM Quadrature Amplitude Modulation.
QoS Quality of Service.

r.v. random variable.
RA Receiver Address.
RD Reverse Direction.
RF Radio Frequency.
RTO Retransmission Time-Out.
RTS Request To Send.
RTT Round Trip Time.
rwnd Receiver Window.

SA Sender Address.
SAP Service Access Point.
SDCA Selective DCA.
SDTV Standard-Definition television.
SIFS Short Inter-Frame Space.
SISO Single-Input / Single-Output.
SNR Signal-to-Noise Ration.
ssthresh slow start threshold.
STA station.
STD State Transition Diagram.
SVM Support Vector Machine.
SYN Synchronize.

TA Transmitter Address.
TCP Transmission Control Protocol.

TGn Task Group N.
TID Traffic Identifier.
ToS Type of Service.
TSPEC Traffic Specification.
TTL Theoretical Throughput Limit.
TUL Throughput Upper Limit.
TXNAV Transmitter Network Allocation Vector.
TXOP Transmission Opportunity.

UDP User Datagram Protocol.
UP User Priority.

VI Video.
VO Voice.
VoIP Voice over IP.
VPR Variable Packet Rate.

WG Working Group.
WLAN Wireless Local Area Network.
WWW World Wide Web.

ZF Zero Forcing.

Chapter 1

Introduction

Over the last decade, the use of wireless and mobile devices has expanded rapidly. The advantages that these systems possess, such as interoperability, mobility, flexibility and cost effective, have gained a huge support across enterprises, homes, and service providers. Mobile wireless connectivity has changed our lifestyles dramatically. It allows people to transmit information over the "air" no matter how the protocols are designed, what data they want to share, or where their devices are physically located. The preliminary inspiration for wireless access was initiative for Mobile Internet and it seems that this has changed our lives almost as much as the advent of the Internet and World Wide Web (WWW) itself. Three major factors have had a great input to this evolution: allocation of unlicensed frequency bands, cheaper wireless components and standardization.

The most popular wireless networks are the Wireless Local Area Networks (WLANs). Examples of such networks can be found not only in major corporations but also in universities, hospitals, airports, libraries, hotels, residences and even in local shops. WLANs are considered as a viable communication system and excellent complement to wired ones, studies so that users have the habit to use wireless network even for heavy bandwidth applications, such as streaming or file sharing, despite the presence of a high-speed wired fibre-optic links. Evidently, the convenience of a wireless solution outweighs the limited bandwidth of an Institute of Electrical and Electronics Engineers (IEEE) 802.11 network. However, many unresolved issues still exist and part of the problem is the increasing end-user's prospects along with the volatile demands from new higher data rate applications, such as High-Definition television (HDTV), video teleconferencing, multimedia streaming, VoIP, file transfer, and on-line gaming. Hence, a lot of research is being carried out that aims to provide higher data rates, improved security and most importantly for real-time systems, better Quality of Service (QoS).

1.1 Working Towards High-Throughput WLAN

In June 1997, a standard for WLAN connectivity, known as IEEE 802.11 [6], was emerged by IEEE. At the same time, other notable developers of industrial standards had approached alternative solutions, such as HiperLAN [7] by European Telecommunications Standards Institute (ETSI). However, IEEE's proposition was and still remains to be considered as a universal leading standard. The legacy IEEE 802.11 standard specifies the Media Access Control (MAC) sublayer and features various original modulation techniques for the Physical (PHY) layer. Since it was first introduced, numerous changes (referred as amendments) have been applied to the original IEEE 802.11 with the scope to offer capabilities of higher throughputs and QoS support.

Meanwhile, the IEEE 802.11 Working Group was seeking alternative methods to increase data rates because upcoming multimedia and real-time applications begun to require higher throughputs [8]. In July 2002, the IEEE 802.11 standard Working Group (WG) established the High-Throughput Study Group (HTSG) with the aim to achieve promising higher data rate solutions by means of existing PHY and MAC mechanisms [8, 9]. Their first interest was to achieve a MAC data throughput over 100 Mb/s using the IEEE 802.11a standard [10]. However, their objective proved to be infeasible as the estimated throughput bounds well below the theoretical maximum link rate because of the existing PHY and MAC overhead [11]. In September 2003, the HTSG set off the IEEE 802.11n ('n' stands for next-generation) resolution in order to compose an High Throughput (HT) extension of the current WLAN standard that will increase transmission rate and reduce severe overhead. The main goal of IEEE 802.11 Task Group N (TGn) was to define an amendment that would have maximum data throughput of at least 100 Mb/s, as measured at the MAC data Service Access Point (SAP), and at the same time to allow coexistence with legacy devices.

Some of the proposed features are innovative extensions of the IEEE 802.11e [12], an amendment that includes efficient MAC improvements which they can also increase throughput but its objective is to provide QoS. After numerous ballots and excessive delays, in 2007, the consortium presented an advanced HT amendment, known as IEEE 802.11n standard [13]. The specifications offer significant increase in the maximum net data rate from 54 Mb/s to 600 Mb/s[1]. Some of the most popular enhancements introduced by the new standard are Multiple-Input / Multiple-Output (MIMO) and Frame Aggregation.

1.2 Motivations

A major dilemma while researching new proposals is how new ideas can be coalesced with previous and current standards. Moreover, additional factors need to be taken in mind, such as user requirements, service capabilities, physical infrastructure, available bandwidth, financial resources, etc. For example, nowadays, real-time applications such as Voice over IP (VoIP) and video broadcasting have become widely popular but also have strict performance constraints (delay boundaries). However, older MAC schemes seemed adequate to resolve these issues since there were only designed for supporting simple, non-detrimental and insensitive traffic with several flaws, such as wasting channel resources and having difficulties to calculate transmission times [14]. Consequently, new mechanisms that provide acceptable levels of QoS using differentiation and prioritisation had to be defined, resulting into the emerged IEEE 802.11e amendment [12].

For the purpose of QoS within WLANs, the amendment proposes Hybrid Coordination Function (HCF). The function offers two separate methods for channel access, the HCF Controlled Channel Access (HCCA) and the Enhanced Distributed Channel Access (EDCA) for synchronous and asynchronous data transmission, respectively. So, a decentralized type of wireless network (e.g. ad hoc), manages distributed channel access manner with differentiated services by letting the associated applications to set the required level of QoS importance for

[1]acquired only under certain conditions and with all the optional features set active

their offered traffic. These can be defined with specific tags known as User Prioritys (UPs) and are included within the forwarded packet's header elements. Primitively speaking, the higher the UP assigned to a packet, the greater the delay-constraints from the originated application. During the priority selection process, the packets are mapped to separate Access Categorys (ACs) with their own queue buffer and a unique set of parameters that control the average waiting period in their queue buffers. The main point is that higher priority categories are capable of acquiring more bandwidth than the lower priority categories when they are competing against each other and since channel access is "expensive" this can cause starvation to lower ACs. Although this situation can induce unfairness to the lower ACs, this is the most adequate mechanism for the higher ACs to attain channel access within the delay-constraints appointed from the originated application.

As we mentioned earlier, IEEE 802.11n specification document builds upon these probabilistic priority mechanisms along with other MAC enhancements. Nevertheless, the maximal ideal throughput is bounded by a maximum relative MAC data rate that is just over half of the average peak PHY rate. From the bottom layer perspective, this behaviour is mainly caused because of the packet's supplemented overhead, which is the additional required information preceding the transmission of each payload. This deficiency can be tackled or treated with a method known as frame aggregation. There are two main types of aggregation proposed in the IEEE 802.11n standard and both follow the same principle that of: all packets contained in the same transmission buffer and destined to the same receiver can be concatenated within a single frame. However, in the interest to increase the aggregated size, there is a need of packets to be piled in the stack. But, as the waiting period is decreasing, so are the number of packets that trail the first arrived packet, consequently the aggregate size is small. In most cases, frame aggregation adheres due to the EDCA scheduler's priority mechanism, resulting in the network's poor overall performance. There is a trade-off of choice that has to be taken into account when we want to improve network performance, minimizing the waiting time or maximizing utilization, thus efficiency.

In general, real-time video and audio streaming applications are designed to be more persistent to occasional lost packets, thus User Datagram Protocol (UDP) is a more suitable and flexible protocol suite to use. But, in order to support UDP-based real-time applications over the Internet, it is necessary to provide bandwidth to the UDP applications within the network so that their performance will not be seriously affected during periods of congestion. UDP flows do not typically back off when they encounter congestion, but aggressively use up more bandwidth than Transmission Control Protocol (TCP) flows. On the other hand, TCP flows emphasize reliability over reduced latency, thus are extensively utilized by heavy duty applications, such as peer-to-peer (P2P), WWW, File Transfer Protocol (FTP), etc. Nevertheless, the TCP protocol imposes many issues in conjunction with the properties of the wireless medium [15, 16, 17] and IEEE 802.11e's probabilistic prioritization mechanism [18].

Our aim in this thesis is to address these key challenges and act accordingly.

1.3 Major Contributions

1.3.1 Mitigating Overhead Further for Very High-Speed WLANs

The new IEEE 802.11n standard provides enough capacity to service immense offered loads. Nevertheless, the PHY enhancements are not sufficient to guarantee significant throughput performance. The principle of Frame Aggregation is to form larger frames for transmission by collecting multiple packets inside an aggregate buffer [1, 4]. Currently, when a frame arrives in the transmission queue, the Carrier Sense Multiple Access with Collision avoidance (CSMA/CA) scheduling mechanism directs the station (STA) when to access the wireless medium. There are four (4) ACs defined in EDCA where they have different priorities in order to differentiate services for separate applications. Therefore, a higher priority flow acquires more bandwidth than a lower priority one but the former tends to have smaller aggregate sizes due to shorter waiting time. To understand the impact of this behaviour, we first develop an analytical model for the aggregate buffer queue. Mathematical analysis show that for networks with small load rates, we need to intentionally defer the channel access procedure in order to introduce additional packets in the aggregate buffer. We therefore adapt the basic characteristics of Delayed Channel Access (DCA) [19] and extend it further in order to provide support for both UDP and TCP protocols. Results also suggest that DCA based algorithms are a promising MAC technique for very high-speed WLANs.

1.3.2 Restoring Fairness in QoS Networks

The QoS support in EDCA is provided by the introduction of prioritization via distinguishing the traffic flows into ACs. Consequently, there are distinct sets of contending entities with relative priority in medium access per AC. The main idea is to use four (4) coupled CSMA/CA queue mechanisms one for each AC that behaves as a single enhanced Distributed Coordination Function (DCF) contending entity, and all to contend simultaneously to access the same wireless medium. However, each AC is parametrised with different set of values, so higher priority traffic has certain parameters to allow it to gain access to the channel earlier and more often than the lower priority traffic. We use Model Analysis to describe the magnitude of impact that high ACs have over lower ACs. Nevertheless, this unfair behaviour is apparent and important for applications to meet the QoS requirements. But with emerging technologies, high priority applications have become more demanding and acquire further channel resources. Through extensive simulation runs, we demonstrate that DCA and its extensions can provide great fairness over lower ACs by deferring the transmission for all flows, including high ACs. Results show that for all contenting entities great improvement over the channel utilization and the total throughput while still obeying all QoS requirements.

1.3.3 Buffer Sizing for TCP Flows

During the DCA research, we observe that there can be a close interference between DCA, and with the MAC layer in general, and the queue size assigned to the TCP Window buffer. Surprisingly, this buffering issue has received little attention in the 802.11 literature, probably because it is only recently that high-throughput wireless networks have become main research subjects for the industry and the academia. The classical rule of thumb is to provision buffers to be equal to the Bandwidth-Delay Product (BDP), which is defined as the bandwidth of the link multiplied by the average delay of the flows utilising this link [20]. However, Operating Systems (OSs) and firmware specifications set as maximum TCP Window to much lower values and the unstable conditions that take place in a WLAN, such as channel contention, link quality and random nature of the channel access scheduling operation, makes it difficult to determine the size of a TCP Window at a certain point. So, we first consider an adapting sizing algorithm that is based on certain measurements of the current and previous packet traffic, and feedback from DCA's triggering mechanism. Our second approach is to classify flows based on duration, number and size of packets per flow, and inter-packet arrival time. In addition, with the aid of a cognitive agent, the proposed extension will be able to determine the type of transportation protocol that these flows use. We design and implement two distinct enhancements for DCA, known as Adaptive DCA and Selective DCA. Each has a different operational approach but the end objective is the same. The effectiveness of these algorithms is demonstrated via extensive simulations and experimental measurements.

1.4 Thesis Outline

This thesis is organised as follows. In Chapter 2, we describe and evaluate the proposed MAC enhancement of the Frame Aggregation method but firstly we indicate the throughput limitations of the archetype IEEE 802.11 standard. In Chapter 3, we portray Frame Aggregation as an $M/G^{[a,b]}/1/K$ queueing model, then we review the model definition and present numerical results for various classes of service processes, different service starting or batch collection rules under various load conditions. In Chapter 4, we discuss and demonstrate the luck of performance improvement of IEEE 802.11n in conjunction with QoS prioritization mechanisms. In Chapter 5, we review the motivation, design and effectiveness of DCA, Also, we investigate and explain the negative behaviour of DCA with TCP traffic flows. In Chapter 6 and Chapter 7, we describe and evaluate two distinct enhancements for DCA, known as Adaptive DCA and Selective DCA, respectively. The thesis concludes in Chapter 8 and discusses some ideas for future work.

Chapter 2

IEEE 802.11n

Some of TGn's initial draft proposals, eventually led to today's standard [13], introduced as main techniques for the PHY the utilization of MIMO antennas with Orthogonal Frequency Division Multiplexing (OFDM), plus various channel binding schemes, and for the MAC the use of frame aggregation with multiple protection schemes, designed to allow coexistence of 'n' with "legacy" devices. Within this chapter, we describe and evaluate the proposed MAC enhancement of the Frame Aggregation method but firstly we indicate the throughput limitations of the archetype IEEE 802.11 standard.

2.1 Throughput Limits of IEEE 802.11

To understand the inefficiency of 802.11 over higher data rates, we must briefly describe the legacy DCF. The MAC architecture is based on the logical coordination functions that determine who and when to access the wireless medium at any time. It supports fragmentation and encryption and acts as an interface between the Logical Link Control (LLC) sub-layer and the PHY layer. In the legacy 802.11 standard, there are two types of access schemes: the mandatory DCF, which is based on the CSMA/CA mechanism; and the optional Point Coordination Function (PCF), which is based on a poll-and-response mechanism. The former method, the one that interest us, operates with a First-In-First-Out (FIFO) transmission queue that is in situ for receiving and buffering incoming data from the higher layers. The basic operation of DCF is illustrated in Figure 2.1.

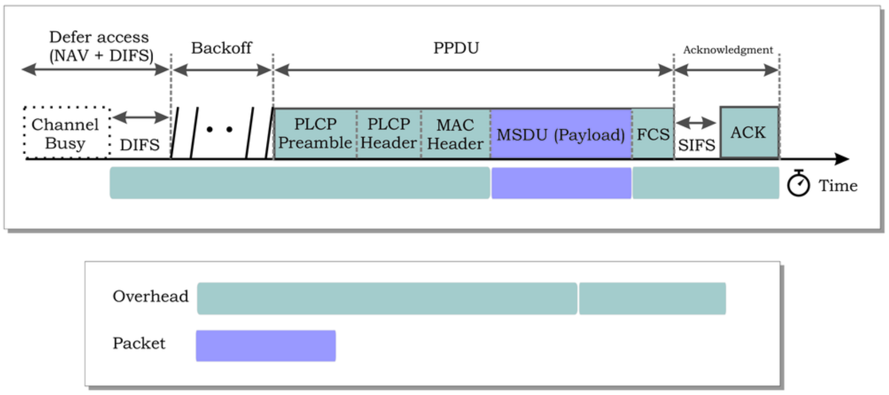


Figure 2.1: The DCF basic operation

Following a frame, also known as a MAC Service Data Unit (MSDU), arriving from the LLC at the head of the transmission queue, the DCF operation instructs the MAC to wait for a global defined interframe interval called DCF Inter-Frame Space (DIFS) before any other actions can be taken. In addition within the QoS amendment there is a set of intervals known as Arbitrary Inter-Frame Space (AIFS). If the PHY reports back to the MAC that the wireless channel is busy, the STA's MAC halts until the medium becomes free. On the other hand, if the medium remains idle during DIFS deference, the STA enters a back-off procedure where a slot is selected from a random back-off counter within a Contention Window (CW). Next, the counter starts a decrement process while the channel remains idle for each slot interval. When the counter reaches zero, the STA obtains an affirmation to send the information through the wireless link.

Now, each WLAN point that receives a data frame, utilizes an error checking processes to detect any presence of received errors. If no errors are found, it sends back an Acknowledgement (ACK) frame after a specified Short Inter-Frame Space (SIFS) to verify that the information was successfully received. If the sending STA does not receive an ACK after SIFS, it will assume that the communication was broken or interfered, and it will start a new DCF process for retransmission. Should there be a case of collision, then the MAC extends its CW, selects a new slot, and repeats the previous steps. Finally, there is an optional mechanism known as Request To Send (RTS)/Clear To Send (CTS) that intends to resolve the so-called hidden and exposed node scenarios that usually occur in ad-hoc networks. With RTS/CTS, after a STA is granted access to transmit, it first sends an RTS frame and then holds back for the CTS response from the receiver. Obviously, this situation can be disadvantageous if the actual data frame size is small because the RTS/CTS exchange produces further overhead and consequently reduces the effective throughput. For reasons of simplicity, the RTS/CTS method is omitted on the following examples.

From Figure 2.1, we can clearly comprehend the consequences of that hefty overhead on the system throughput. The figure exemplifies the required procedure that each single packet traverses from the time it arrives at the MAC until it is successfully received by the receiver, with different headers and tails added over different sub-layers on the original payload. Note that [6] states that all Physical Layer Convergence Protocol (PLCP) preambles and PLCP headers shall be transmitted using the basic link rate, which is much less than the rate used for data transmission and most of the time the minimal rate. A complete transmission cycle of a simple DCF consists of DIFS deferral, back off, data transmission, SIFS deferral, ACK transmission, and propagation delay, so to transmit a data packet, a large overhead is accumulated. Also, we assume that the transmission would be successful with the first attempt and no re-transmissions are needed, something that would exponentially affect the existing overhead.

Be aware that the overhead shown does not correspond to real time lengths, as payload varies, but it shows the additional time that is required to have a successful transmission. So, the higher the packet rate meaning the number of packets that are injected to the MAC per second the higher the relative overhead the system introduces. This assumption can be easily demonstrated through a simple numerical analysis. The calculation methodology follows the recommendations given in [11, 21, 22] for an IEEE 802.11a OFDM network since the TGn's proposal follows the same modulation but on higher rates. For the sake of completeness, we present the four-step methodology that derives the Theoretical Throughput Limit (TTL), also known as Throughput Upper Limit (TUL).

For simplification, the TTL is calculated from constant-size data frames transmitted over a single unidirectional data path when the system is in the best case scenario. In a best case scenario, we assume that at any transmission cycle the channel is idle with negligible Bit-Error-Rate (BER), there are no losses from collisions or packet overflow over the receiver's side, and that there is only one STA that continually transmits frames which accordingly are received successfully and acknowledged by the receiver. Also, we do not consider fragmentation, RTS/CTS control and management frame overhead, such as beacons and association frames.

The TTL calculation methodology is based on four sequential steps:

1. Calculate $TxTime$ – the time in microseconds to transmit one data frame, including frame preamble, frame header, and Radio Frequency (RF) modulation parameter fields.

   $$TxTime = T_{preamble} + T_{signal} + T_{symbol} * \left\lceil \frac{L_{pad} + (L_{payload} * 8) + L_{tail}}{N_{dbps}} \right\rceil$$

   where $T_{preamble}$, T_{signal}, T_{symbol}, L_{pad}, L_{packet}, L_{tail} and N_{dbps}, represent the time (in μs) to transmit a PLCP preamble, a signal field, a service field (define symbol clock and code), the length of the PLCP Protocol Data Unit (PPDU) pad bit field, the packet's size in octets, the PPDU tail bit field and the number of data bits per symbol, respectively.

2. Calculate $FS\text{-}to\text{-}FS\ Interval$ – the time in microseconds to transmit one data frame, with acknowledgement (T_{ACK}), including SIFS, DIFS and backoff time. For backoff time, the average backoff, $\bar{BO}$, in the best case is used, where the medium is available at first attempt and the number of backoff slots is selected from $[1 \ldots CW_{min}]$.

   $$FS\text{-}to\text{-}FS\ Interval = TxTime + SIFS + T_{ACK} + DIFS + \bar{BO}$$

   where $\bar{BO} = (CW_{min}/2) * Slot\ Time.$

3. Calculate $Frame\,Rate$ – the number of frames per second that can be transmitted across the air interface.

$$\textit{Frame Rate} = \textit{FS-to-FS Interval}^{-1} * 10^6$$

4. Derive TTL in Mb/s – this value represents an upper boundary on 802.11 network performance at the MSDU level.

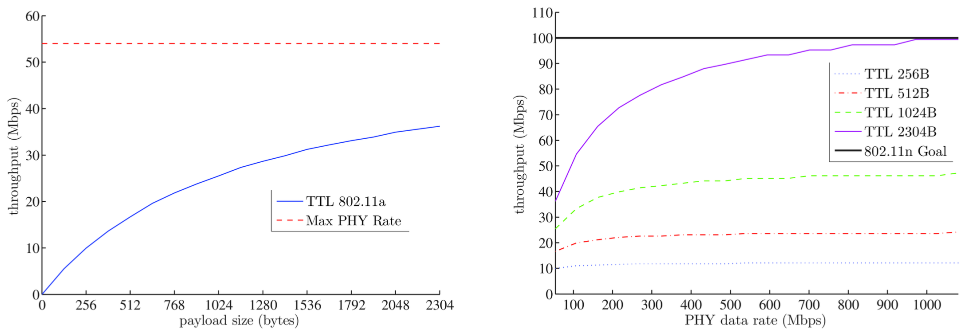
$$TTL = \textit{Frame Rate} * (L_{payload} * 8) * 10^{-6}$$

The default timing parameters of the IEEE 802.11a necessary for the calculation of the TTL are listed in Table 2.1. Note that since the maximum bandwidth that the 802.11a can support is bounded to 54 Mb/s, in order to test higher PHY rates, we use a scaling factor over both basic and control rates. Figure 2.2 shows the impact of the PHY and MAC overheads have over the maximum ideal throughput that the network provides over various payload's sizes or link rates.

Parameter	Value	Parameter	Value
$T_{preamble}$	16 μs	L_{pad}	16 bits
T_{signal}	4 μs	$L_{payload}$	variable
T_{symbol}	4 μs	L_{tail}	6 bits
T_{ACK} @ 24 Mb/s	28 μs	N_{dbps} @ 54 Mb/s	216 dbps
$SIFS$	16 μs	$DIFS$	32 μs
CW_{min}	15	$Slot\ Time$	9 μs

Table 2.1: Timing Parameters of IEEE 802.11a

Figure 2.2a shows the TTL for various payload sizes, starting from 0 bytes up to the maximal permitted MSDU size of 2, 304 bytes. When the payload size takes its highest value, the achieved MAC Throughput is bounded at around 36.12 Mb/s, well below the offered raw PHY data rate of 54 Mb/s. Relatively, over the same link data rate the throughput performance degrades further as the packet length reduces. Previous research has shown that on average, Internet flows embody MSDU packets with sizes less than 200 bytes. Therefore, it is essential to introduce a method the wireless medium can be utilized efficiently by using larger frame sizes. Figure 2.2b illustrates the TTL graphs for four different payload sizes: 256, 512, 1,024 and 2,304 bytes, while the PHY data rate is increased to extremely high values. From the shape of the curves we can deduce that the graphs for 256, 512 and 1,024 bytes of MSDU length sizes are bounded to below 11 Mb/s, 23 Mb/s and 46 Mb/s, respectively, with no potentials to increase further. Conjointly, when we circulate larger MSDUs, the maximum TTL that is actualized at the fringe of our simulation is 99 Mb/s, just below TGn's main goal of 100 Mb/s but then again a raw data rate to the region of 1,080 Mb/s is unrealistic in today's wireless networks.



(a) Throughput vs. Payload Size (b) Throughput vs. PHY Date Rate

Figure 2.2: Theoretical Throughput Limits for IEEE 802.11a

In conclusion, the above analysis has shown that the maximum ideal MAC throughput is bound well below the offered peak raw PHY rate, even if that increases to infinity. The reason behind this limitation is the excessive overhead that is needed when a single packet is transmitted and since this information can not be omitted, it is essential to adopt alternative innovative techniques in order to reach a target of higher than 100 Mb/s for the MAC throughput. An effective resolution is to reduce the frequency of the overhead by frame concatenation or aggregation, in other words join multiple packets together and transmit them as a single data frame. Various methods of aggregation have been proposed, but all of them follow a similar logic [13, 23, 24, 25, 26, 27, 28]. The are two main types of aggregation established by TGn and these are described in detail in the following section.

2.2 Frame Aggregation Mechanisms

The legacy IEEE 802.11 WLAN efficiency is severely compromised as the data rate increases since the throughput is increasingly dominated by the overheads for high data rates, as shown above. Therefore, reducing MAC overheads and pursuing higher data rates are both necessary for designing and implementing HT WLANs. Frame aggregation is one of the various MAC enhancements that can maximize goodput and at the same time increase efficiency. Data aggregation was first introduced in the QoS amendment of the standard, IEEE 802.11e [12], and was carried out through a process known as Controlled Frame-Bursting (CFB) [29, 30, 31, 32, 33, 34, 35, 36, 37]. A station gaining the channel transmits the frames available in its buffer successively provided that the duration of transmission does not exceed a certain threshold, referred to as the Transmission Opportunity (TXOP) limit. Each frame is acknowledged by an ACK after a SIFS interval. The next frame is then transmitted immediately upon receiving this ACK. If the transmission of any frame fails the burst is terminated and the station contends again for the channel to retransmit the failed frame. Such schemes benefit from amortizing the control overhead over multiple data packets.

Frame aggregation for a HT is specified in [13] and is one of the various MAC enhancements that maximizes goodput and increases efficiency. There are two main ways to perform frame aggregation, known as Aggregated MAC Service Data Unit (A-MSDU) and Aggregated MAC Protocol Data Unit (A-MPDU). The main distinction between MSDU and MAC Protocol Data Unit (MPDU) is that the former corresponds to the information that is imported to or exported from the upper part of the MAC sub-layer, from or to the higher layers, respectively, while the later relates to the information that is exchanged from or to the physical link by MAC's lower part; assuming that we are referencing to the Open Systems Interconnection (OSI) Model [38]. Aggregate exchange sequences are made possible with a protocol that acknowledges multiple MPDUs with a single Block Acknowledgement (BlockAck) in response to a Block Acknowledgement Request (BAR) [39, 40].

2.2.1 Aggregated MAC Service Data Unit

The principle of the A-MSDU is to allow multiple MSDUs being sent to the same receiver, concatenated in a single MPDU. This definitely improves the efficiency of the MAC layer, specifically on small packets over congested networks, which is the most persistent and prevalent case [41, 42, 43]. This supporting function for A-MSDU within the IEEE 802.11n is mandatory at the receiver. However, the transmitter is free to choose the use of an A-MSDU on the Traffic Specification (TSPEC). In order for an A-MSDU to be formed, a layer at the top of the MAC receives and buffers multiple packets (MSDUs). The A-MSDU is completed either when the size of the waiting packets reaches the maximal A-MSDU threshold or the maximal delay of the oldest packet reaches a pre-assigned value. Its maximum length can be either $3,839$ or $7,935$ bytes, this is 256 bytes shorter than the maximum PHY PHY Service Data Unit (PSDU) length ($4,095$ or $8,191$ bytes respectively), as predicted space is allocated for future status or control information. The size can be found in the HT Capabilities Element that is advertised from an HT STA in order to declare its HT status. The maximal delay can be set for 1 μs or an independent value for every AC.

There are also certain constraints when constructing an A-MSDU: *a*) all MSDUs must have the same Traffic Identifier (TID) value, *b*) the A-MSDU's lifetime should be corresponding to the maximum lifetime of its constituent elements, and *c*) the Destination Address (DA) and Sender Address (SA) parameter values in the subframe header must match to the same Receiver Address (RA) and Transmitter Address (TA) in the MAC header. Thus, broadcasting or multicasting is not allowed.

Figure 2.3 describes a simple structure of a carrier MPDU which contains an A-MSDU. Each subframe consists of a subframe header followed by the packet arrived from the LLC and $[0\ldots3]$ bytes of padding. The padding size depends on the rule that each subframe, except for the last one, should be a multiple of four bytes, so the end-receiver can approximate the beginning of the next subframe. A major drawback of using A-MSDU is under error-prone channels. By compressing all MSDUs into a single MPDU with a single sequence number, for any subframes that are corrupted, the entire A-MSDU will have to be retrans-

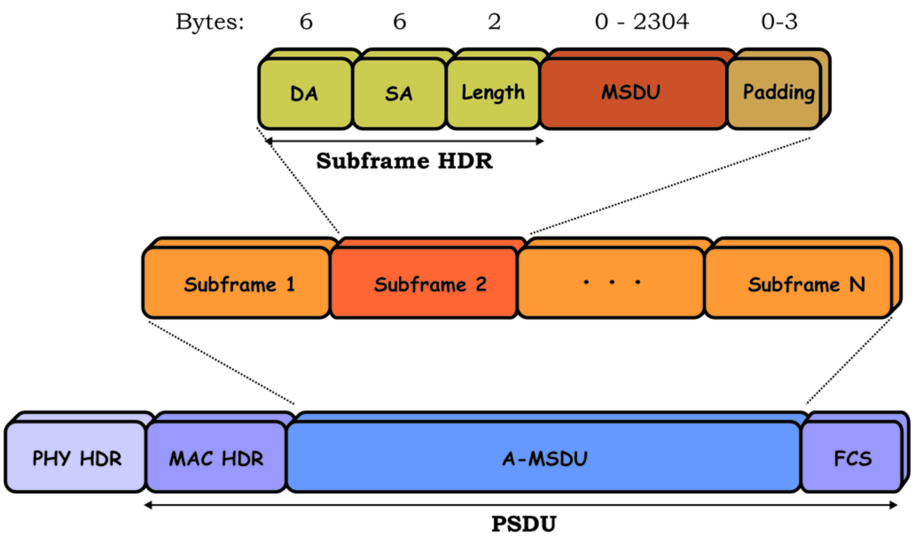


Figure 2.3: The A-MSDU structure

mitted. This situation could easily lead to poor utilization of the channel in case of transmission errors and has been addressed at [35, 26] where additional frame structures or optimum frame sizes have been proposed. This will definitely improve performance under noisy channels.

2.2.2 Aggregated MAC Protocol Data Unit

The concept of A-MPDU aggregation is to join multiple MPDU subframes with a single leading PHY header. A key difference from A-MSDU aggregation is that A-MPDU functions after the MAC header encapsulation process. Consequently, A-MSDU's restriction of aggregating frames with matching TIDs is not a factor with A-MPDUs. However, all the MPDUs within an A-MPDU must be addressed to the same RA. Also, there is no waiting/holding time to form an A-MPDU so the amount of MPDUs to be aggregated is totally dependant on the number of packets already in the transmission queue. The maximum length that an A-MPDU can obtain, in other words the maximum length of the PSDU that may be received, is $65,535$ bytes but it can be further constrained according to the STA's capabilities found in the HT Capabilities element. The utmost number of subframes that can be held is 64 since a BlockAck bitmap field is 128 bytes in length where each frame is mapped using two bytes. Note that these two bytes are required to acknowledge up to 16 fragments but since A-MPDU does not allow fragmentation these extra bits are excessive. As a result, a new variant has been implemented, known as Compressed Block ACK [44] with bitmap field of eight bytes long. Finally, each subframe's size is limited to $4,095$ bytes as the length of a PPDU can not exceed 5.46 ms time limit; this can be derived from the maximum length divided by the lowest PHY rate which is 6 Mb/s and is the highest duration of an MPDU in IEEE 802.11a.

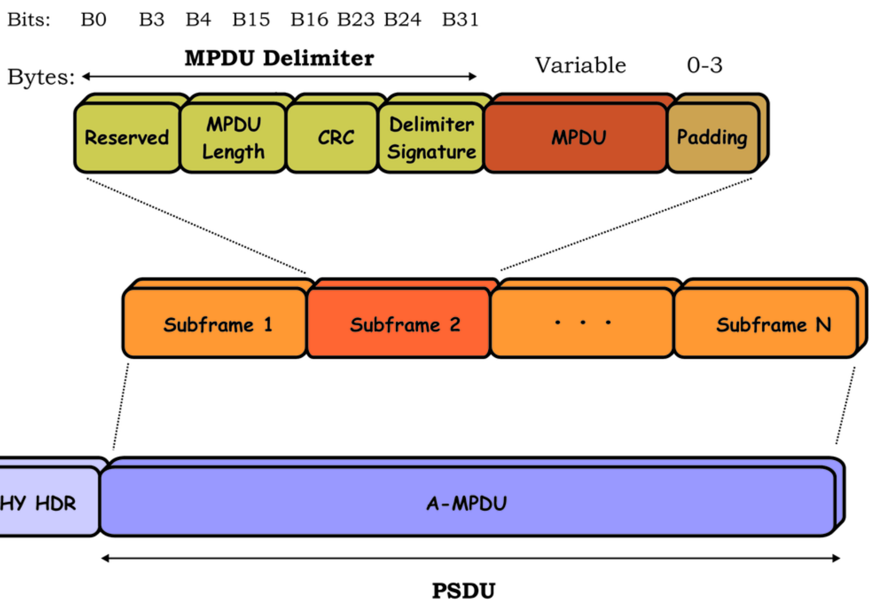


Figure 2.4: The A-MPDU structure

A basic illustration of the A-MPDU structure can be seen in Figure 2.4. A set of fields, known as delimiter are inserted before each MPDU and padding bits varied from $[0 \ldots 3]$ are positioned afterwards, at the tail of the frame. The basic operation of the delimiter header is to define the MPDU's position and length inside the aggregated frame. It is noted that the Cyclic Redundancy Check (CRC) field in the delimiter verifies the authenticity of the 16 preceding bits. The padding bytes are added such that each MPDU is a multiple of four bytes in length, which can assist subframe delineation at the receiver's side. In other words, the MPDU delimiters and padding bytes determine the structure of the

A-MPDU. Now, once the A-MPDU is received a de-aggregation process initiates. First it checks the MPDU delimiter for any errors based on the CRC's value. If it is correct, the MPDU is extracted and it continues with the next subframe till it reaches the end of the PSDU. Otherwise, it checks every four bytes until it locates a valid delimiter or the end of the PSDU. Delimiter's signature has a unique pattern in order to assist the de-aggregation process while scanning for delimiters.

2.2.3 Two-Level Aggregation

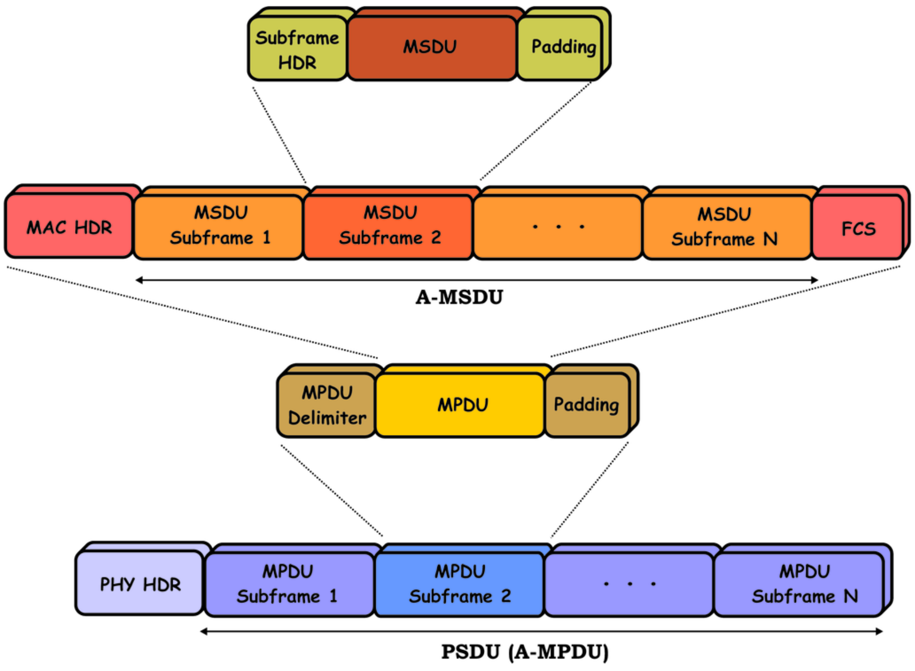


Figure 2.5: The two-level aggregation method

A two-level frame aggregation comprises a blend of A-MSDU and A-MPDU over two stages. In Figure 2.5 we illustrate how this new scheme can be achieved. The basic operation is explained as follows. In the first stage, if any MSDUs that are buffered in the A-MSDU provisional storage area have the same TID and DA then these data units can be compacted into a single A-MSDU. If the TIDs are different, all these aberrant frames can move over the second stage where they will be packed together with any A-MSDUs derived from the first stage or other single MSDUs by using A-MPDU aggregation. However, it has to be mentioned that given that for an A-MPDU data frame the maximum MPDU length is limited to $4,095$ bytes, then any A-MSDUs or MSDUs with lengths larger than this threshold cannot be transmitted. Conjointly, any fragments from an A-MSDU or MSDUs cannot be included in an A-MPDU either. In the next section we are going to evaluate how this synthesis is more efficient in most of the cases than A-MPDU and A-MSDU aggregation operating alone.

2.3 High-Throughput Model in OPNET

A gradually more established technique for network performance analysis is Simulation Modelling. Researchers need to demonstrate and verify that their study is accurate. So, they form simulations through specified models and compare the output figures with their hypothesis. There are two types of analysis that they can perform: analytical modelling [45, 46] and computer simulation (or computational model) [47, 48]. Analytical modelling comprises of mathematical analysis, in other words representing a network design as a set of equations, variables and functions. The main drawback is the over simplistic view of the whole distributed system and the lack of ability to simulate the dynamic nature of a network. The investigation of a complex structure needs a discrete event simulation application, which can process the time that would be associated with real events in a real-life situation. Sometimes the set of equations that need to be solved may be Non-deterministic Polynomial-time (NP)-complete, certain classes of problems are not solvable in realistic time therefore approximations are usually taken. The computer simulation option uses a computer program, a valuable study tool over today's networks with complex architectures and topologies, this attempts to simulate an abstract model of a particular system that is too complicated for analytical solutions [49] [50]. Simulation allows the evaluation of network protocols under varying network conditions. Studying protocols, both individually and as they interact with other protocols, under a wide range of conditions is critical to explore and understand the behaviour and characteristics of these protocols.

Test-beds and laboratory experiments are also important approaches to network research. Since they use real code, experiments run in test-beds or labs automatically capture important details that might be missed in a simulation. This approach also has drawbacks as these are expensive to build, can be difficult to reconfigure and share, and have limited flexibility. Note that whatever approach a researcher follows, for the purpose of a general reassessment he may also want to compare the derived results of his study with an alternative method. However, the outputs and results may prove to be dissimilar and not correlative. A comparison study of two popular simulators with a live network test-bed showed that simulators may not always model the behaviour of a real network adequately [51].

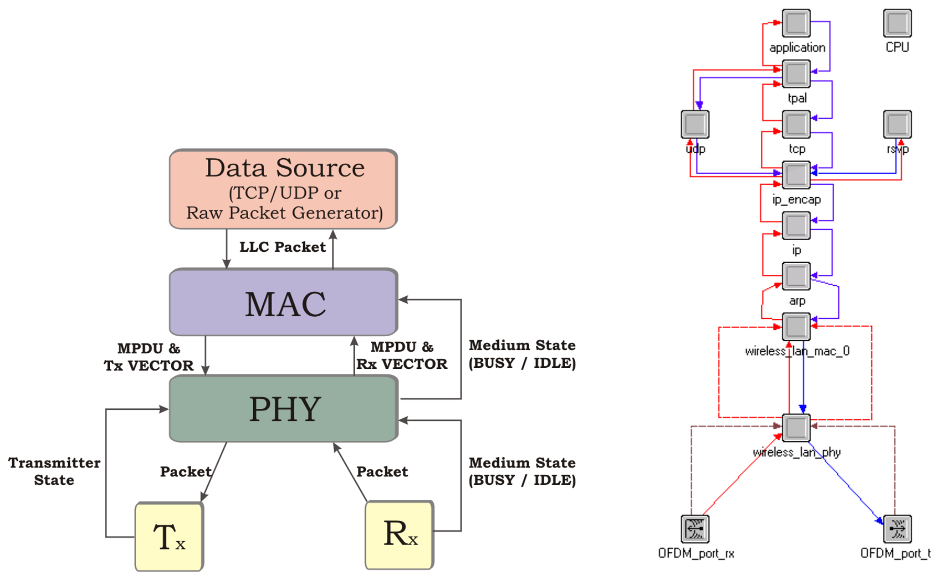
For the evaluation of our research, the majority of the performance assessments will be conducted with the use of computer simulations using a network tool known as Optimized Network Engineering Tool (OPNET) Modeler [52]. OPNET Modeler is a leading environment for network modelling and simulation, allowing us to design and study communication networks, devices, protocols, and applications with unmatched flexibility and scalability. The implementation of network models using OPNET has been widely used [53, 54, 55, 56, 57, 58] and consequently presumes an effective approach. During the IEEE 802.11n standardization process, where the members of the TGn had to narrow numerous propositions before all consent to a final HT standard, a simulation model was developed in order to generate results for evaluation and testing purposes [59, 60]. A prototype IEEE 802.11n OPNET model was designed and implemented by Dmitry Akhmetov and Sergey Shtin of Intel Corporation [61] to fit the purpose. For the simulations of this research, an extended version of this aforementioned model was used in order to produce the majority of the results. Surely, the model had to be altered in order to determine and implement the behaviour of the new proposed enhancements but even so credit is given to the initial developers.

This section, briefly describes the model's design characteristics and explains some of the processes undergone with OPNET's Modeler. It is unnecessary to go through each process thoroughly as it contains thousands of instruction lines. However, it is important to understand how the model is designed as portions of its structure will be altered to implement our proposed enhancements.

2.3.1 PHY & MAC Interaction and Interfaces

The HT model was actually derived from the existing public OPNET model of IEEE 802.11b. Basically, in OPNET the model, or node as known in network modelling, characterizes the whole behaviour of a network object. This behaviour can be defined over one or multiple modules where their underlying functionality is consented with the process entities. The latter are represented by the Finite State Machines (FSMs) and the operations performed in each state are described in code blocks implemented in *Proto-C*, *embedded C* or *C++*. So, everything can be independently separated from the whole model and if any alterations are required, these can be done without difficulty or meddling. Usually each module represents a separate layer of the OSI model but the original 802.11b node has its MAC and PHY FSMs assembled into a single one. Furthermore, the model's T_x and R_x blocks cannot support MIMO functionality, multiple streams for transmission or reception of packets, as each block has only a single stream for the standard Single-Input / Single-Output (SISO) operation. For that reason, even though the new model is based on the existing, quite a few considerations had to be taken into account during implementation.

An overview of OPNET's HT proposed model [62] can be seen in Figure 2.6. Now, the model includes two separate MAC & PHY process layers with each one performing separate functions, while OSI's higher layers operation remain unchanged (see Figure 2.6b). New underlying communication links between these two processes were established for exchanging information. These links are bi-directional and may include: pushing MPDU packet streams from one layer to another, statistic information about the T_x and R_x states and special structures (in Figure 2.6a are labelled as T_x VECTOR and R_x VECTOR) which contain information about the length in bits of packets and a number of packets that will be transmitted or contain training information and reception rate, respectively. In OPNET, the latter has been implemented through a special data type called Interface Control Information (ICI) and contain fields for user-defined parameters to be shared by multiple entities in the network and consecutively these are referenced in calls to Kernel Procedures from within process models [63]. The transmission of MPDUs from MAC to PHY and from PHY to MAC is done by a



(a) Black-Box overview of the model

(b) Model's structure in OP-NET

Figure 2.6: OPNET node model for IEEE 802.11n

set of communication data streams for every data rate (up to four T_x/R_x blocks). From the PHY to MAC process model there are two statistic wires to inform the latter the Clear Channel Assessment (CCA) status (BUSY/IDLE). The PHY process model encapsulates preamble and PLCP header transmission/reception functionality and is also responsible for channel's state analysis. The PHY is connected to T_x and R_x blocks by two streams for transmission and reception of packets and statistic wires are connected to deliver the status of the transmitter and receiver.

2.3.2 MAC Process Model Design

The existing OPNET 802.11b model provides the most basic MAC layer functionalities of the IEEE 802.11 standard, except some management frame generation. So, the model of a IEEE 802.11n compliance MAC process will be built on a base of an existing model, with some modification and function additions in order to fulfil the new standard's specifications and requirements [5]. Note that all this new features will not disrupt the functionality of the existing 802.11 MAC model, therefore it can still be used for the legacy devices. A list of the most common existing model's characteristics and the most important newly added assets can be viewed in Table 2.2.

Standard Model	HT Model
DCF & PCF	QoS features (EDCA & HCCA)
DATA+ACK sequences	Frame Aggregation
RTS/CTS+DATA/ACK sequences	Block ACK sequences
(De)Fragmentation	New reassembly and reordering rules
Duplicate filtering	New queuing mechanism
AP operation	Enhanced AP functionality
STA operation	Reverse (bi-directional) data flow
etc.	HT frame management support
	etc.

Table 2.2: Existing IEEE 802.11 and HT model features

The behaviour of the new MAC layer can be displayed with the MAC's State Transition Diagram (STD) (Figure 2.7) taken out from OPNET's design interface. At first, an important modification for QoS support [64] as defined in IEEE 802.11e [12] was important to implement. The original MAC process model maintains one transmission queue for each destination address. However, in order to provide EDCA of four ACs, four T_x queues for each DA are required. Also, the queuing process has been slightly changed to the extended model, as every fragment of a fragmented MSDU is stored as a separate frame to allow easy aggregate content forming. So far, the legacy model performs fragmentation just before the transmission since it only has to send one frame per transmission. Consequently, packets arriving in the queuing buffer are not undergoing the fragmentation process until the sequencer extracts a single MSDU from queue, performs fragmentation and places fragments into the fragment buffer ready to be transmitted. But, in order to operate aggregation this operation needed to get advanced so the MSDUs can be fragmented before queuing. Besides these major additions to the model, additional enhancements where introduced in order to express exactly the specifications of the new standard, such as Power Save Multi-Poll (PSMP) operation, Reverse Direction (RD) communication links, and protection mechanisms supporting coexistence with non-HT STAs, features that we don't elaborate on further as they are unconnected with our research.

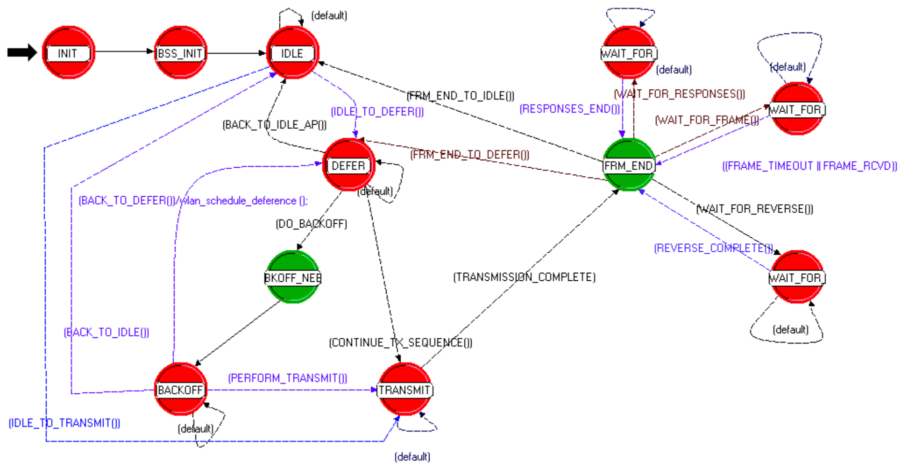


Figure 2.7: The STD for the MAC process

The transmission process, an exemplification is given in Figure 2.8, initiates an action when a “Send Packet” interrupt is received. Then, the transmission sequencer agent extracts a number of fragments (e.g. MPDUs) from the queue into the transmission buffer queue. Once the aggregate is formed, it transmits the bulk as a whole within a single burst. It is important to understand that the decisions for the scheduling and the choice of the MPDUs to transmit is clearly taken by the transmission sequencer block, which is responsible for selecting and aggregating the data from the queues, and also from the processor’s interrupts because scheduling heuristic is based on time events such as maximum delays, transmission opportunities, beacon intervals and others.

As part of the evaluation and quality check of the proposed MAC process model, a test specification document was created. The semantics of that document is the basis for a notion of an exhaustive test set. So, if the results of some hypothesis on the model under test are equivalent to the results in the specification text, then that test set can be assumed to be a success [65]. The execution of a simulation under evaluation on a finite subset of its input domain and the interpretation of the obtained results is known as dynamic test [66]. So, once all new behaviour was implemented on a new HT model, the initial step of

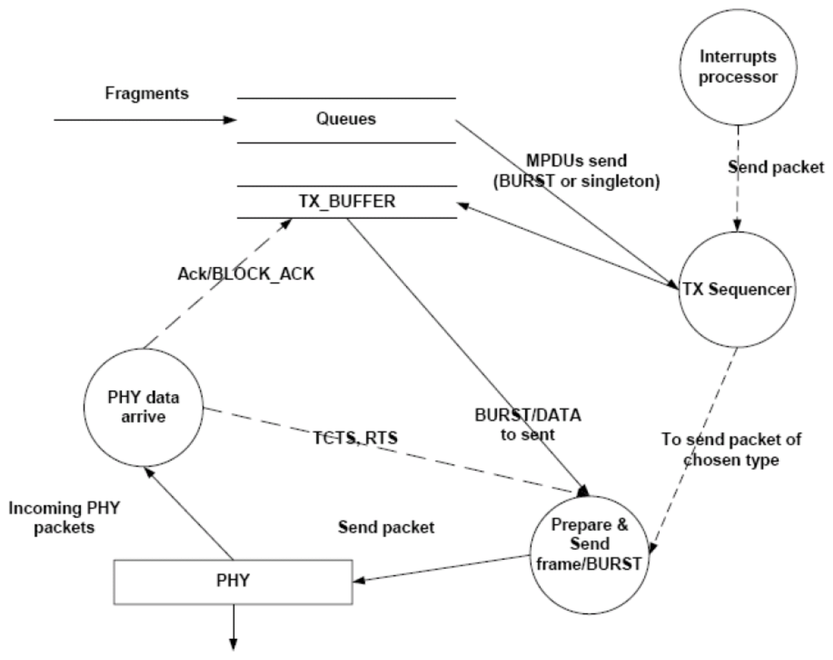


Figure 2.8: Transmission process flow [5]

the testing process sequence was to gather statistics and compare them with the non-modified model. The creation of the hypothesis of the scenarios, also know as usage cases, were based on a finite set of all possible states that the model would be in a real environment, e.g. testing of the "RTS/CTS+DATA+ACK" transmission, simple "DATA+ACK" transmission, retransmissions, cases of missing ACK or CTS, partial MSDU removal, along with others. The developers were able to simplify the quality check by isolating the MAC process from all other layers and by creating dumb processes that will only perform specific operations, e.g the PHY process functioning as a CCA sensor. The large number of exhaustive tests that were carried out during implementation and the fact that this exact model was utilized during the IEEE 802.11n standardization, may assure us that the model surely provide accurate results and no further evaluation will be needed.

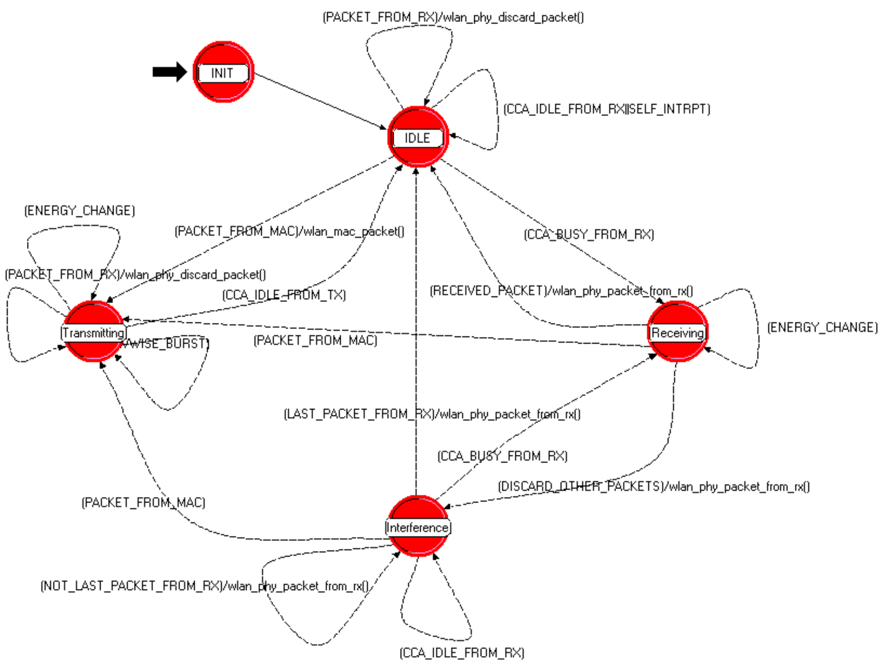

(PACKET_FROM_RX)/wlan_phy_discard_packet()
INIT
IDLE
(CCA_IDLE_FROM_RX||SELF_INTRPT)
(ENERGY_CHANGE)
(PACKET_FROM_MAC)/wlan_mac_packet()
(CCA_BUSY_FROM_RX)
(PACKET_FROM_RX)/wlan_phy_discard_packet()
(CCA_IDLE_FROM_TX)
(RECEIVED_PACKET)/wlan_phy_packet_from_rx()
Transmitting
/WISE_BURST)
(PACKET_FROM_MAC)
Receiving
(ENERGY_CHANGE)
(LAST_PACKET_FROM_RX)/wlan_phy_packet_from_rx()
(CCA_BUSY_FROM_RX)
(DISCARD_OTHER_PACKETS)/wlan_phy_packet_from_rx()
(PACKET_FROM_MAC)
Interference
(NOT_LAST_PACKET_FROM_RX)/wlan_phy_packet_from_rx()
(CCA_IDLE_FROM_RX)

Figure 2.9: The STD for the PHY process

2.3.3 PHY Process Model Design

The PHY process has been separated from the joint MAC-PHY process supplied by the existing OPNET model and furthermore it models the behaviour of fast link adaptation over OFDM-MIMO transmissions. For a detailed structure about the model, information can be found in [67] but generally speaking, it is a simple interface between the T_x/R_x pipeline stages with multiple options for different transmission and rate modes. It supports the new frame structures (A-MSDU and MPDU) and allows the use of short guard intervals. Figure 2.9 shows the STD of this extended PHY as implemented in OPNET. A summary of the main functions that can support are: converting MPDU(s) data to PPDU format and vice versa, transferring or receiving PPDU(s) data to T_x or from R_x ports, respectively, providing CCA reports from the R_x port (e.g. the wireless medium is busy or idle), checking if the packets can be decoded according to the channel's BER or received power and providing access to external models that describe the performance of the PHY modulation, coding schemes and channel models. For the latter, OPNET allows us to define external models that can function along with the process's operation, which is an advantage for object oriented platforms. So, many of the complex functions that calculate the performance of the PHY modulation and coding schemes have been implemented outside OPNET's platform.

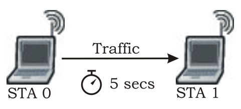
2.4 Performance Evaluation

In this section we will perform various simulations in order to assess the performance of 802.11n compliance standard. The TGn has predefined some specific usage models [68] based on various market-based use cases that intend to support the definitions of network simulations. These will allow them to evaluate performance of various proposals, like frame aggregation, in terms of network throughput, delay, packet loss, and other metrics. A detail description for a set of these use cases that we are going to use throughout this document have been given in the Chapter Usage Models.

2.4.1 Point-to-Point HT Goodput Test

This scenario is also known by the *Usage Models* reference document as *Model 17 - Point-to-Point Goodput Test* (Figure 2.10). In this part, we test the efficiency of aggregation over two HT STAs as shown in Figure 2.10a. The first station has a compound data source providing 100 Mb/s or 200 Mb/s Offered Load (OL) of Voice (VO) traffic with MSDU size of $1,500$ bytes (see Table 2.10c) with an MIMO-Zero Forcing (ZF) [69] channel (see Table 2.10b). Note that the 802.11 header has been modified to add a new field to classify the type of traffic, e.g. a VO traffic. The TID is used to select a UP for prioritized QoS or a TSPEC for parametrized QoS. TID values between $0 - -7$ are considered user priorities and these are identical to the IEEE 802.1D [70] priority tags and values between 8–16 refer to TSPECs.

Before we can discuss the simulation results, let us define two measurements for the performance evaluation: Aggregate Goodput and MAC Efficiency. Aggregate Goodput is the aggregate number of bits in MSDUs received at the MAC



(a) Layout

Parameter	Value
Antenna Config.	2 x 2
Band	20 Mhz
T_x Power	17 dBm
R_x Noise	10 dB
PHY Type	MIMO(ZF)
Basic Rate	16QAM 1/2
Oper. Rate	16QAM 1/2

(b) Channel Parameters

Traffic Generation Parameters		
MSDU Size		1,500 bytes
Offered Load		100 & 200 Mb/s
VO Traffic		
STAs 0 & 1	TID	7
	TXOP	0.032 sec
	Min. CW	3
	Max. CW	7
	AIFS	2

(c) Traffic Parameters

Figure 2.10: Scenario 17 - Layout & General Parameters

SAP within the specified delay bound of the application's defined QoS requirements, divided by the simulation duration. A commonly related term is Throughput, except that its end results include flows that fail to meet their QoS objectives. On the other hand, MAC Efficiency is a measurement that can determine in terms of a percentile the aggregate goodput divided by the average physical layer data rate.

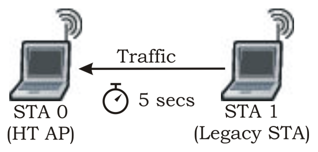
Table 2.3 shows the results of the aforementioned scenario within a simulation run of 5 second interval. The OL for the first and second cases is 100 Mb/s and 200 Mb/s, respectively. Also, for both cases, we can observe that the achieved PHY data rate that is achieved is in the region of 143 Mb/s, more than the goal rate that TGn has set. The reason why the packets that are sent (Indicated MSDUs) are twice the amount of frames that are received Received MPDUs, is because of the A-MSDU aggregation algorithm. So, for a 100 Mb/s offered load, the achieved Goodput is approximately 99.98 Mb/s with a MAC Efficiency of 69.6%. Similarly, for the case of 200 Mb/s offered load, the achieved Goodput is approximately 137.2 Mb/s with a MAC Efficiency of 95.6%. Although, the efficiency deviation for the above cases is 26%, it doesn't mean that there is an improvement with the overall performance. A given simple explanation is that in the initial case the system's resources have not been stretched as in the second case, where it is more likely to reach saturation quicker.

From	To	Received MPDUs	Indicated MSDUs	R_x Rate	Offered Load	Goodput	Efficiency
STA 0	STA 1	20,829	41,658	143.7382	100	99.9792	69.6%
STA 0	STA 1	28,583	57,166	143.5238	200	137.1984	95.6%

Table 2.3: Test HT with Scenario 17 - Numerical Results

2.4.2 Point-to-Point Legacy STA with HT AP

The following scenario describes an occasion where a legacy STA chooses to use a HT Access Point (AP). In order to review the consequences of this particular situation regarding the network's performance and STAs behaviour, we need to keep a simple structure that includes only the two participating nodes (e.g. like in Figure 2.11a). The traffic generation parameters are set as above in Table 2.10c but only for the case of an offered load of 100 Mb/s. Since the network includes a legacy STA the channel parameters will be set accordingly (see Table 2.11b). This scenario case follows *Model 18 - Point-to-Point Legacy Throughput Test* description of the *Usage Models* document.



(a) Layout

Parameter	Value	
	HT AP	Legacy STA
Antenna Config.	2 x 2	1 x 1
Band	20 Mhz	20 Mhz
T_x Power	17 dBm	17 dBm
R_x Noise	10 dB	10 dB
PHY Type	MIMO(ZF)	SISO
Basic Rate	16QAM 1/2	16QAM 1/2
Oper. Rate	64QAM 3/4	64QAM 3/4

(b) Channel Parameters

Figure 2.11: Scenario 18 - Layout & Channel Parameters

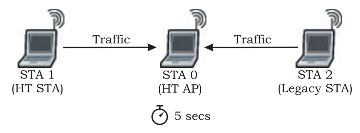
Our main concern over this simulation is the AP to be able to operate sufficiently with the legacy device. From the results shown in Table 2.4, we derive the conclusion that the STA was able to reach a throughput 44.72 Mb/s close to its PHY rate without any complications; keep in mind that the maximum data rate using OFDM is 54 Mb/s. The AP is capable to adapt SISO operation of a OFDM modulation for any legacy devices that it associates with, managing MAC Efficiency ratings of more than 88%. Note that the Received MPDUs and Indicated MSDUs show a kind of aggregation but this is because of legacy bursting and it is not a type of a new frame aggregation enhancement.

From	To	Received MPDUs	Indicated MSDUs	R_x Rate	Offered Load	Throughput	Efficiency
STA 0	STA 1	9,316	18,632	54.00	100	44.72	82.8%

Table 2.4: Test HT with Scenario 18 - Numerical Results

2.4.3 Point-to-Point Legacy and HT Co-existence

Earlier, we showed that a legacy device operates effectively over a network containing a single HT AP. Now, we are going to investigate a scenario (Figure 2.12) where two separate STAs, one HT STA and one STA belonging to the legacy standard, concurrently operate with the HT AP. This case in the *Usage Models* reference document is known as *Model 19 - Point-to-Point Legacy Sharing Throughput Test*. So, automatic rate adaptation in CSMA/CA WLANs may cause drastic throughput degradation for high speed bit rate STAs. The CSMA/CA medium access method guarantees equal long-term channel access probability to all hosts when they are saturated. The saturation throughput of any STA is limited by the saturation throughput of the STA with the lowest bit rate in the same infrastructure [71, 72, 73]. For example, it has been demonstrated that an IEEE 802.11g compliance network will achieve less throughput, in many cases halved, when an IEEE 802.11b depended STA shares the same resources with other IEEE 802.11g based STAs. To simulate this, we place both stations beside the HT AP over equivalent distances, which an illustration of the layout can also been see in Figure 2.12a, set the channel parameters as shown in Table 2.12b and configure with a VO traffic flow as before.



(a) Layout

Parameter	Value		
	HT AP	HT STA	Legacy STA
Antenna Config.	2 x 2	2 x 2	1 x 1
Band	20 Mhz	20 Mhz	20 Mhz
T_x Power	17 dBm	17 dBm	17 dBm
R_x Noise	10 dB	10 dB	10 dB
PHY Type	MIMO(ZF)	MIMO(ZF)	SISO
Basic Rate	16QAM 1/2	16QAM 1/2	16QAM 1/2
Oper. Rate	64QAM 3/4	64QAM 3/4	64QAM 3/4

(b) Channel Parameters

Figure 2.12: Scenario 19 - Layout & Channel Parameters

From	To	Received MPDUs	Indicated MSDUs	R_x Rate	Offered Load	Goodput	Efficiency
STA 1	STA 0	26463	26463	129.5194	100	63.5112	49.0%
STA 2	STA 0	4480	4480	53.9999	100	21.6829	40.1%

Table 2.5: Test HT with Scenario 19 - Numerical Results

The simulation outcome from Table 2.5 confirms that the multi-rate fairness issue still exists. The resolution of co-existence is not in favour of the system's overall performance as the higher throughput device achieves maximum goodput of 63.5 Mb/s while the legacy device manages a throughput of 21.7 Mb/s. Both STAs fall to half of their PHY data rate as the HT STA and legacy HT maintain a MAC Efficiency of 49% and 40%, respectively.

2.4.4 Frame Aggregation Evaluation

In this section, we compare the performance of A-MSDU and A-MPDU aggregation schemes along with the two-level aggregation as defined in the latest amendment of the IEEE 802.11 for HT devices. For the simulations, we used the aforementioned simulation model implemented by Intel and based on the OPNET Modeler with the latest PHY and MAC enhancements. So, each simulation run will be defining four cases for each combination of an aggregation scheme that the STA supports:

- Both A-MPDU and A-MSDU are enabled
- Only the A-MPDU algorithm is used
- Only the A-MSDU algorithm is used
- No aggregation at all

For each case, the traffic generation rate is configured high enough to saturate the air link rate that corresponds to the “PHY Peak” (144 Mb/s) on each figure, and the maximal A-MSDU length is 4 KB. In the first simulation, the OL is incremented by just varying the packet size while keeping constant the packet generation interval, also known as Constant Packet Rate (CPR), which is 40 μs. The initial OL is 25 Mb/s, and it increases up to 300 Mb/s with the packet size varying gradually (i.e.,125/250/500/750/$1,000/1,500$ bytes). Figure 2.13 illustrates the throughput results (in Mb/s) obtained from the MAC SAP while the OL from the associated HT STA is accumulating gradually.

As shown in Figure 2.13, we observe that all throughputs increase according to the load. In general, as the packet size increases, the A-MSDU stays below 75 Mb/s, while A-MPDU and the two-level aggregation achieve maximum throughputs of 136 and 134 Mb/s, respectively. When the packet size maintains values

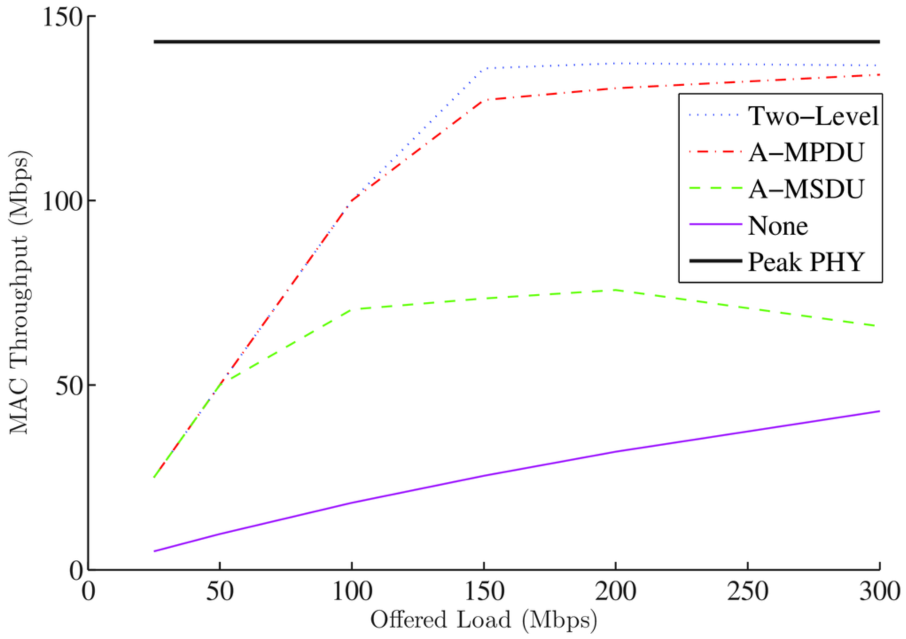


Figure 2.13: Throughput vs. increased offered load by varying the packet size

of 125 bytes (see OL = 25 Mb/s) and 250 bytes (see OL = 50 Mb/s), the corresponding throughputs for any type of aggregation are alike. This is because A-MSDU can aggregate several small packets within a single MPDU, even if the length is limited to 4 KB; the same way that A-MPDU can place multiple MPDUs in a single PSDU. Thus, for small packet sizes, we can choose any type of aggregation. On the other hand, when the packet size is larger than 250 KB, the throughput of A-MSDU distinguishes significantly from A-MPDU and the two-layer aggregation, with much lower values because the number of MSDUs that fit into a single A-MSDU is becoming less than the other cases. We can monitor this behaviour even more closely when packet size increases from $1,000$ bytes (see OL = 200 Mb/s) to $1,500$ bytes (see OL = 300 Mb/s). The A-MSDU throughput drops slightly for the reason that we had four packets of 1 KB, fitting to one A-MSDU, where in the case of $1,500$ bytes packet size, only two of them can occupy the same space. The same behaviour occurs with the two-level aggregation but only because of the A-MSDU stage. This is also why the A-MPDU throughput increases further when two-level aggregation remains at the same levels. The throughput for the no-aggregation case always increases with the increase of the OL by varying the packet size even after the channel is saturated. However, by only increasing the packet size up to the maximum Ethernet transmission unit (1, 500 bytes), without aggregation, will achieve throughput about three times lower than that of the A-MPDU and the two-level aggregation. This clearly demonstrates that small packet size is the key factor that lowers the throughput efficiency.

In the second simulation, we increase the OL by altering the packet interarrival rates instead of increasing the packet sizes, a situation known as Variable Packet Rate (VPR) evaluation. So the packet size remains constant at 1 KB during the simulation test. So, to attain OLs for a range of $25 \ldots 320$ Mb/s, we increase the packet interarrival rates from 320 μs to 25 μs. Note that the rates and the interval periods are reverse proportional, so if we want to increase the rate, we need to decrease the time interval analogous.

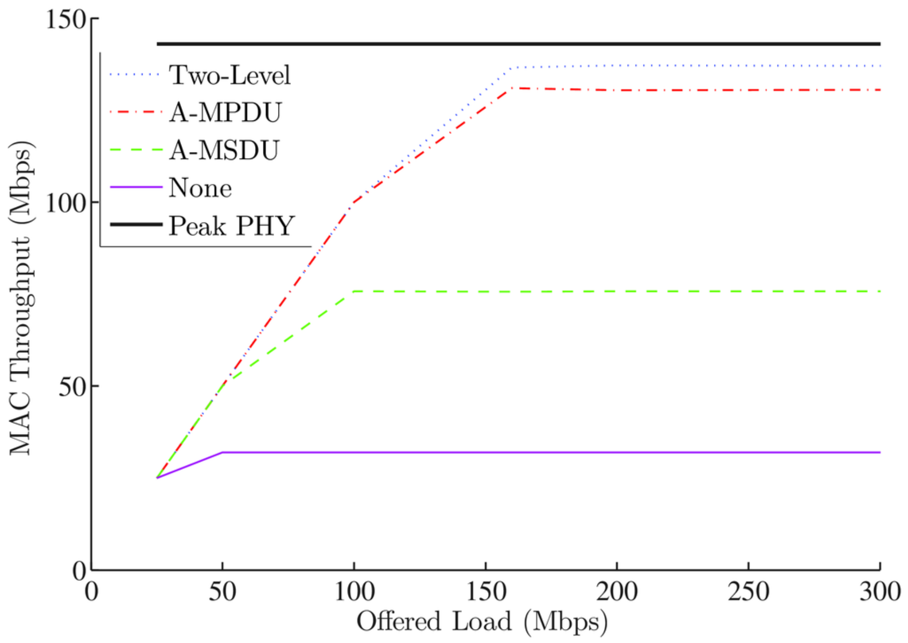


Figure 2.14: Throughput vs. increased offered load by varying the packet arrival interval (packet size = 1 KB)

From Figure 2.14, we can observe that the MAC throughput performance for all aggregation schemes increases respectively with the increase of the OL. Up to the second step of the simulations, with independent variable at 50 Mb/s, all three schemes provide an equivalent achieved goodput of 50 Mb/s too. For OLs 100 Mb/s and onwards, the A-MSDU fail to cope with the loading traffic capacity and remain to a level of throughput up to 75 Mb/s. Similar to previous results, the two-level aggregation again outperforms slightly the A-MPDU method with a deviation of around 6 Mb/s. Still the maximal goodput to the all aggregation schemes enabled is around 136 Mb/s and considering that the PHY data rate is 143.17 Mb/s, we can calculate the MAC Efficiency at 95.6%. Each case has a different saturation point yet the saturation behaviour illustrated by the graphs is slightly different than before. This has to do with the chosen value for the constant packet size, as we set it to the medium size, 1 KB, in order to avoid stretching the system's impending demand too early in the simulation runs. Furthermore, we can point out that the throughput achieved by the A-MPDU and the two-level aggregation after saturation is approximately 4.5 times higher than the no aggregation scheme.

This last simulation represents a scenario for a fixed OL of 100 Mb/s with variant packet sizes and interval times. There is no channel saturation since the required traffic demand stays at moderate level. Along with the throughput values for each type of aggregation, we investigate the degree of aggregation that the two-level aggregation performs, by comparing the number of indicated MSDUs and the received MPDUs.

In Figure 2.15, the A-MPDU and the two-level aggregation achieve the 100 Mb/s goal that HTSG has set, whereas A-MSDU falls below that threshold at around 75 Mb/s. Although the A-MSDU mechanism can achieve higher throughput than the legacy IEEE 802.11 standards, it does not utilize the channel as fully as A-MPDU and the two-level aggregation do. However, there is an exception for the A-MPDU function when packet size is 125 bytes and packet interval rate at 10 μs each. This shows that this type of aggregation cannot handle consecutive accumulate small size packets and a small portion of the overhead problem still remains. It is very important to understand how this blend of A-MSDU and A-MPDU, in most cases, is capable of improving the effectiveness of the MAC layer,

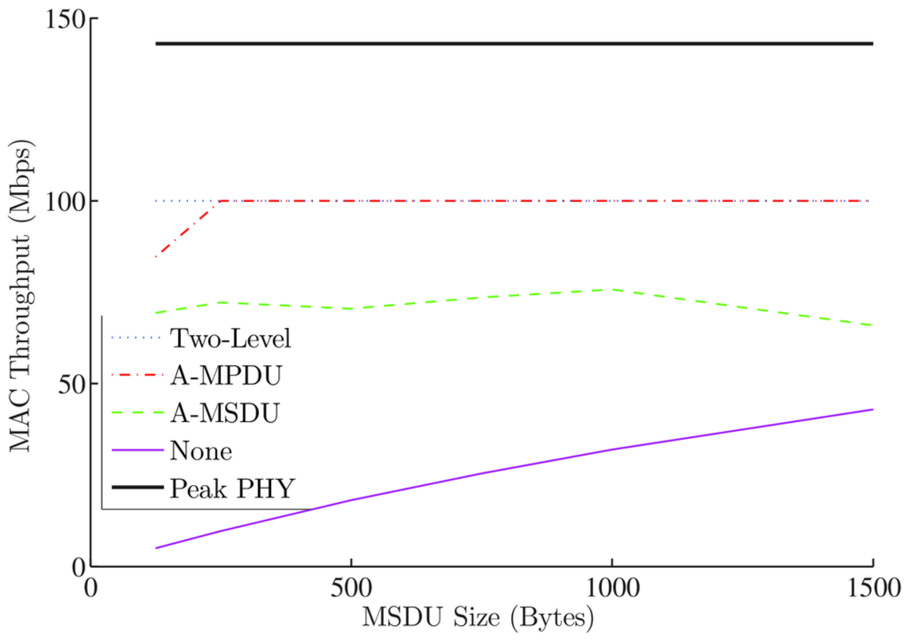


Figure 2.15: Throughput vs. increased MSDU size

specifically when there are many small MSDUs, such as TCP acknowledgements. For example, when the packet size is set for 125 bytes, there are approximately 999, 775 MSDUs generated during the simulation period. In spite of this, the number of MPDUs received at the receiver is 34, 476 MPDUs, so a single MPDU included on average around 29 MSDUs. The latter suggests that a huge MAC and PHY overhead was avoided. For the largest packet size of 1, 500 bytes, there are approximate 83, 324 indicated MSDUs and about 41, 662 received MPDUs, exactly two packets in each MPDU as it is bounded by the maximum A-MSDU length of 3, 839 bytes. In conclusion, although all schemes employ the same PHY techniques, a huge difference can be seen between the aggregation scheme and the one which disables them. A MAC efficiency of the two-level aggregation is calculated at 70% and for the non-aggregation option is well below at 30%. The latter clearly shows the efficiency over the system's performance of the frame aggregation enhancements.

2.5 Summary

Within this chapter, we investigated the main types of frame aggregation as proposed in the IEEE 802.11n standard: A-MSDU, A-MPDU and the combined two-level aggregation. Our simulation results demonstrated that those concatenation/packing mechanisms performing over different sub-layers can actually increase the channel efficiency of the 802.11 MAC in the next-generation WLANs. We also demonstrated that by exclusively increase the PHY layer transmission rate, the results won't show analogous action and the initial output is around 4.5 times less bandwidth than the two-level aggregation.

All types of aggregation are highly required as they resolve the fundamental problem of inefficiency which is thePHY/MAC overhead. However, the standard only identifies the concepts and the data frame structures, which in a flawless environment they can deliver attracted results but in terms of their functionality there are still questions and issues that need further investigation. For example, the processing time that is needed to compute these aggregates can increase the overall delays. Actually, as the efficiency of the aggregation increases, its operation becomes more complex (e.g. two-level aggregation). Another question is how large should the devices set the concatenation threshold. Ideally, the maximum value is preferable but in a noisy environment, short frame lengths are preferred because of potential retransmissions. The concatenation schemes only operate over the packets that are already buffered in the transmission queue, so if the data rate is low then efficiency will be low too. There are many ongoing studies that are introducing alternative queuing mechanisms than the usual FIFO. A combination between frame aggregation and an enhanced queuing algorithm could increase channel efficiency further.

Chapter 3

Frame Aggregation as a $M/G^{[a,b]}/1/K$ Queue

The first batch service models initially derived to describe the efficiency of batch service workstations within the production and manufacturing environment [74]. Other examples can also been seen in the customer services sector and in the area of communication and information systems. Probabilistic model-based techniques can show the reliability and availability for current or future system's configuration on various scenarios with versatile customer workload that is served as bulks [75]. Throughout this paper, the majority of the content is evaluated using empirical deductive methods of measurement and the results are derived from the execution of various simulations. Nevertheless, in this specific instance, it is easy to assume that the process of the frame aggregation that was described in the previous section can easily correspond to a $M/G^{[a,b]}/1/K$ queueing system so stochastic and probabilistic methods can be applied to obtain conclusions about the performance and reliability properties.

There is a huge debate over which queuing model is suitable for describing a WLAN. Historically, the most common approach to model the traffic's interarrival times is the exponential distribution [76] with memoryless properties. Poisson models have been in use in the literature since the advent of computer networks, and before that in the telecommunications arena [45]. These models are very attractive from an analytical point of view [45], and with proper selection of parameters a Poisson model can be fit to most network traffic traces reasonably well for short periods. Later studies also show that in long term, data traffic has self-similar characteristics [77, 78, 79] and so Poisson-based models do not adequately model self-similar processes [80], either in modelling traffic for analysis or in generating traffic for simulations. Recent refinements attempted to add traffic “burstiness” and self-similar property on top of Poisson distributions, such as Compound Poisson [81], Markov-Modulated Poisson Process (MMPP) [82, 83], Packet Trains [84], etc. However, self-similar models may prove to be complex in analytical solutions and not always appropriate [85]. Thus, $M/G/1/K$ as a queuing model for WLANs could prove as a wise choice as it is stable, simple, accurate over short term and widely used.

Within an elementary queueing system $M/G/1/K$ [86] and taken that its finite buffer queue is not full, newly arriving jobs will be served while the system is “idle”, otherwise are buffered to a storage area and wait for their turn. Similarly, within a network system, the packet that arrives at the transmission link will be either buffered and wait if channel is “busy” or set for transmission over the communication channel while the channel remains “idle”. As our STA’s transmission queue serves packets in batches, we can represent that system’s behaviour with the use of a $M/G^{[a,b]}/1/K$ queueing model. In [87, 88], Chaudhry and Gupta reduce the computation effort required to derive the queue statistics of interest and combined with the classical algorithms for the $M/G^{[a,b]}/1/K$ queue [74, 75, 89, 90, 91], we are able to introduce and determine the behaviour of frame aggregation and its threshold variables $(a,\ b)$. In this section, we review the model definition and present numerical results for various classes of service processes, different service starting or batch collection rules under various load conditions. The main purpose of this chapter is to empirically prove that there isn’t an optimal set of $(a,\ b)$ that can maximize utilization and minimize queue waiting times. Instead, each system’s performance will be dependent over the service time and traffic intensity

3.1 Model Description

Let us assume that the model consists of a finite capacity queue (waiting space) of size K and it is served by a single batch server according to a commencing scheme which is driven by the number of packets waiting in the queue. Let packets arrive in a Poisson process (exponential distribution, M) at average arrival rate λ induced by a Poisson Arrivals See Time Averages (PASTA) property. The server has a maximum capacity of b, where $b < K$ and the service times are independent identically distributed (i.i.d.) random variables (r.v.s) with distribution function (d.f.) $B(t)$ having mean $1/\mu$ (general distribution, G). When the server is idle and there are less than a number of packets in the queue (quorum), the server remains idle until enough packets have been accumulated. At the end of a service phase, the server will proceed according to the number of waiting packets. If there are more than a number of packets in the queue at the scheduling time, the server will start the next service immediately by taking up to b waiting packets. Packets seeing upon arrival a full queue are thought of to be dropped (blocked). The traffic intensity is given by the utilization factor ρ and is defined to be $\lambda/b\mu$. A simple illustration of the single node queueing model can be seen in Figure 3.1.

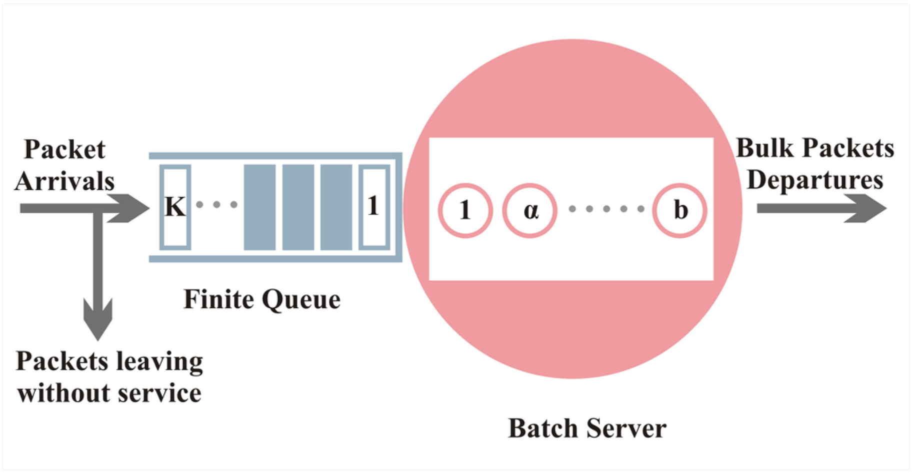


Figure 3.1: The basic $M/G^{[a,b]}/1/K$ queueing model

We approximate as $Z(t) = (X(t), U(t))$ a two-dimensional Markov stochastic process, with $X(t)$ denoting the number of packets in the queue at time t and $U(t)$

the remaining service time for the batch actually in service at time t. Because of the imbedded nature of the process it is known as an imbedded Markov chain model. Further, define $P_{k,0}(t)$ and $P_{k,1}(t)$ the probabilities that there are K waiting packets in the queue at a random epoch (arbitrary point) and the server is "idle" and "busy", respectively.
More specifically,

$$\begin{aligned} P_{k,0}(t) &= P\{X(t)=k, U(t)=0\}, && 0 \leqslant k \leqslant a-1, \\ P_{k,1}(u,t)du &= P\{X(t)=k, u < U(t) \leqslant u+du\}, && u \geqslant 0, 0 \leqslant k \leqslant K. \end{aligned}$$

Let

$$\begin{aligned} P_{k,0} &= \lim_{t\to\infty} P_{k,0}(t), && 0 \leqslant k \leqslant a-1, \\ P_{k,1}(u) &= \lim_{t\to\infty} P_{k,1}(u,t), && 0 \leqslant k \leqslant K. \end{aligned}$$

Now, let P_k^+ $(0 \leqslant k \leqslant K)$ be the probability of k packets in the queue immediately after a transmission of a batch (departure epoch). Since, this probability is relative to $P_{k,1}(0)$ $(0 \leqslant k \leqslant K)$, we can derive some association between them

$$P_k^+ = \frac{P_{k,1}(0)}{\sum\limits_{i=0}^{K} P_{i,1}(0)} \tag{3.1}$$

Note that from the summation $\sum\limits_{k=0}^{K} P_{k,1}(0)$, we can get the departure rate of the aggregated frames given that the server is busy. Now, in order to resolve equation 3.1, we would first need to derive some relations of the states of the system at times t and $t + \Delta t$ in steady state but even so the computation of $P_{k,1}(0)$ would had been complex. Only for a simple queuing system $M/G^{[a,b]}/1/K$ with $a = b = 1$ could be simple. To overcome this problem, we are employing an imbedded Markov chain technique similar to the one carried out by Gold and

Tran-Gia in [92] and the process will initially help us to get P_k^+. Subsequently, we make use of equation 3.1 to develop relations between arbitrary($P_{k,0}$ and $P_{k,1}$) and departure(P_k^+) epoch probabilities.

The imbedded Markov chain analysis carried out in [92] is briefly discussed here for the sake of completeness. The $\{P_k^+\}$ can be obtained by solving the system of equations $\mathbf{P}^+\mathbf{P} = \mathbf{P}^+$, where $\mathbf{P}^+ = [P_0^+, P_1^+, \ldots, P_K^+]$ and $\mathbf{P} = (p_{ij})$ is one-step transition probability matrix with p_{ij}s given by

$$p_{ij} = \begin{cases} d_j, & i = 0, 1, \ldots, b,\ j = 0, 1, \ldots, K-1 \\ d_j - (i-b), & j \geqslant i-b,\ i = b+1, b+2, \ldots, K,\ j = 0, 1, \ldots, K-1 \\ 1 - \sum\limits_{r=0}^{K-1} p_{ir}, & i = 0, 1, \ldots, K,\ j = K \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

and

$$d_j = \int_0^\infty \frac{e^{-\lambda v}(\lambda v)^j}{j!}\,\mathrm{d}B(v) \tag{3.3}$$

The expressions for d_j represent the probability of j arrivals during a service period and can be easily obtained for various service-time distributions.

It has been found from [87] the following association between state probabilities at arbitrary epoch, $P_{k,0}$ $(0 \leqslant k \leqslant a-1)$ and $P_{k,1}$ $(0 \leqslant k \leqslant K)$, and departure epochs P_k^+. These relations are given by the following equations (3.4, 3.5)

$$P_{k,0} = \frac{\sum\limits_{j=0}^{k} P_j^+}{\rho b + \sum\limits_{i=0}^{a-1}(a-i)P_i^+}, \qquad 0 \leqslant k \leqslant a-1 \tag{3.4}$$

$$P_{k,1} = \frac{\sum\limits_{j=k+1}^{min(b+k,K)} P_j^+}{\rho b + \sum\limits_{i=0}^{a-1}(a-i)P_i^+}, \qquad 0 \leqslant k \leqslant K-1 \tag{3.5}$$

and

$$P_{K,1} = 1 - \left(\sum_{k=0}^{a-1} P_{k,0} + \sum_{k=0}^{K-1} P_{k,1} \right) \tag{3.6}$$

The above relations are much simpler than the one derived in [88, 87] and are also computationally more efficient even though both approaches provide matching results.

3.2 Substitute Service Time Distribution Functions

The service time distribution is assumed from the description to be generally distributed. In order to have a parametric representation of the service time, for each time state the expressions of the transition probabilities given in the Laplace-Stieltjes domain need to be transformed by the Laplace inversion procedure. Also, in practice it often occurs that the only information of random variables that is available, is their mean and standard deviation and not the real data. Consequently, the calculations are of higher complexity or infeasible. However, for reasons of computing efforts, to obtain an approximating distribution it is common to fit a phase-type distribution on the mean, $E(X)$, and the coefficient of variation, c_X, of a given positive random variable X by using the two-moment approximation technique, as proposed in [93, 94, 92]. Note that some argue that an approximation of the distribution of the MAC layer service time can be represented by a Chi Square distribution (X^2) with degrees of freedom according to the set (a, b) [95] or a negative binomial distribution for a parametric representation of stochastic processes could have been used in discrete time domain. Nevertheless, as substitute processes we choose a simple combination of phases, allowing the approximation of any process with respect to their first and second moments. Two cases are considered: i) a hypo-exponential process type ($0 \leqslant c_B \leqslant 1$), comprised by a series of a deterministic (D) and an exponential (M) phase, and ii) a hyper-exponential process type ($c_B > 1$), an alternate of two exponentials (H_2) with balanced means. Mathematically, these processes are described by

Case 1: $0 \leqslant c_B \leqslant 1$

$$F_B(t) = \begin{cases} 0, & 0 \leqslant t \leqslant t_1 \\ 1 - e^{-(t-t_1)/t_2}, & t \geqslant t_1 \end{cases} \tag{3.7}$$

where $t_1 = E(B)(1 - c_B)$ and $t_2 = E(B)c_B$

Case 2: $c_B > 1$

$$F_B(t) = 1 - pe^{-t/t_1} - (1-p)e^{t/t_2} \tag{3.8}$$

where $t_{1,2} = E(B)\left(1 \pm \sqrt{\frac{c_B^2-1}{c_B^2+1}}\right)^{-1}$ and $p = E(B)/2t_1$, $pt_1 = (1-p)t_2$

We remark that the results are slightly dependent on the chosen type of the substitute process. If there is evidence of a much different process characteristic, appropriate other substitutes can be chosen too. The authors in [96, 97], suggest a choice of more sophisticated use of phase-type distributions by trying to match the first three (or even more) moments of the random variable or at least to approximate the shape. Also, for a two-moment fit, one can apply an Erlang distribution for the case $0 \leqslant c_B \leqslant 1$ or even brake it down to a subclass of $c_B > 0.5$ which a Coxian-2 (a mixture of two Erlang) distribution can be used [98]. Finally, we note that if the component process are Markovian, the resulting process is Markovian again.

3.3 Performance Measures

The most relevant performance measures in the analysis of queueing models and of interest to our purpose are:

- The blocking probability. The probability that the system will reject new arrivals as the queue buffer has exceeded its limits. Usually, this occurs when $\rho > 1$.

- The waiting time and the sojourn time of a packet. The sojourn time is the waiting time plus the service time.

- The number of packets in the system (including or excluding the one or those in service).
- The amount of packets the server takes at each start.
- The busy (occupied) or idle periods of the server. These are periods of time during which the server is working continuously or remains without packets, respectively.

In particular, we are interested in mean performance measures, such as the mean waiting time, the mean queue length, etc. The mean waiting time can of course be calculated from the Laplace-Stieltjes transform (L.S.T.) by differentiation of the main function. Equivalently, the mean waiting time can also be determined directly (i.e., without transforms) with the mean value approach. This happens because for systems with Poisson arrivals, a very special property holds, that arriving customers find on average the same situation in the queueing system as an outside observer looking at the system at an arbitrary point in time.

There are numerous approximations for the blocking probability; usually denoted as $B(K, \rho)$ but since we express as $B(\cdot)$ the service time distribution function, we use the term P_B. So, the arbitrary time state probability from equation 3.6 can be used directly to calculate the blocking probability P_B.

A new arriving packet, first has to wait for the residual service time of the packets in service (if there is one) and then continues to wait for the servicing of all packets who were already waiting in the queue on arrival. In general, the mean waiting time, $E(W)$, in the queue can be easily be derived by applying Little's Law, so:

$$E(W) = \frac{E(L_q)}{\lambda(1 - P_B)} \tag{3.9}$$

where $E(L_q)$ is the mean queue length and is given by equation 3.10

$$E(L_q) = \sum_{k=0}^{a-1} k(P_{k,0} + P_{k,1}) + \sum_{k=a}^{K} kP_{k,1} \tag{3.10}$$

The amount of accepted traffic is $\lambda(1-P_B)$, Thus, again with Little's theorem, we get the equation for the mean number of packets in the server, $E(S)$:

$$E(S) = \lambda(1 - P_B)E(B) \tag{3.11}$$

And more specifically, we are interested in the number of packets the server aggregates at each start. This indicates the efficiency of the transmitter handling and starting rules. The measure of the mean number of packets per start, $E(S^A)$, is given by

$$E(S^A) = a\sum_{k=0}^{a-1} P_k^+ + \sum_{k=a}^{b-1} kP_k^+ + b\sum_{k=b}^{K} P_k^+ \tag{3.12}$$

Once we know P_k^+, the expected occupied and idle periods, $E(O)$ and $E(I)$, respectively, can be calculated as

$$E(O)\frac{1}{\sum\limits_{k=0}^{a-1} P_k^+} \tag{3.13a}$$

$$E(I) = \frac{\sum\limits_{k=0}^{a-1}(a-1)P_k^+}{b\rho\sum\limits_{k=0}^{a-1} P_k^+} \tag{3.13b}$$

3.4 Numerical Examples

In this part, we present numerical results for various classes of service processes, different service starting or batch collection rules under various load conditions. In the discussion of the results we stress the influence of firstly, the variation of the service process, secondly, the service starting threshold dimensioning and finally the traffic intensity on the mean waiting time and on the average number of packets per start. In accordance to the substitute distributions discussed in Section 3.2 we use a series of a deterministic and an exponential D+M distributions and a H_2 distribution with balanced means to achieve service time handling with coefficients $0 \leqslant c_B \leqslant 1$ and $c_B \leq 1$, respectively.

Let us assume that the time variables are standardized by $E(B) = \mu^{-1} = 1$, consequently the offered traffic intensity becomes $\rho = \lambda/b$. For the following

$M/G^{[a,b]}/1/K$ system we set as server capacity, b, and maximum waiting space, K, 32 packets and 64 packets, respectively.

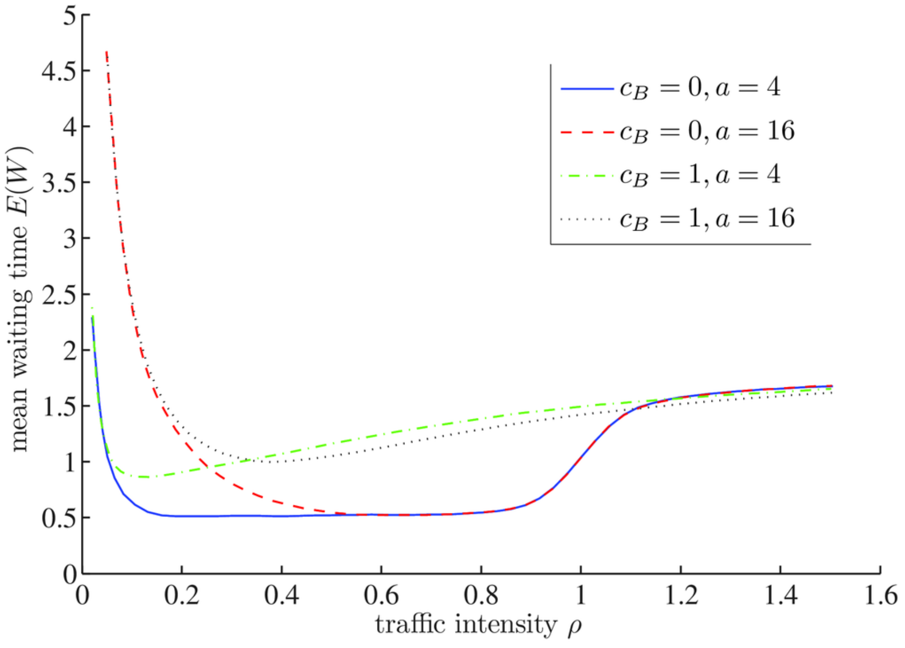


Figure 3.2: Waiting time behaviour

For the first results, we set a combination of variation of the service starting threshold, 'a', and the coefficient of the service time, 'c_B'. So for this simulation we choose two set of values ($a = 4$, $a = 16$) and ($c_B = 0$, $c_B = 1$), for the service quorum and c_B, respectively. The aim is to distinguish the behaviour of the system over a wide range of the utilization factor ρ ($0 \leqslant \rho \leqslant 1.5$) with a close to unity ($a = 4$) and much larger quantity which is half the server capacity ($a = 16$), in correspondence with both aforementioned distributions, thus $c_B = 0$ and $c_B = 1$. Figure 3.2 shows the mean waiting time, $E(W)$, as a function of the traffic intensity, ρ. As can be seen, with traffic intensity very low, the mean waiting time is very high especially when the service starting threshold 'a' is much larger than 1 and as the traffic intensity increases it draws towards its minimum expected waiting time. After it exceeds its stability point at approximately $\rho \approx 0.9$, it starts ascending again. As the limit for $\rho \to \infty$ all graphs tend to $E(W) = K/b = 2$.

In order to understand the influence of the service process, Figure 3.3 shows

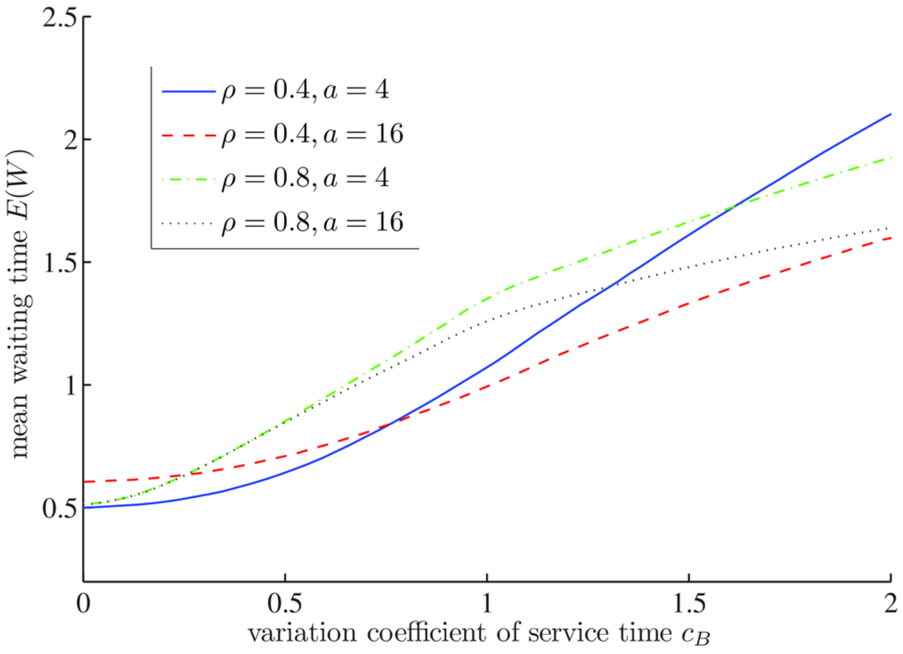


Figure 3.3: Influence of service process

the mean waiting time of a packet, $E(W)$ over a range of c_B that correspond to $D+M$ and H_2 distributions, for that reason $0 \leqslant c_B \leqslant 1$ and $c_B \leq 1$. The utilization ρ is set in a suitable way to provide stationary behaviour ($\rho < 1$) but yet $\rho = 0.4$ and $\rho = 0.8$ can distinguish low from high traffic intensity. As a general observation, it appears a discontinuous behaviour of the curves at the interchange point($c_B = 1$) of these distribution types substitutes which unquestionably is due to the unnatural element of the representation of the service time distribution. Regardless, for the case of deterministic service time the best batch collection rule is not to collect batches at all but to start the server even with only a single packet in the queue. Also, in the case of $\rho = 0.4$ there is a crossover of the waiting time diagrams for service starting threshold $a = 4$ and $a = 16$. This is due to the fact that normally during shorter service periods less packets will arrive and thus the server is often caused to work inefficiently. Overall, the reduction of waiting time gained by choosing the service starting threshold 'a' appropriately gets larger with growing coefficient of variation of the service time and diminishes slightly with higher traffic intensity.

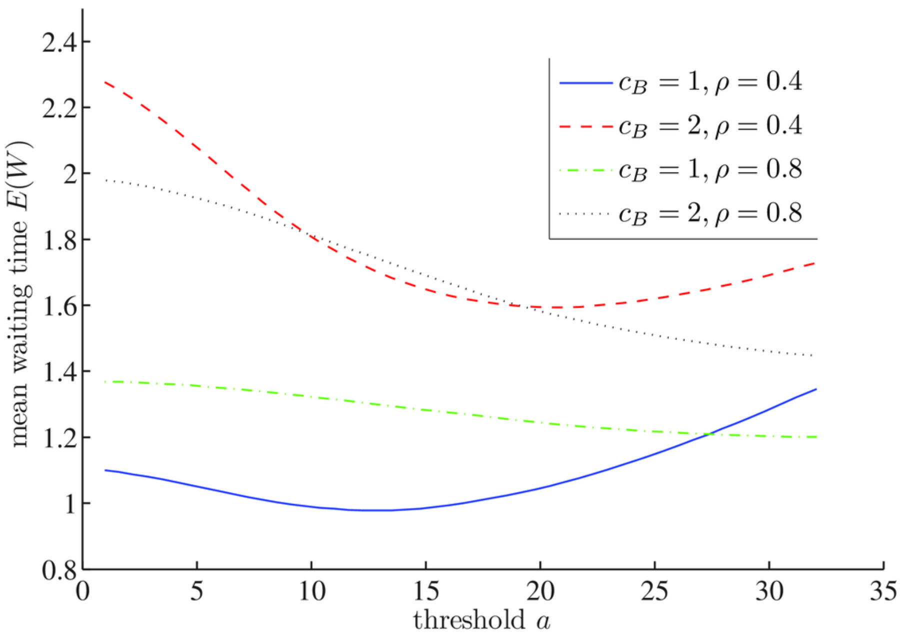


Figure 3.4: Threshold dimensioning aspects

Figure 3.4 demonstrates that the optimal choice of the quorum is not always as clear as for the case with $c_B = 0$, e.g. the best batch collection rule is not to collect batches at all but to start the server even with only a single packet in the queue, but becomes more critical for higher variations of the service time. The superposition of the diverse dependencies of the waiting time leads to a special appreciation for the choice of the service starting threshold 'a' for each set of parameters c_B and ρ.

So far, we analysed an optimized solution for minimizing the waiting time for various traffic types by considering the batch collection rule. However, if we are seeking to utilizing the server in an efficient way, we may have to consider a different approach. Figure 3.5 shows the average number of packets per start, $E(S^A)$, as a function of the traffic intensity, ρ. We derive the deduction that for constant service time and low traffic intensity, the average number of packets per start with small service starting threshold is significantly smaller than with larger service starting threshold. On the other hand, in the case of high traffic intensity, the service starting threshold has no influence on the average number

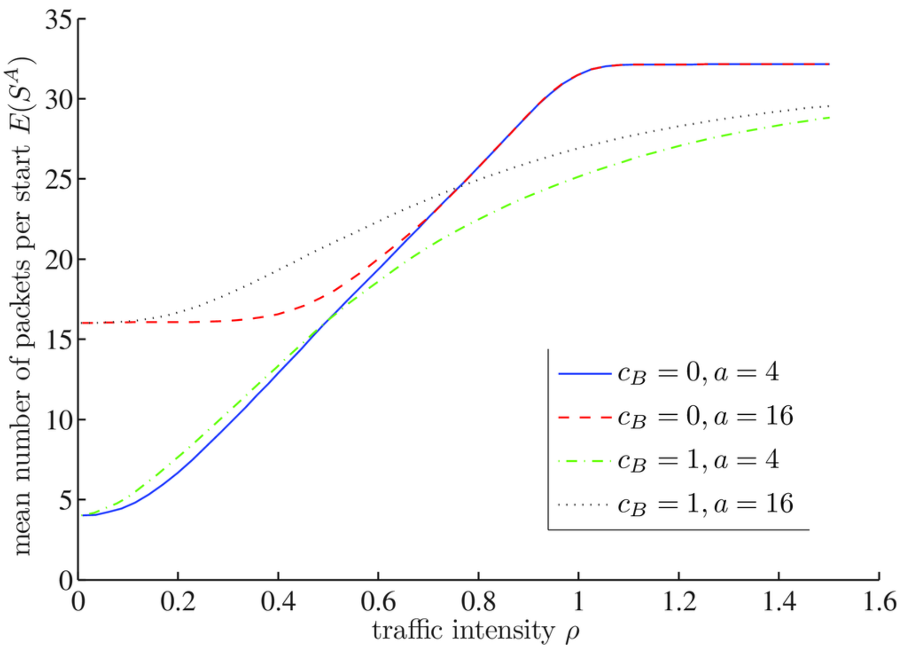


Figure 3.5: Server Utilization

of packets per start. As the limit for $\rho \rightarrow \infty$ all graphs tend to the maximum server capacity, $b = 32$. Regarding the coefficient of variation of service time, when the parameter is set to $c_B = 1$, the choice of a effects the average number of packets per start even when traffic intensity is high; this tendency can also be seen over our previous results (e.g. see Figure 3.2) where service starting threshold signifies analogous the mean waiting time. On the contrary, despite the value of c_B, in the case of low traffic intensity, the service starting threshold 'a' imposes contradictory consequences between minimizing mean waiting time and maximizing server utilizations, as the former is inverse proportional and the latter proportional.

In Figure 3.6, we conduct a model simulation for the blocking probability P_B versus the traffic intensity ρ when the service starting threshold is set with values $a = 4$ and $a = 16$ and for the coefficient of variation of the service time two diverse values are assigned, $c_B = 0$ and $c_B = 2$. As expected, the blocking probability tends to high results as the traffic intensity increases. Nevertheless, for smaller coefficient of variation of the service time the blocking probability is

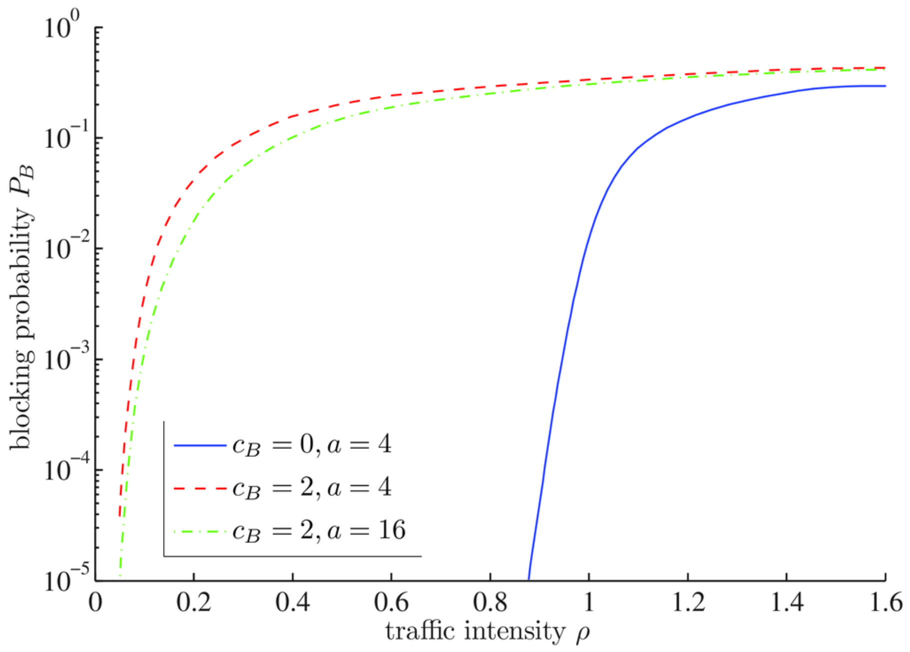


Figure 3.6: Blocking probability

less and for the curves with coefficient $c_B = 2$, the P_B increases as the service starting threshold is set to smaller values.

3.5 Summary

In conclusion, depending on the question which viewpoint is more important, minimizing the waiting time or maximizing utilization, a trade-off choice has to be taken into account. We point out that it doesn't exist a batch collection rule which is generally valid so each individual case has to be considered separately. Except of designing and implementing an algorithm which would be able to control the server's initiation mechanism besides the server starting threshold according to some pre-set parameters. This method will dynamically determine and adapt the starting based upon the traffic's characteristics, the packet maximum delay requirements and the server's maximum capacity.

Chapter 4

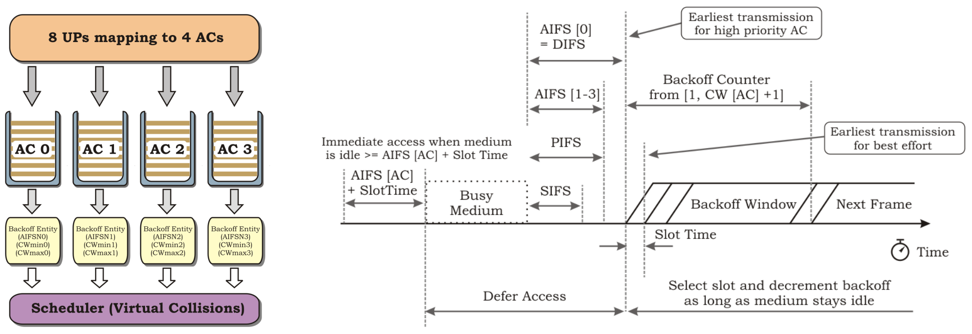
IEEE 802.11n and QoS in Conjunction

The IEEE 802.11n latest amendment attains rates of 100+ Mb/s by introducing innovative enhancements at the PHY and MAC, e.g. MIMO and Frame Aggregation, respectively. However, the performance improvement potentials may be limited by the interaction between prioritized and parameterized channel access scheduling mechanisms defined for the QoS support and the enhanced asynchronous data service for the increased MAC efficiency. So, if STAs of multiple priorities share the wireless medium at the same time, the IEEE 802.11e amendment defines a prioritization method where the higher priorities STAs should maintain shorter waiting channel access periods. Consequently, as we will show here, the higher priority STAs will tend to have small aggregate sizes and lower priority STAs a small channel access frequency, both effects result in poor channel utilization and overall poor network performance. In the following sections, we will briefly describe 802.11e's EDCA mechanism and how this interferes with the new HT enhancements.

4.1 IEEE 802.11e and EDCA

Several related aspects of traffic grade of service standards for the most recent HT amendment builds upon IEEE 802.11e's probabilistic priority mechanisms. This

QoS standard is considered of critical importance for delay-sensitive applications, such as VoIP over WLAN and streaming multimedia. It enhances the legacy DCF and PCF, through a new coordination function, known as HCF. Within the HCF, there are two methods of channel access, similar to those defined in the legacy 802.11 MAC, the HCCA and the EDCA. Both EDCA and HCCA have been thoroughly studied and discussed about the QoS improvements over the legacy standard by the research and academic community [99, 100, 101, 102, 103]. Since the distributed coordination mechanism that is based on the CSMA/CA function is of interest for this study, HCCA will not be discussed any further. A simple illustration of EDCA's operation and mechanism can be seen in Figure 4.1.


8 UPs mapping to 4 ACs
AC 0
AC 1
AC 2
AC 3
Backoff Entity (AIFSN0) (CWmin0) (CWmax0)
Backoff Entity (AIFSN1) (CWmin1) (CWmax1)
Backoff Entity (AIFSN2) (CWmin2) (CWmax2)
Backoff Entity (AIFSN3) (CWmin3) (CWmax3)
Scheduler (Virtual Collisions)
AIFS [0] = DIFS
Earliest transmission for high priority AC
AIFS [1-3]
Backoff Counter from [1, CW [AC] +1]
Earliest transmission for best effort
Immediate access when medium is idle >= AIFS [AC] + Slot Time
PIFS
AIFS [AC] + SlotTime
Busy Medium
SIFS
Backoff Window
Next Frame
Slot Time
Time
Defer Access
Select slot and decrement backoff as long as medium stays idle

(a) The four ACs (b) IEEE 802.11e interframe space relationship

Figure 4.1: An example of EDCA operation

The QoS support in EDCA is provided by the introduction of prioritization via distinguishing the traffic flows into ACs including a set of backoff entities for each AC, such as minimum and maximum CW and AIFS duration, seen in Figure 4.1a. Note that, the AIFS timers assigned by IEEE 802.11e are all defined as one SIFS value plus a variable number of slots times, known as AIFS-number (AIFSN), times T_{slot}, the duration of a time slot set by the physical layer encoding method in-use. The formula to calculate the AIFS in time slots for each AC is given by the equation $AIFS[AC] = SIFS + AIFSN * T_{slot}$. There are four ACs defined as FIFO queues, according to their target application, i.e., Best Effort (BE), Background (BK), Video (VI) and VO, also known as $[AC\_0, AC\_1, AC\_3, AC\_4]$ with the enumeration denoting the order of importance from low to high priority and following the same order of the applications given above. So, BE traffic maintain the lowest priority while VO the highest. Consequently, there are four distinct sets of contending entities with separate values between them that define relative priority in medium access per AC. The main idea is to use four coupled CSMA/CA queue mechanisms one for each AC that behaves as a single enhanced DCF contending entity, and all to contend for access to the same medium. However, each AC is parametrised with different set of values, so higher priority traffic has certain parameters to allow it to gain access to the channel earlier than the lower priority traffic which have longer backoff timers and Inter-Frame Space (IFS) periods. An example of the default set values for the parametrization of each AC as defined in the IEEE QoS standard are given in Table 4.1.

Parameter	**AC_BE**		**AC_BK**		**AC_VI**		**AC_VO**	
AIFSN	7		3		2		2	
CW_{min}	15		15		7		3	
CW_{max}	1,023		1,023		15		7	
TXOP Limit	802.11a,g,n	0 μs	802.11a,g,n	0 μs	802.11a,g,n	3,008 μs	802.11a,g,n	1,504 μs
	802.11,b	0 μs	802.11,b	0 μs	802.11,b	6,016 μs	802.11,b	3,264 μs

Table 4.1: EDCA parameters for each AC

An 802.11e STA that obtains medium access must not utilize radio resources for duration longer than a specified limit. This important new attribute of the 802.11e MAC is referred to as a TXOP. A TXOP is an interval of time during which a backoff entity has the right to deliver MSDUs and therefore is an important means to control delivery delay. A TXOP is defined by its starting time and duration which is limited by a parameter that takes a default set value from the standard. When TXOP is equal to 0, the standard defines that a single MSDU, PPDU, A-MSDU or A-MPDU is allowed to be transmitted. But in a nutshell, an HT STA that is a TXOP holder may transmit multiple MPDUs of the same AC within an A-MPDU as long as the duration of transmission of the A-MPDU plus any expected BlockAck response is less than the remaining Transmitter Network Allocation Vector (TXNAV) timer value that was initialized with the duration from the Duration/ID field in the frame most recently transmitted successfully.

As high priority flows have poor channel utilization because of their traffic characteristics, the low priority flows throughput can be amerced even further. Apart from the traffic load, where a high offered load from the application will signify a big pile in the MAC stack, we need to investigate analytically the operation of EDCA on each prioritized flow.

4.2 An Analytical Model for EDCA

Most of the recent analytical work on the performance 802.11e EDCA stems from the simple and fairly accurate model proposed by Bianchi [104] to calculate saturation throughput of the legacy DCF. Later, Ziouva and Antonakopoulos [105] improved the model by stopping the backoff counter during busy slots, which is more consistent with the standard, and to find saturation delays but traffic differentiation still was not considered. Based on these analysis, Xiao [106, 107] and many others extended the model to prioritized schemes for EDCA by introducing multiple ACs with distinct parameter settings, such as minimum and maximum CW, different AIFS and TXOP parameters, finite retry limit, etc. Many of the related works cover assumptions of a fully saturated channel [108, 109, 110, 111] and other non-saturated cases [112, 113, 114]. By saturation, we mean the network is overloaded and each node always has packets to transmit. As a matter of fact, all the nodes are continuously contending for accessing the channel, leading to a high level of packet collisions especially in the presence of a large number of nodes. On the other hand, if the system operates under unsaturated traffic conditions, the network has less contention as not all nodes accessing the channel at every time event, resulting in less packet collisions and packet retransmissions.

We assume a system consists of n_i, $(i = 0, \ldots, N-1)$ STAs for each N ACs priority i classes, also here we consider the order of 0 corresponding to the lowest priority and $N-1$ to the highest. Following the considerations of [106], for a given STA that belongs to AC_i, $(i = 0, \ldots, N-1)$ priority class, $b(i,t)$ is defined as a random process representing the value of the backoff counter at time t, and $s(i,t)$ is defined as the random process representing the backoff stage j $(j = 0, 1, \ldots, L_{i,retry})$ where $L_{i,retry}$ is the retry limit for the priority AC_i class. The value of the backoff counter $b(i,t)$ is uniformly chosen in the range $(0, 1, \ldots, W_{i,j} - 1)$, where $W_{i,j}$ is defined in Equation (4.1).

$$W_{i,j} = \begin{cases} 2^j W_{i,0} & \text{for } j = 0, 1, \ldots, m_i - 1, \text{ if } L_{i,retry} > m_i \\ CW_{i,max} & \text{for } j = m_i \ldots, L_{i,retry}, \text{ if } L_{i,retry} > m_i \\ 2^j W_{i,0} & \text{for } j = 0, 1, \ldots, L_{i,retry}, \text{ if } L_{i,retry} \leq m_i \end{cases} \tag{4.1}$$
$$m_i = \log_2 \left(CW_{i,max} / CW_{i,min} \right)$$

Let p_i denote the probability that the transmitted frame collides and also that a station in the backoff stage for the priority AC_i class senses the channel busy. Therefore, the two-dimensional random process $\{s(i,t), b(i,t)\}$ can be modelled as a discrete-time Markov chain. Thus, the state of each STA in the AC_i is described by i, j, l, where j stands for the backoff stage and l stands for the backoff delay. The state transition diagram of AC_i is depicted in Figure 4.2.

Let $b_{i,j,l} = \lim_{t \to \infty} \{s(i,t) = j, b(i,t) = l\}$ be the stationary distribution of the Markov chain. In steady state, we have

$$b_{i,j,0} = p_i^j b_{i,0,0} \qquad 0 \leq j \leq L_{i,retry} \tag{4.2}$$

$$b_{i,j,l} = \frac{W_{i,j} - l}{W_{i,j}} \frac{1}{1 - p_i} B_{i,j,0} \qquad 0 \leq j \leq L_{i,retry},\ 1 \leq l \leq W_{i,j} - 1 \tag{4.3}$$

Also, by imposing the normalization condition for stationary distribution, we get

$$\sum_{j=0}^{L_{i,retry}} \sum_{l=0}^{W_{i,j}-1} b_{i,j,l} = 1 \tag{4.4}$$

Then, from Equations (4.2), (4.3) and (4.4), we can derive a generic approximation of the initial state for each ACs and is given by Equation (4.5).

$$\frac{1}{b_{i,0,0}} = \sum_{j=0}^{L_{i,retry}} \left[1 + \frac{1}{1 - p_i} \sum_{l=1}^{W_{i,j}-1} \frac{W_{i,j} - l}{W_{i,j}} \right] p_i^j \tag{4.5}$$

Let $p_{t,i}$ the probability that a station in the AC_i priority class transmits during a generic time slot. The following relations can be derived,

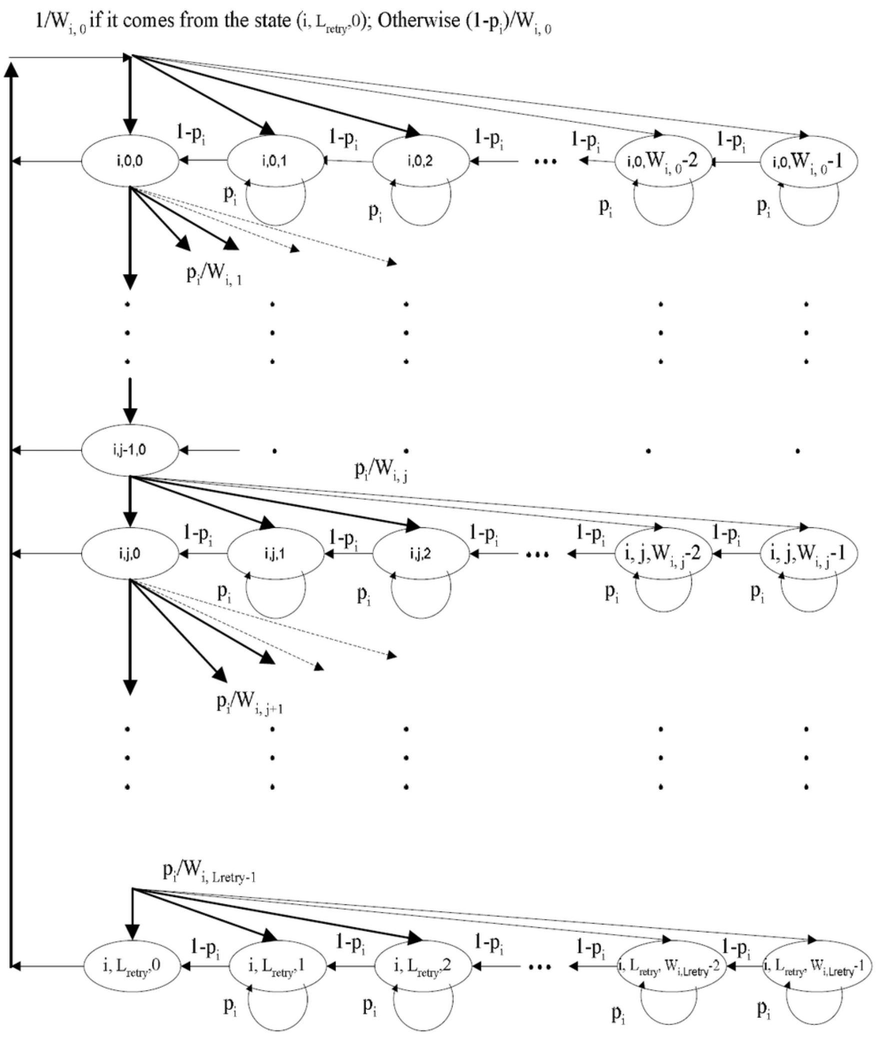


Figure 4.2: The state transition diagram for AC_i

$$p_{t,i} = \sum_{j=0}^{L_{i,retry}} b_{i,j,0} = b_{i,0,0} \frac{1 - p_i^{L_{i,retry}+1}}{1 - p_i} \tag{4.6}$$

$$p_i = 1 - \left(\prod_{h=0}^{i-1} (1 - p_{t,h})^{n_h} \right) (1 - p_{t,i})^{n_i - 1} \left(\prod_{h=i+1}^{N-1} (1 - p_{t,h})^{n_h} \right) \tag{4.7}$$

The Equations (4.5), (4.7) & (4.6) represent a non-linear system of equations with unknowns $b_{i,0,0}$, $p_{t,i}$, and p_i, which can be solved by numerical results using any numerical computing environment such as MATLAB [115].

4.2.1 Saturation Throughput

The probability that the channel is busy, p_b, can be written as

$$p_b = 1 - \prod_{h=0}^{N-1} (1 - p_{t,h})^{n_h} \tag{4.8}$$

Let $p_{s,i}$ denote the probability that a successful transmission occurs in a time slot for the AC_i, and let p_s denote the probability that a successful transmission occurs in a time slot in general. So, we have

$$p_{s,i} = n_i p_{t,i}(1 - p_{t,i})^{n_i - 1} \prod_{h=0,h\neq i}^{N-1} (1 - t_h)^{n_h} \tag{4.9}$$

$$p_s = \sum_{i=0}^{N-1} p_{s,i} = \sum_{i=0}^{N-1} \frac{n_i p_{t,i}}{1 - p_{t,i}}(1 - p_b) \tag{4.10}$$

The average length of a time slot comprises of idle period (I), successful transmission period (I) and collision period (T_c). Let T_{slot}, $T_{E(L_{packet})}$, $T_{E(L_{packet^*})}$, T_H and δ be the duration of a time slot, the time to transmit the average payload, the time to transmit the payload of the longest frame involved in a collision, the time to transmit the MAC & PHY headers plus any pads or tails, and the time of the propagation delay, respectively. So the mean I, mean T_s and mean T_c can be derived as

$$E(I) = (1 - p_b)T_{slot} \tag{4.11}$$

$$\begin{aligned} E(T_s) &= p_s T_s \\ &= p_s(T_H + T_{E(L_{packet})} + SIFS + \delta + T_{ACK} + DIFS + \delta) \end{aligned} \tag{4.12}$$

$$E(T_c) = (p_b - p_s)T_c = (p_b - p_s)(T_H + T_{E(L_{packet^*})} + DIFS + \delta) \tag{4.13}$$

According to the Equations (4.8)-(4.13), the normalized saturation throughput, S_i, for an AC_i can be written as

$$\begin{aligned} S_i &= \frac{E(\text{payload successful transmission time of } AC_i)}{E(\text{total period between two successive transmission})} \\ &= \frac{p_{s,i} T_{E(L_{packet})}}{E(I) + E(T_s) + E(T_c)} \end{aligned} \tag{4.14}$$

4.2.2 Saturation Delay

According to Xiao, the saturation delay for an AC_i, denoted here as D_i, is the average delay under the saturation condition of a class i priority, and includes the medium access delay, transmission delay, and any IFS.

Let X_i $(i = 0, \ldots, N-1)$ denote the r.v. representing the total number of backoff slots a packet a priority i class experiences without backoff counter freezes. Similarly, let X_i $(i = 0, \ldots, N-1)$ the r.v. of the total number of times that the queue of AC_i senses the medium busy before its backoff timer reaches zero. In other words, the total number of times that the counter freezes for a frame of the priority i class. Let $N_{i,retry}$ $(i = 0, \ldots, N-1)$ denote the r.v. of the total number of retries for the priority i class. Consequently, we can derive the following mean values

$$E(X_i) = \sum_{j=0}^{L_{i,retry}} \frac{p_i^j (1 - p_i)}{1 - p_i^{L_{i,retry}+1}} \sum_{h=0}^{j} \frac{W_{i,h} - 1}{2} \tag{4.15}$$

$$E(B_i) = \frac{E(X_i)}{(1 - p_i)} p_i \tag{4.16}$$

$$E(N_{i,retry}) = \sum_{j=0}^{L_{i,retry}} \frac{j p_i^j (1 - p_i)}{1 - p_i^{L_{i,retry}+1}} \tag{4.17}$$

Let T_o denote the time that a station has to wait when its frame transmission has failed before it can sense the channel again and let $T_{ACK_{Timeout}}$ express the interval of which a STA has to wait before sensing the channel again or before assuming a packet collision occurred.

$$T_o = SIFS + T_{ACK_{Timeout}} \tag{4.18}$$

Given the Equations (4.15)-(4.18), the average saturation delay of an AC_i queue can be calculated from the following

$$E(D_i) = E(X_i)\delta + E(B_i)\left[E(T_s) - E(T_c)\right] + E(N_{i,retry})(T_c + T_o) + T_s \tag{4.19}$$

4.3 Numerical Results Using MATLAB

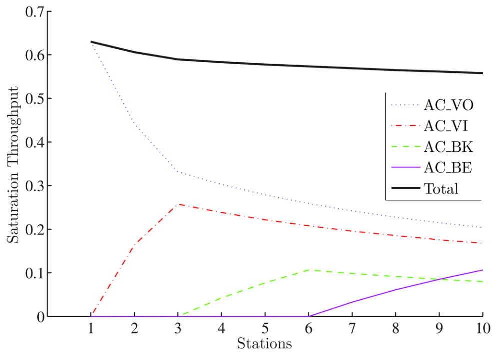
We conduct analytical results to evaluate the performance of the 802.11e EDCA medium access mechanism in terms of saturation throughput and access delay. More specifically, we would like to gain a better understanding on how the behaviour of the QoS prioritization of high priority ACs over lower priority effect the overall performance, which sometimes could lead into starvation for the lower priority ACs. We use the IEEE 802.11a as an example and its parameters can be found in [6, 11]. The data rate is 54 Mb/s and the control rate is kept at 6 Mb/s. The packet lengths are fixed at $1,500$ bytes for each STA and the number of STAs varies according to the scenario in consideration. For simplicity reasons the RTS/CTS frame exchange as a collision protection mechanism has not been utilized as the derived objective conclusions are analogous, in any case it can be easily applied.

For our results we defined three numerical experiments where each STA deploys one backoff entity of one and only AC to contend for the channel. Since we can't vary the packet length or the time slot for individual STAs in order to increase the offered load of a specific priority class, we add more STAs of that AC. For the first scenario, there are ten (10) wireless STAs that comprise of one backoff entity per AC_VO (STA1), two backoff entities for AC_VI (STA2 and

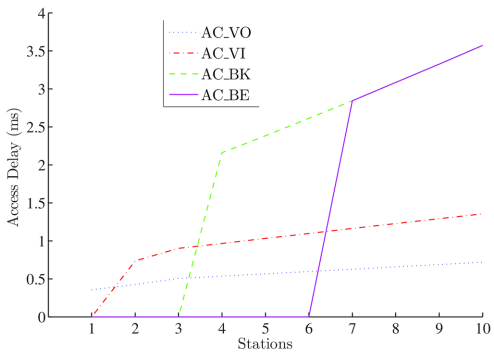
STA3), three per *AC_BK* (STA4, STA5 and STA6) and four per *AC_BE* (STA7, STA8, STA9 and STA10). For the second scenario, there are again ten (10) STAs but a reverse approach is followed, instead of increasing the lower priorities, now we increase the higher priorities. So, the second scenario includes one backoff entity per *AC_BE* (STA1), two backoff entities for *AC_BK* (STA2 and STA3), three per *AC_VI* (STA4, STA5 and STA6) and four per *AC_VO* (STA7, STA8, STA9 and STA10). The third scenario has a totally different set-up. We employ thirty (30) STAs of BE traffic that constantly transmit throughout the simulation run and alongside ten backoff entities for *AC_VO* (STA1 to STA0) are gradually imported. Finally, for all three (3) scenarios, we perform ten (10) separate simulation steps where we add to the system progressively STA1 to STA10, one by one, and collect individual performance measurements.

Figure 4.3 shows the collected results for saturation throughput and expected mean access delay for each AC as non-QoS STAs increase. It can be observed that the total throughput (Figure 4.3a) for all participated ACs tends to a constant value around ~ 0.6 of the normalised data rate. This again verifies the tendency of the poor channel efficiency, already high lighted in previous sections. As more STAs are included in the results, the saturation throughputs of each AC is affected analogously. Even though the majority of the wireless nodes belong to the lower priority classes, the saturation throughputs for the higher priorities yield graphs higher than the lower ACs, despite being less. For example, although the single STA for the *AC_VO* backoff entity is effected from the increasing offered loads, it uses the channel more often. This behaviour is also derived from Figure 4.3b where the real-time applications satisfy their delay requirements, in contrast to heavy profiled traffic flows with no QoS demands which expect higher delay. More specific, the mean access delay for STA1 that is comprised with *AC_VO* traffic flows remains around ~ 0.5 ms throughout the simulation runs, while the subsequent AC, the *AC_VI*, has a small incline every time a new STA is introduced to the system. Furthermore, the commencing mean delay for the *AC_BE* is six (6) times more than the highest priority AC.

On the other hand, in Figure 4.4 we change the ratio of non-QoS and QoS STAs by introducing more of the latter. Again, for the saturation throughput results (Figure 4.4a), we observe an influential attitude towards the higher pri-

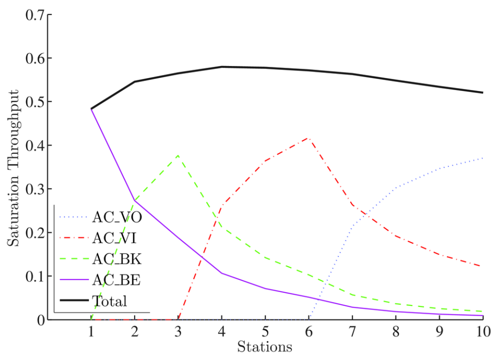


(a) Saturation Throughput vs. Stations

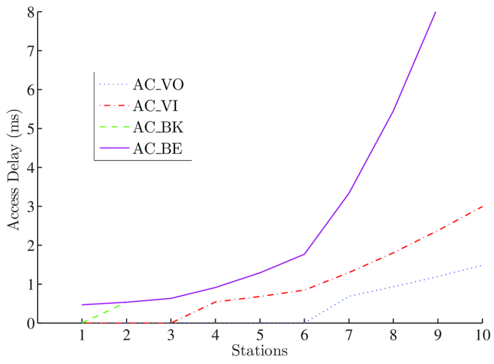


(b) Access Delay vs. Stations

Figure 4.3: EDCA performance measurements as non-QoS STAs increase



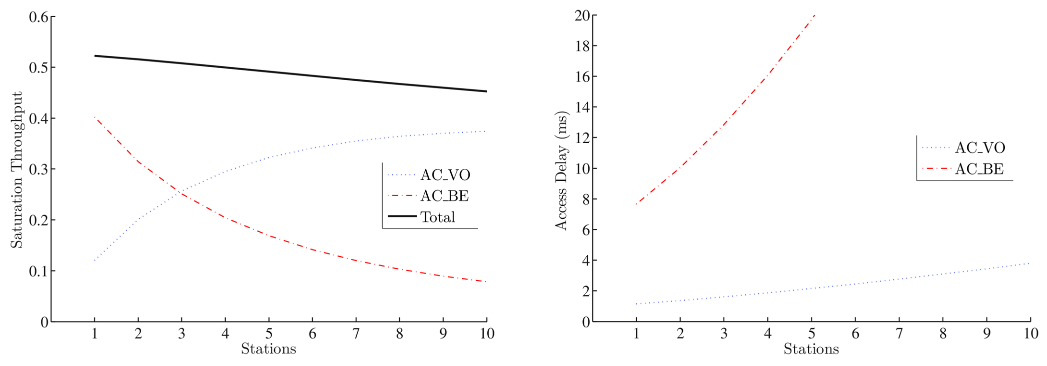
(a) Saturation Throughput vs. Stations



(b) Access Delay vs. Stations

Figure 4.4: EDCA performance measurements as QoS STAs increasing

ority traffic since they 'steal' bandwidth from the lower priorities. Even though there are instances where the non-QoS flows show peaks of transmission, these are immediately dropped once higher ACs are introduced over the subsequent simulation steps. At the beginning of the simulation, the single STA of AC_BE attains a normalized throughput of 0.4833. As newly arrived traffic (or STAs) enter the network the throughput for the BE application rapidly decreases with an exponentially rate and at the last simulation run achieved only 0.0283. The prioritization mechanism even effects the channel access delay as for the AC_BE we observe a rapid increase while more QoS delay bounded flows enter the system. Even though the number of AC_VO backoff entities prevail in this scenario, they hold the least channel access delay (Figure 4.4b).



(a) Saturation Throughput vs. Stations (b) Access Delay vs. Stations

Figure 4.5: QoS STAs get more greedy

In order to demonstrate the magnitude of this unfairness over the lower ACs, for this third scenario we have introduced heavy-load traffic conditions of AC_BE traffic. So, thirty (30) STAs constantly contend the channel and after are granted access permission, they transmit a payload size of $1,500$ bytes. In addition, at every simulation step, a QoS provisioned STA of AC_VO traffic will be introduced in the system. Figure 4.5 shows both saturation throughput and channel access delay of this scenario. From the saturation throughput (Figure 4.5a), it can be observed that as the first AC_VI backoff entity enters the system, the best effort flows maintain a higher portion of the channel's bandwidth. Yet again, the delay results (Figure 4.5a) indicate that there is a huge burden over the low backoff entities as from the start the BE traffic flows count large values of delay which rapidly increases as the simulation runs progress. Also, for the third QoS provisioned STA that enters the network we observe a cross-over point for the ACs normalized throughputs, where the most prioritized traffic begins to absorb more bandwidth. Note that at that point, the number of STAs for the low priority AC are thirty (30) while the high priority AC STAs are only three (3). On the other hand the channel access delay for the delay bounded traffic remains in low values throughout the simulation. The EDCA through the prioritization process starves AC_BE STAs in order to serve AC_VO STAs that have delivery time boundaries.

4.4 Simulations Using OPNET

In addition, we argue that frame aggregation adheres due to the EDCA scheduler's priority mechanism from IEEE 802.11e, resulting in the network's poor overall performance. Although this situation can induce unfairness to the lower ACs, this is the most adequate mechanism for the higher ACs to attain channel access within the delay-constraints appointed from the originated application. But, as the waiting period is decreasing, so are the number of packets that trail the first arrived packet, consequently the aggregate size is small. As we described in the previous chapter, there is a trade-off of choice that has to be taken into account when we want to improve network performance, minimizing the waiting time or maximizing utilization or efficiency. So, far we show that in order to minimize the access channel waiting periods, we impose a burden to the best-effort applications which also compromise most of network's traffic. In this section, we will show how 802.11e's EDCA function decreases the efficiency of the higher priority transmission queues over the IEEE 802.11n environment.

We also evaluate closely the performance of EDCA through various simulations using OPNET. The design and choice of network architecture for each scenario corresponds to a home, a large enterprise and a hot spot environment. Therefore, for the scenarios' configuration and layout, we follow, as before, the 802.11n usage *Scenario 1*, *Scenario 4* and *Scenario 6* from TGn's *Usage Models* document [68]. A detailed description for a set of the case scenarios which we are going to use throughout this section have also been given in the Appendix Usage Models. The following set of standard performance metrics are collected: the total goodput for the WLAN and for each individual flow (in Mb/s), the average aggregated sizes, the maximum and average latency values for every AC (in sec), and the Packet Loss Rate (PLR), but only for the QoS flows since PLR is defined as the percentage of packets that have not been delivered within the allowed maximal delay set by the QoS bounded originated application. Something that may be out of scope for the time being but will prove significant on the following chapters, for all the scenarios the TCP *New Reno* flavour is used and the receiver's TCP window buffer is set at $655,350$ bytes.

4.4.1 Residential Scenario

The first scenario represents an indoor (room to room) residential network with several HT devices. A total of twelve (12) HT wireless nodes are spread over a residential platform, eleven (11) STAs plus a single QoS AP. Distinguished examples of application usage, is the viewing of Standard-Definition television (SDTV) and HDTV anywhere in the house and simultaneous talk on VoIP telephones, surfing the Internet or listening to MP3 music that is stored on a central wireless unit or even playing games on-line via wireless consoles. A further study of the UPs used in this scenario, will show that the majority of the applications are real-time, thus delay bounded. There are only two BE flows, an Internet and Local file transfer applications of OL of 1 and 30 Mb/s, respectively. The simulation is run for an adequate enough time and results are gathered after the first second passes in order to allow time for the system to establish communication links, e.g. TCP slow start.

From	To	UP	Offered Load	Goodput	Average Aggregate	Maximal Delay	Average Delay	PLR
STA 0	STA 1	5	19.2	19.041	21.89	0.08823	0.03082	0
STA 0	STA 3	5	24	23.679	23.49	0.09434	0.03295	0
STA 0	STA 4	5	4	3.978	13.41	0.28929	0.04302	0
STA 0	STA 4	0	1	0.991	13.41	0.28929	0.04302	N/A
STA 0	STA 7	7	0.096	0.096	1.01	0.01264	0.00248	0
STA 0	STA 8	7	0.096	0.096	1.01	0.01274	0.00261	0
STA 0	STA 9	7	0.096	0.096	1.01	0.01284	0.00275	0
STA 0	STA 10	5	2	1.989	11.91	0.08563	0.02228	0
STA 0	STA 11	5	0.128	0.127	1.85	0.08516	0.02732	0
STA 1	STA 0	5	0.06	0.06	1.71	0.05398	0.00924	0
STA 3	STA 0	5	0.06	0.06	1.72	0.05603	0.00911	0
STA 4	STA 10	0	30	6.366	40.04	1.14062	0.68815	N/A
STA 5	STA 6	5	0.5	0.498	2.25	0.06179	0.01326	0
STA 6	STA 5	5	0.5	0.499	2.15	0.05688	0.01148	0
STA 7	STA 0	7	0.096	0.096	1.02	0.01527	0.00323	0
STA 8	STA 0	7	0.096	0.096	1.01	0.01495	0.00322	0
STA 9	STA 0	7	0.096	0.096	1.01	0.01526	0.00327	0
STA 10	STA 0	7	1	1	1.18	0.01168	0.00224	0
STA 11	STA 10	7	0.5	0.5	2.25	0.01648	0.00187	0
AC_BE			83.524	59.362	26.73	0.71495	0.36558	N/A
AC_VI					8.93	0.09681	0.02217	0
AC_VO					1.19	0.01398	0.00271	0

Table 4.2: Detailed simulation results of 802.11n for Scenario 1

Table 4.2 lists the simulation results for the Residential case. The total offered load from all applications is 83.524 Mb/s, while the network delivers a Goodput of 59.362 Mb/s. Consequently, we observe that the scenario accomplishes around 71% of the total offered load. Regarding the aggregation mechanisms, the frames concatenate on average 26.73, 8.93, 1.19 packets for the AC_BE, AC_VI and AC_VO, respectively. This validates our previous argument that higher priority flows access the channel in shorter periods, therefore their aggregates are smaller, e.g. see $AC\_VO \mapsto 1.19$. Nevertheless, all delay requirements are met and there are no packets lost due to network's restraints.

4.4.2 Large Enterprise Scenario

Briefly, the *Scenario 4* from the usage models document contains one (1) AP and thirty (30) associated STAs. The mixture of applications varies from internet and local file transfers, video conferencing, VoIP to some media player usage. The range of applications in this scenario share both high and low priority ACs but with the best-effort flows calling for a high bandwidth demand. In a nutshell, it captures the users' daily peak activity of some company's wireless network domain.

Table 4.3 lists the simulation results for the Large Enterprise case. The total offered load from all applications is 460.176 Mb/s, while the network delivers a Goodput of 62.061 Mb/s. Consequently, we observe that the scenario accomplishes around 13.5% of the total offered load. Regarding the aggregation mechanisms, the frames concatenate on average 42.95, 2.66, 1.13 packets for the AC_BE, AC_VI and AC_VO, respectively. Again, these results confirm the issue that emerges when the enhancements of 802.11n are utilized with the QoS mechanisms in conjunction. From individual results we derive the conclusion that only a very small portion of the requested bandwidth from the best-effort applications was served. Nevertheless, all delay requirements are met and the packets lost due to network's restraints is negligible (PLR for VO = 0.2%).

4.4.3 Hot Spot Scenario

On the other hand, *Scenario 6* has forty-one (41) STAs and one (1) AP, all present within AP's range. The configuration is arbitrary like most hot-spot networks are and the traffic applicable is VoIP, high and medium quality audio with video streaming, SDTV broadcasting and Internet File (IF) transfers. The bandwidth requests are not excessively large as we show with previous scenarios and the offered load is equally distributed between best-effort and real-time applications, if not the latter retain a slight advantage. Most of the applications, such as voice

From	To	UP	Offered Load	Goodput	Average Aggregate	Maximal Delay	Average Delay	PLR
STA 0	STA 1	0	1	0.798	61.41	1.64487	1.04880	N/A
STA 0	STA 2	0	1	0.806	56.94	1.63768	1.04401	N/A
STA 0	STA 3	0	1	0.810	62.84	1.62690	1.06538	N/A
STA 0	STA 4	0	1	0.791	52.74	1.62377	1.06626	N/A
STA 0	STA 5	0	1	0.923	63.47	1.63517	0.88424	N/A
STA 0	STA 6	0	10	4.358	55.38	1.63434	0.78534	N/A
STA 0	STA 7	5	1	1	2.1	0.05037	0.00711	0
STA 0	STA 8	5	1	1	2.23	0.05099	0.00773	0
STA 0	STA 9	5	2	2	2.93	0.03936	0.00639	0
STA 0	STA 10	5	2	2	3.15	0.04341	0.00689	0
STA 0	STA 11	0	30	2.808	39	1.43707	1.02419	N/A
STA 0	STA 12	0	30	2.76	43.81	1.62735	0.98273	N/A
STA 0	STA 13	0	30	0.627	29.86	1.32756	1.00657	N/A
STA 0	STA 14	0	30	3.198	48.45	1.64367	1.07792	N/A
STA 0	STA 15	0	30	3.678	43.79	1.63651	0.95851	N/A
STA 0	STA 16	0	30	1.638	42	1.31235	0.95273	N/A
STA 0	STA 17	0	30	2.742	45.7	1.63142	0.96771	N/A
STA 0	STA 18	0	30	2.139	41.94	1.27320	0.93958	N/A
STA 0	STA 19	0	30	2.469	43.32	1.64988	1.02852	N/A
STA 0	STA 20	0	30	3.126	45.3	1.44636	0.89745	N/A
STA 0	STA 25	7	0.096	0.096	1.06	0.01905	0.00414	0
STA 0	STA 26	7	0.096	0.096	1.06	0.01918	0.00423	0
STA 0	STA 27	7	0.096	0.096	1.06	0.01939	0.00434	0
STA 0	STA 28	7	0.096	0.096	1.07	0.01959	0.00445	0
STA 0	STA 29	7	0.096	0.096	1.08	0.02425	0.00474	0
STA 0	STA 30	7	0.096	0.096	1.1	0.02447	0.00508	0
STA 1	STA 0	0	0.256	0.308	22.89	0.22494	0.04354	N/A
STA 2	STA 0	0	0.256	0.309	23.79	0.16898	0.04057	N/A
STA 3	STA 0	0	0.256	0.305	23.47	0.15510	0.04087	N/A
STA 4	STA 0	0	5	3.786	34.34	0.15084	0.04751	N/A
STA 5	STA 0	0	10	4.651	55	0.53720	0.13319	N/A
STA 6	STA 0	0	0.256	0.421	29.49	0.27016	0.05749	N/A
STA 7	STA 0	5	1	1	2.71	0.05279	0.00874	0
STA 8	STA 0	5	1	1	2.82	0.05128	0.00912	0
STA 21	STA 0	0	30	0.942	34.89	0.14798	0.09065	N/A
STA 22	STA 0	0	30	3.42	38	0.28590	0.09713	N/A
STA 23	STA 0	0	30	3.567	39.63	0.16145	0.07232	N/A
STA 24	STA 0	0	30	1.53	39.23	0.13310	0.09225	N/A
STA 25	STA 0	7	0.096	0.096	1.18	0.02701	0.00567	0
STA 26	STA 0	7	0.096	0.096	1.17	0.02701	0.00606	0
STA 27	STA 0	7	0.096	0.096	1.2	0.03699	0.00598	0
STA 28	STA 0	7	0.096	0.096	1.21	0.02732	0.00612	0
STA 29	STA 0	7	0.096	0.096	1.19	0.03731	0.00598	1
STA 30	STA 0	7	0.096	0.096	1.18	0.03262	0.00590	1
AC_BE			460.176	62.061	42.95	1.03937	0.63252	N/A
AC_VI					2.66	0.04803	0.00766	0
AC_VO					1.13	0.02618	0.00522	0.2

Table 4.3: Detailed simulation results of 802.11n for Scenario 4

and video traffic, is transmitted using UDP and only for the IF transfers we do apply the TCP protocol.

In general, real-time video and audio streaming applications are designed to be more persistent to occasional lost packets, thus UDP is a more suitable and flexible protocol suite to use. But, in order to support UDP-based real-time applications over the Internet, it is necessary to provide bandwidth to the UDP applications within the network so that the performance of the UDP applications will not be seriously affected during periods of congestion. UDP flows do not typically back off when they encounter congestion, but aggressively use up more bandwidth than TCP friendly flows [116]. This scenario has a twofold importance,

From	To	UP	Offered Load	Goodput	Average Aggregate	Maximal Delay	Average Delay	PLR
STA 0	STA 1	0	2	0.769	53.42	2.45539	1.14827	N/A
STA 0	STA 2	0	2	0.770	55.78	2.45565	1.11628	N/A
STA 0	STA 3	0	2	0.764	55.35	2.44974	1.10791	N/A
STA 0	STA 4	0	2	0.737	55.82	2.49027	1.13431	N/A
STA 0	STA 5	0	2	0.737	53.43	2.49131	1.15636	N/A
STA 0	STA 6	0	2	0.766	53.17	2.50471	1.18454	N/A
STA 0	STA 7	0	2	0.769	53.38	2.45179	1.18854	N/A
STA 0	STA 8	0	2	0.760	55.09	2.44876	1.11164	N/A
STA 0	STA 9	0	2	0.753	52.29	2.46219	1.10786	N/A
STA 0	STA 10	0	2	1.032	49.57	1.60842	0.86487	N/A
STA 0	STA 11	5	2	1.792	43.75	0.46845	0.21803	52
STA 0	STA 12	5	2	1.775	43.33	0.43233	0.21443	50
STA 0	STA 13	5	2	1.783	38.69	0.47600	0.22433	54
STA 0	STA 14	5	2	1.814	37.68	0.49668	0.22221	53
STA 0	STA 15	5	8	7.009	48.55	0.50869	0.26744	71
STA 0	STA 16	5	8	7.022	48.63	0.51228	0.26655	71
STA 0	STA 17	5	8	7.004	49.21	0.51202	0.26559	69
STA 0	STA 18	5	5	4.470	25.69	0.48250	0.23943	59
STA 0	STA 19	5	5	4.476	26.64	0.49225	0.23176	54
STA 0	STA 20	7	0.096	0.096	1.52	0.05023	0.00920	3
STA 0	STA 21	7	0.096	0.096	1.28	0.03668	0.00669	1
STA 0	STA 22	7	0.096	0.096	1.28	0.03680	0.00680	1
STA 0	STA 23	7	0.096	0.096	1.29	0.03691	0.00690	1
STA 0	STA 24	7	0.096	0.096	1.39	0.03830	0.00781	1
STA 0	STA 25	7	0.096	0.096	1.39	0.03852	0.00791	1
STA 0	STA 26	7	0.096	0.096	1.3	0.03703	0.00701	1
STA 0	STA 27	7	0.096	0.096	1.32	0.03714	0.00710	1
STA 0	STA 28	7	0.096	0.096	1.34	0.03725	0.00719	1
STA 0	STA 29	7	0.096	0.096	1.35	0.03737	0.00732	1
STA 0	STA 30	7	0.096	0.096	1.4	0.04549	0.00826	2
STA 0	STA 31	7	0.096	0.096	1.35	0.03748	0.00744	1
STA 0	STA 32	7	0.096	0.096	1.36	0.03764	0.00755	1
STA 0	STA 33	7	0.096	0.096	1.38	0.03786	0.00766	1
STA 0	STA 34	7	0.096	0.096	1.38	0.03808	0.00778	1
STA 20	STA 0	7	0.096	0.096	1.56	0.06600	0.01026	3
STA 21	STA 0	7	0.096	0.096	1.55	0.06351	0.01007	3
STA 22	STA 0	7	0.096	0.096	1.53	0.05985	0.01026	3
STA 23	STA 0	7	0.096	0.096	1.38	0.04639	0.00842	1
STA 24	STA 0	7	0.096	0.096	1.45	0.04957	0.00905	2
STA 25	STA 0	7	0.096	0.096	1.51	0.03713	0.00957	2
STA 26	STA 0	7	0.096	0.096	1.56	0.06379	0.00999	2
STA 27	STA 0	7	0.096	0.096	1.51	0.04598	0.01000	3
STA 28	STA 0	7	0.096	0.096	1.52	0.04480	0.00986	3
STA 29	STA 0	7	0.096	0.096	1.54	0.06292	0.00984	3
STA 30	STA 0	7	0.096	0.096	1.61	0.07350	0.01082	5
STA 31	STA 0	7	0.096	0.096	1.58	0.07546	0.01073	4
STA 32	STA 0	7	0.096	0.096	1.54	0.04675	0.00977	3
STA 33	STA 0	7	0.096	0.096	1.58	0.05420	0.01037	4
STA 34	STA 0	7	0.096	0.096	1.32	0.03623	0.00738	1
AC_BE			64.88	47.879	53.73	2.38182	1.11206	N/A
AC_VI					40.24	0.48680	0.23886	59.2
AC_VO					1.44	0.04696	0.00888	2

Table 4.4: Detailed simulation results of 802.11n for Scenario 6

first to check the interaction between high and low ACs but also to understand the behaviour of UDP over TCP protocols over the Transport Layer.

Table 4.4 lists the simulation results for the Hot Spot case. The total offered load from all applications is 64.88 Mb/s, while the network delivers a Goodput of 47.879 Mb/s. Consequently, we observe that the scenario accomplishes around 64.3% of the total offered load. Regarding the aggregation mechanisms, the frames concatenate on average 53.73, 40.24, 1.44 packets for the AC_BE, AC_VI and AC_VO, respectively. Within this scenario, we observe a high number of concatenated frames for the AC_VI traffic. The reasoning behind this effect has to do with the delayed channel access for this AC caused by the immediate higher priority flow from it, the AC_VO. This natural delayed channel access causes the aggregation queues to form larger aggregates and this behaviour is the base for our delayed channel access algorithm described over the next chapter. However, since this delay cannot be controlled or dynamically adjusted, we observe a high PLR for the AC_VI traffic, $PLR_{VI} = 59.22\%$ and hence failure of their respective QoS requirements. On the other hand, the delay requirements for the AC_VO flows are met and the packets lost due to network's restraints is negligible with $PLR_{VO} = 2\%$.

4.5 Summary

The new IEEE 802.11n standard provides enough capacity to service immense offered loads. Nevertheless, the PHY enhancements are not sufficient to guarantee significant throughput performance. The principle of Frame Aggregation is to form larger frames for transmission by collecting multiple packets inside an aggregate buffer. In the interest to increase the aggregated size, there is a need of packets to be piled in the stack. However, we've demonstrated that the performance improvement potentials may be limited by the interaction between prioritized and parameterized channel access scheduling mechanisms defined for the QoS support and the enhanced asynchronous data service for the increased MAC efficiency. Mainly because, the waiting period for delay-sensitive traffic decreases, but so are the number of packets that trail the first arrived packet, consequently the aggregate size is small.

Also, within the analytical analysis of EDCA, we've pointed out that the prioritization process starves AC_BE STAs in order to serve AC_VO STAs that have delivery time boundaries. Consequently, not only the higher priority STAs will tend to have small aggregate sizes but also the lower priority STAs maintain a small channel access frequency. In most cases, frame aggregation adheres due to the EDCA scheduler's priority mechanism, resulting in the network's poor overall performance.

Chapter 5

Delayed Channel Access and the TCP Problem

The new IEEE 802.11n standard provides enough capacity to service immense offered loads. Nevertheless, the PHY enhancements are not sufficient to guarantee significant throughput performance. As indicated earlier, having a large amount of data in each aggregate buffer is crucial to achieve high channel utilization and MAC efficiency. In order to achieve that there are two approaches to follow: proactive, e.g. set a quorum, a minimum required number of packets before the queue request channel access and during that time it remains idle or reactive, e.g. once a STA requests permission to access the channel, a scheduling agent can determine if there is enough data in the queue and if not, delay it's channel access granting so it can let the buffer collect more packets.

On the other hand, traffic intensity is a considerable factor that affects the size of these bursts. Meaning the ratio between how often the packets arrive at the queue, towards the interval that has to wait before these packets can be served, has to be taken into consideration too. We also show that IEEE 802.11e's probabilistic prioritization mechanism for QoS provides shorter waiting intervals for high priority entities in relation to lower priority. Although this situation can induce unfairness to the lower ACs, this is the most adequate mechanism for the higher ACs to attain channel access within the delay-constraints appointed from the originated application. But, as the waiting period is decreasing, so are the number of packets that trail the first arrived packet in the queue, consequently small aggregates are formed.

Regarding the question of which viewpoint is more important, minimizing the waiting time or maximizing utilization, a trade-off choice has to be taken into account in respect of the overall system's performance and QoS. A specific batch collection rule it doesn't exist, indicating that each scenario is distinctive with its individual characteristics, exclusive requirements and particular behaviour. However, an algorithm to determine all the aforementioned aspects and dynamically adapt, in both proactive and reactive manner, specific scheduling parameters that directly interfere with the channel access delay and act alongside the system's distributed coordination function, could be a solution.

5.1 A Description of the DCA

The concept of implementing an archetype delayed channel access algorithm was first introduced by Liu and Stephens in [19] and it was designed to intentionally commence a further delay at the MAC layer in order to increase the number of packets that can be buffered at the AC's queue, resulting in increased network overall performance. A good measurement for each station's channel load is the channel access delay. The channel access delay for a frame arriving at the MAC is defined as the period from the time that the frame arrives at the front of the queue buffer till its successful transmission to the intended receiving STA, excluding the wireless propagation delay (depicted in Figure 5.1). However, a set of conditions need to be applied so that it can match the aggregated packet formation with the traffic burst within an appropriate time scale.

A traffic burst of a flow describes a sequence of packets bounded by the first and last packets [117]. The frame compound formed with two or more aggregated packets requires to be proportional to the channel load, given that traffic burst can be either high when the channel load is elevated or low when the channel load is minimal. Packetised traffic exhibits bursty and self-similar or fractal-like characteristics [118]. Data analysis collected from an Ethernet network, show that self-similar traffic typically intensifies as the number of active traffic sources increases, contrary to generally accepted argument that aggregate traffic becomes

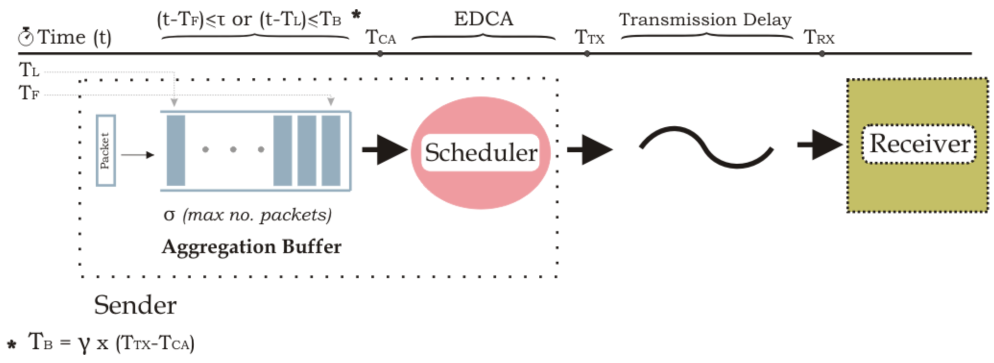


Figure 5.1: An illustration of the channel access delay

smoother (less bursty) as the number of traffic sources increase [77, 78, 79]. To keep track of the traffic burst, we identify T_B as the inter-arrival time of the traffic burst under review and satisfy two conditions *a*) the expected inter-arrival times of any two neighbouring packets belonging to the same burst is less than T_B; *b*) packets in different bursts satisfy a separation timing constraint. It is natural to use this flow characteristic as an initial condition to form an aggregate for each traffic burst.

Then, there is a robust association between aggregation length and waiting time, further waiting time signifies larger aggregate sizes. However, this situation might lead to unnecessary idling even when the packet queue isn't empty or the supplementary deferment may cause unpredictable issues to delay-bounded applications. The set appropriate time scale must identify the QoS requirements of the incoming traffic and foresees that the expected delivering times do not surpass the delay constraints. So, we set a positive constant, τ, which designates the maximal waiting time for a packet in the aggregation buffer. Note that, taking in consideration the time needed for a frame to be successfully received while setting τ, excess jitter can be avoided.

In an 802.11e WLAN, best-effort traffic is bounded by the large channel access waiting periods due to prioritization given to higher ACs. In addition, these non-delay constrained BE flows comprise a huge share of the total load and belong to bulk transfer applications, also known to be bandwidth-hungry (e.g. file-sharing and peer-to-peer applications) [119]. Regarding the transport protocol, the majority of the traffic is generated by TCP connections but it has been shown that the TCP throughput decreases in accordance to the amount of channel bandwidth occupied by the UDP traffic. Mathematical analysis and network observations show that in the saturated regime aggregate throughput obtained by the UDP flows can be more than the aggregate throughput achieved by the TCP persistent flows by a factor equal to the total number of UDP traffic flows [120]. This behaviour is independent of the QoS support mechanisms but is based over the upper layer transport protocol characteristics and their dynamics. Nevertheless, TCP flows are not delay bounded, neither does the traffic burst depend on the channel load since this is mainly occupied by the UDP real-time applications. Therefore, a different condition is set for its channel access delay, a constant number of packets to be formed in an aggregate. So, let σ the maximum number of packets in the aggregation buffer before aggregation is triggered.

5.2 The DCA Algorithm

The DCA algorithm maintains three attributes (see Table 5.1) and these are extremely important for the determination of the algorithm's decisions.

Parameter	**Description and Recommended Value**
γ	A positive constant that is the ratio of the inter-arrival time to the channel access delay. It is recommended that γ takes a value no less than 2. (Default = 10)
τ	A constant that is the maximal waiting time for packets in the aggregation buffer. For example, for a video flow with 200ms maximal delay, we recommend 100ms for low packet loss ratio requirement and 50ms for high packet loss ratio requirement. (Default = 'Low PLR')
σ	A constant that determines the maximal number of packets in the aggregation buffer before aggregation is triggered. (Default = 48).

Table 5.1: DCA algorithm's parameters

The algorithm delays the channel access as long as the number of packets in the aggregation buffer hasn't reached σ packets, or the period since the first packet that was received hasn't exceeded the maximal waiting time τ, or the duration from the last received packet remains below the time that was last needed to access the channel by a factor of γ.

The DCA scheduler follows this basic idea in two steps:

- It identifies the traffic burst by an inter-arrival time proportional to the last MAC channel access delay and puts all packets belonging to the corresponding traffic burst in an aggregation buffer.
- When the identified traffic burst is completed, it aggregates all the packets in the buffer in one aggregate for channel access and transmission.

In order for the scheduler to identify the traffic burst and control the triggering mechanism, each station needs to maintain a number of state variables for each AC. The following table (see Table 5.2) presents these variables, including their initial values, and a small description about their functionality. The state variables are also illustrated in Figure 5.1.

Variable	**Initial Value**	**Description**
T_B	10^{-6}	The inter-arrival time in seconds for the current burst.
N_{MSDU}	0	The number of packets from upper layer that are still in the aggregation buffer.
T_F	N/A	The arrival time of the first packet from upper layer in the current aggregation buffer.
T_L	N/A	The arrival time of the last packet from upper layer.
T_{CA}	N/A	The channel-access starting time for an aggregate.
T_{TX}	N/A	The transmission starting time for an aggregate.
t	N/A	The current time.

Table 5.2: DCA state variables

The pseudo-code below (labelled as Algorithm 1), describes in a manner of conditional and iteration consequence statements, the steps that the algorithm undertakes during its operation. These events also affect the values of the state variables, defined earlier. So, during the DCA operation, a *While-Loop* will repeatedly allow newly packets to tail the aggregate buffer queue till the boolean conditional trigger mechanisms are met. Note, that the iteration process is time and not packet dependant, meaning that DCA algorithm can remain in an "IDLE" state when there are no packet arrivals from the upper layer but yet it still carries conditional checks over the resting and total period of time.

Algorithm 1 DCA Pseudo-code

```
if $N_{MSDU} = 0$ then
    $T_F \leftarrow t$
else
    while $(t - T_F < \tau)$ or $(t - T_L < T_B)$ do                ▷ DCA triggers
        if new packet arrival then
            $N_{MSDU} \leftarrow N_{MSDU} + 1$                      ▷ form an aggregate
            $T_L \leftarrow t$
            if $(N_{MSDU} < \sigma)$ then                           ▷ DCA trigger
                break iteration
            end if
        end if
    end while
end if                                                              ▷ start channel access
$T_{CA} \leftarrow t$
$N_{MSDU} \leftarrow 0$
repeatWAIT
until ready to transmit
$T_{TX} = t$                                                        ▷ and transmit frame
repeatWAIT
until ACK or BA is received
$T_B \leftarrow \gamma(T_{TX} - T_{CA})$
```

5.3 Performance Evaluation of DCA

In this section, we evaluate closely the performance of DCA through various simulations using OPNET's model. The design and choice of network architecture for each scenario corresponds to a home, a large enterprise and a hot spot environment, *Scenario 1*, *Scenario 4* and *Scenario 6* from TGn's Usage Models document [68], respectively. For each scenario, a simulation is run with and without DCA and the channel is regarded as error-free. The simulation results are compared to determine the performance gain provided by DCA. In all the simulations, we collect the standard performance metrics, including the goodput for WLAN and each individual flow when applicable, the mean aggregated sizes of the frames, the latency values for the ACs, and the PLR for the QoS flows. Note that for all TCP traffic we are choosing the TCP New Reno extension [121] and the TCP receiver's advertised window size is set to $655,350$ bytes following the recommendations in [122] where it is suggested that the maximum TCP window size should be at least as large as the bandwidth-delay product of the wireless link. Last, we have disabled the A-MSDU aggregation mechanism and the QoS attributes for the DCA are set to $\gamma = 10$, $\tau \leq \frac{1}{2}$ maximal delay (e.g. $\tau_{BK\&BE} = 0.15$ ms, $\tau_{VI} = 0.1$ ms and $\tau_{VO} = 0.008$ ms), and $\sigma = 48$ packets.

In addition, for a home-based scenario, we choose to design a supplementary case which is still characterised as overloaded but yet is much simpler and consists of only four (4) STAs, as illustrated in Figure 5.2. We call this new scenario *Scenario 2* for future reference. So, for *Scenario 2* we consider an overloaded 802.11n WLAN that includes three (3) STAs and a single AP. All STAs are relatively close to each other and in Line of Sight (LoS). Their operational PHY rate is at 117 Mb/s since we've set a channel with 64-Quadrature Amplitude Modulation (QAM), a $\frac{3}{4}$ coding rate and 800 ns guard interval (see Modulation and Coding Scheme (MCS) index table for two spatial streams at 20 MHz in [13]).

Also, we set three different types of flows along with different sorts of protocol connections (UDP TCP). The IF task, as well as being supported by TCP protocol, has also been categorized as a BE AC. While the HDTVs are considered as VI and are streamed over UDP connections. The offered load, the source and

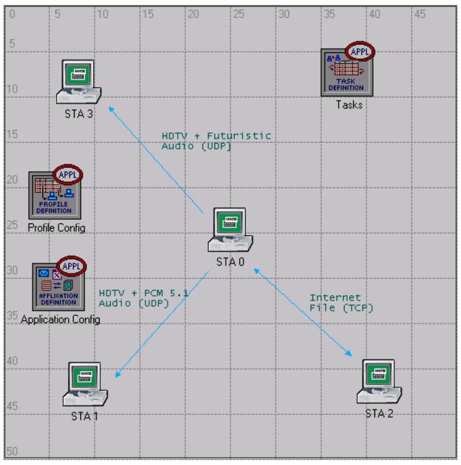


Figure 5.2: Spatial distribution in OPNET for custom Scenario 2

destination addresses, and other values are given in the in Table 5.3. Note that although the AP has not got any direct application requests towards STA2, there is in practise a transmission link to that wireless node since TCP is a bi-directional communication process.

STA Name	Role	Dest. STA	Mean Rate	Rate Distrib.	MSDU	Delay	Application
STA 0	HDTV + PCM 5.1 Audio	STA 1	24 Mb/s	Constant, UDP	1,500 B	200 ms	VoD Control Channel
STA 0	HDTV + Futuristic Audio	STA 3	19.2 Mb/s	Constant, UDP	1,500 B	200 ms	VoD Control Channel
STA 2	Internet File Transfer	STA 0	120 Mbps	Constant, TCP	1,500 B		P2P Downloading

Table 5.3: Role and configuration for each STA for custom Scenario 2

Table 5.4 and Table 5.5 show the results for the overloaded WLAN *Scenario 2* when DCA algorithm is disabled and enabled, respectively. Now, both HDTV average aggregated sizes have been increased dramatically, from 1.80 to 13.11 and from 1.31 to 12.21. The successfully received data over the total offered load has been increased from 33.7% (no DCA) to 58.2% (with DCA) as the goodput is increase from 54.987 Mb/s to 94.98 Mb/s, respectively. So, we observe that by deferring the channel access by introducing DCA, the end-to-end delays of both HDTV traffic flows have an insignificant increase and the maximum delays remain way below the 200 ms delay boundary. DCA doesn't override the AC's priority but limits the frequent channel accesses from high ACs to less and more efficient ones. It is obvious that the DCA algorithm has increased the system's performance to the point of ensuring a better channel utilization.

Name	Goodput (Mb/s)	Avg. Aggregate Size (MPDUs)	Max Delay (sec)	Avg. Delay (sec)	Max PLR (%))
HDTV	23.994	1.80	0.01267	0.00115	0
HDTV	19.197	1.31	0.01120	0.00099	0
Internet file transfer	11.796	24.57	0.65408	0.39693	N/A

Table 5.4: Numerical results for Scenario 2 w/o DCA

Name	Goodput (Mb/s)	Avg. Aggregate Size (MPDUs)	Max Delay (sec)	Avg. Delay (sec)	Max PLR (%))
HDTV	23.865	13.11	0.04457	0.01342	0
HDTV	19.116	12.21	0.04471	0.01520	0
Internet file transfer	51.999	25.27	0.13416	0.08866	N/A

Table 5.5: Numerical results for Scenario 2 with DCA

Table 5.6 shows the simulation results for *Scenario 1* when the DCA function is disabled and enabled for comparison reasons. It can be seen that the DCA improves the system's goodput from 59.362 Mb/s to 83.154 Mb/s. The introduction of DCA brought a massive increase of 40.08% in performance and the system's goodput over the total offered load went from 71.07% to 99.56%. In both cases the QoS requirements are met since the PLR results are at reasonable rates for all multimedia flows. For the case where the delayed channel operation is in operation, the PLR has increased slightly but the return gain of the system's overall performance shows significant results. Consequently, the trade-off between delaying channel access, that can also been seen in the delay results, to increase goodput pays out. Similar to previous simulation results, the aggregation sizes of the frames of the HDTV and VoIP traffic has increased respectively 8.93 $\rightarrow$ 13.66 and 1.19 $\rightarrow$ 2.71 as an effect of the further introduced delay before channel access. It is also noticeable that the overall performance has been increased because the BE traffic with the DCA function can utilize more resources and the goodput increases from 7.357 Mb/s to 30.831 Mb/s.

Table 5.7 displays the simulation results for *Scenario 4* with and without the DCA algorithm. Once more, the DCA improves the system's goodput from 62.061 Mb/s to 86.075 Mb/s, an increase of 38.7%. The system's goodput over the total offered load went from 13.6% to 18.7%. In spite of the ratio being so low, the overloaded enterprise-based scenario manages to utilizes more resources

Scenario 1		Off. Load		Goodput		Avg. Aggr.	Max. Delay	Avg. Delay	PLR
DCA Off	BE	83.524	31	59.362	7.357	26.73	0.71495	0.36558	N/A
	VI		50.448		49.93	8.93	0.09681	0.02217	0
	VO		2.076		2.075	1.19	0.01398	0.00271	0
DCA On	BE	83.524	31	83.154	30.883	27.98	0.12536	0.04525	N/A
	VI		50.448		50.195	13.66	0.10769	0.04451	2.33
	VO		2.076		2.076	2.72	0.01881	0.00684	4.86

Table 5.6: Numerical results for Scenario 1 with and w/o DCA

for the immense demand of the BE traffic, going from 52.909 *Mb/s* to 76.964 *Mb/s* (out of total 451.024 *Mb/s*) when DCA is enabled. Again, the average channel access delay has slightly increased as well but all the delay requirements for the multimedia traffic are met. It is evident that the growth of the packets in an aggregated frame for the VI flows has increased from 2.66 to 19.64 packets per frame. Last, the maximum observed PLR for the VI traffic has increased by 0.1% and is considered negligible as it remains well below the maximum permitted PLR of 5%. Again, DCA increases aggregate sizes for high priority flows and hence improves channel utilization.

Scenario 4		Off. Load		Goodput		Avg. Aggr.	Max. Delay	Avg. Delay	PLR
DCA Off	BE	460.176	451.024	62.061	52.909	42.95	1.03937	0.63252	N/A
	VI		8		8	2.66	0.04803	0.00766	0
	VO		1.152		1.1513	1.13	0.02618	0.00522	0.2
DCA On	BE	460.176	451.024	86.075	76.964	42.82	0.60900	0.38581	N/A
	VI		8		7.958	19.64	0.12937	0.05273	0
	VO		1.152		1.152	1.82	0.03404	0.00932	0.3

Table 5.7: Numerical results for Scenario 4 with and w/o DCA

The Hot Spot use case, known as *Scenario 6*, comprises of a large number of STAs that running mainly VoIP applications. This scenario is a typical case of resource starvation over low ACs caused by the higher ACs. This can be noticed from the simulation results given on Table 5.8, where the PLR for video flows, when DCA is not in operation, is 59.22% and hence badly fail their respective QoS requirements. Although the VoIP users do not require excessive bandwidth ($OL = 2.88$ *Mb/s*), the contention prioritized process for gaining access to channel from so many high AC users impacts the immediate below priority access class (here the AC_VI). On the other hand, when DCA is set in operation, the PLR for the VI cease to exist and the flows are served successfully without any jitter. Moreover, the BE traffic raises from 7.856 *Mb/s* to 15.966 *Mb/s* with an overall system performance increase at 26.54%. The DCA function improves the system's goodput from 47.878 *Mb/s* to 60.583 *Mb/s* and by considering that the total offered load is 64.88 Mb/s then the efficiency corresponds to 73.79% and 93.37% , respectively.

Scenario 6		Off. Load		Goodput		Avg. Aggr.	Max. Delay	Avg. Delay	PLR
DCA Off	BE	64.88	20	47.878	7.857	53.73	2.38182	1.11206	N/A
	VI		42		37.144	40.24	0.48680	0.23886	59.22
	VO		2.88		2.8783	1.44	0.04696	0.00863	2
DCA On	BE	64.88	20	60.583	15.966	58.13	0.84084	0.38268	N/A
	VI		42		41.738	27.26	0.11732	0.04036	0
	VO		2.88		2.878	2	0.05212	0.01091	3.4

Table 5.8: Numerical results for Scenario 6 with and w/o DCA

Based on the above results and the given analysis, we can argue that the operation of DCA resolves the negative performance impact which comes into being from the conjoining of EDCA and the new MAC enhancement of frame aggregation.

5.4 TCP Problem with DCA

Previously, during DCA's evaluation, the simulation runs for all the aforementioned scenarios were set with a TCP's window buffer size equal to 655, 350 bytes, following the recommendations in [122] where it is suggests that the maximum TCP window size should be at least as large as the bandwidth-delay product of the wireless link. In a nutshell, the window buffer defines the system's throughput, the amount of outstanding TCP data (unacknowledged by the recipient) that can remain in the network. Since the current scenarios comprises of HT STAs, in order to acquire theoretical performance measurements, it is wise to define a capacious buffer size for the TCP window (e.g. 655, 350 bytes) with the purpose that no result will be affected by it.

For actual TCP rwnd settings, *Microsoft (MS) Windows 98* has a default of 8, 192 bytes, *MS Windows 2000* has a default of 17, 520 bytes, *Linux* and *MS Windows XP* have a default of 65, 535 bytes, and for various hand-held devices the default window size varies depending on the installed mobile OS and the integrating circuit capabilities [123, 124, 125]. Next-generation TCP/Internet Protocol (IP) stacks in later *MS Windows* versions (*Vista and* 7) support TCP receiver window scaling option and no longer uses the *TCPWindowSize* registry value [126] but this is not always absolute. Note that over a WLAN, the TCP performance can also be affected by the link quality as the TCP operation may mistakenly determine a wireless error for congestion [127, 16]. Also, it has been studied that various buffer sizes of the STAs within a WLAN, individually STAs can cause unfairness issues and consequently play a key role in the overall throughput [128].

Going back to the performance evaluation of the DCA for *Scenario 2* and set as a more realistic TCP rwnd equal to $65,535$ bytes, we can derive the simulation results of Table 5.9. This table also includes the previously acquired results for TCP rwnd equal to $655,350$ bytes for comparison reasons. To clarify, we define as small rwnd and as large rwnd the buffer sizes of $65,535$ bytes and $655,350$ bytes, respectively.

DCA	TCP rwnd	Traffic	Offered Load	Goodput	Avg. Delay
Off	655,350	HDTV	19.2	19.197	0.00100
		HDTV	24	23.994	0.00115
		Internet File	120	11.796	0.39693
On	655,350	HDTV	19.2	19.116	0.01520
		HDTV	24	23.865	0.01342
		Internet File	120	51.999	0.08866
Off	65,535	HDTV	19.2	19.2	0.00087
		HDTV	24	23.997	0.00100
		Internet File	120	9.714	0.03194
On	65,535	HDTV	19.2	19.197	0.01520
		HDTV	24	23.988	0.01189
		Internet File	120	4.494	0.07948

Table 5.9: Numerical results for Scenario 2 for various TCP window sizes with and w/o DCA

So, Table 5.9 displays the simulation results for small and large rwnd when DCA is set as enabled and disabled. As an initial observation, we can determine that even with DCA disabled there is an impact of the varying rwnd over the IF traffic. The TCP communication over the MAC has been reduced from 11.796 Mb/s to 9.714 Mb/s, so by decreasing the rwnd to the default setting of MS Windows XP, the adverse impact for the IF goodput over the original value is -17.65%. Up until now, we have seen a great improvement over the network's overall performance when DCA is introduced. Nevertheless, for a small rwnd the DCA operation carries a negative effect as the goodput for the TCP traffic without applying channel access deferring is 9.714 Mb/s and with DCA declines at a diminutive value of 4.494 Mb/s. In conclusion, the reduction of the rwnd in conjunction with DCA functionality may bring unfavourable consequences. For the rest of the multimedia applications, in every case, both HDTV retain a goodput equal to the offered load with the maximal delay increasing slightly when DCA is applied for reasons that have already been explained.

Before we can explain very much about this negative behaviour of DCA with TCP traffic, firstly we need to describe briefly the TCP architecture as the origin of this contradictory reaction lies behind the TCP's protocol operation and calls.

5.5 A Brief Understanding of the TCP

The TCP [129, 130] is one of the core protocols of the IP Suite [131] and was first introduced by Cerf and Kahn in [132]. The TCP is a reliable, robust and connection-oriented method of data delivery and is commonly used over the Internet as it is well known for its flexibility since it adapts the transmission behaviour dynamically according to the network's disparate conditions [133]. Furthermore, it provides transparent segmentation and reassembly of user data and handles flow and congestion control. The main applications that usually employ the TCP protocol are the ones that emphasize reliability over reduced latency, such as WWW, email, remote administration and local or remote file transfers, P2P file sharing, etc. The form of data that passes over to the IP layer id known as segment and the Maximum Segment Size (MSS) is usual equal to the Maximum Transmission Unit (MTU) of the system's data link layer. During operation (see Figure 5.3), the protocol initiates various calls, such as to open and close connections or to send and receive data on established connections.

So, TCP is a connection-oriented protocol, hence when two STAs wish to communicate, first the receiver must bind to a port to open it up for incoming connections and then the sender commences a connection to that port. At that point, a three-way handshake occurs, a series of three calls take place for the connection to be established:

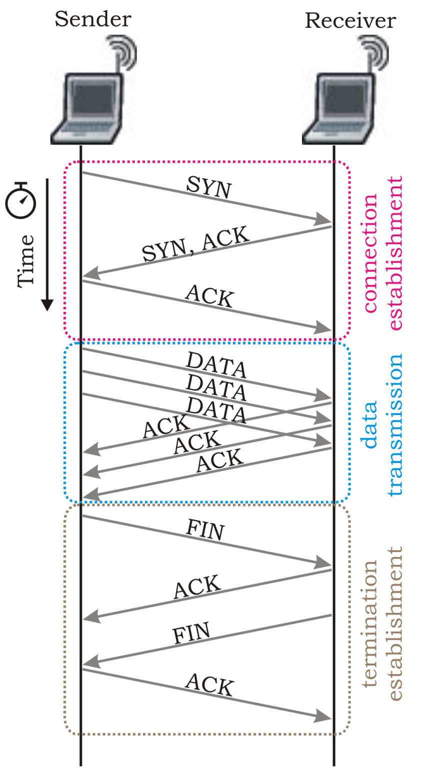


Figure 5.3: An illustration of a typical TCP connection

- The initiator of the session sends a segment with the Synchronize (SYN) flag set to the recipient.
- Upon receipt of the segment, the recipient sends a SYN segment to the initiator with the ACK number set to the sequence number increased by one, and sets a new sequence number for its own end.
- The initiator then sends an ACK of its own in response to the recipient's SYN with the ACK number set to the recipient's sequence number increased by one.

At this point, both the client and the server have received an acknowledgement of the connection establishment and communicating applications can transmit data between each other. Most of the discussion surrounding data transfer requires us to look at flow control and congestion control techniques which we discuss later in this section. Note that in Figure 5.3 the data transfer is illustrated with a bi-direction communication of *DATA* and *ACK* segments but in reality this does not depict an accurate operation of the TCP transfer but only appear for clarity, e.g. a single flow of the data from the sender to the receiver could be more than one segment. When their communication is complete, the connection is terminated or closed, in a similar way, to free the resources for other uses. The connection termination phase uses, at most, a four-way handshake, with each side of the connection terminating independently.

- The initiator of the close sends a Finalize (FIN) segment to the recipient.
- The recipient sends an ACK of the FIN segment.
- The recipient sends a FIN segment of its own to the initiator.
- The initiator responds with an ACK to that FIN segment.

Also, TCP provides reliability by recovering from packet loss using two mechanisms. Each segment is stamped with a sequence number so the end receiver can reply back with corresponding TCP ACK over the segments that it has successfully received, out of sequence packets are generating duplicate ACKs. So, the sender detects a loss when multiple duplicate ACKs arrive, implying that the next packet was lost. However, IP may reorder datagrams, thus TCP cannot immediately assume that all gaps in the packet sequence signify losses. When the session becomes idle or ACKs are lost, TCP detects losses using time-outs. Retransmission timers are continuously updated based on a weighted average of previous Round Trip Time (RTT) measurements. Accuracy is critical, since delayed time-outs slow down recovery, while early ones may lead to redundant retransmissions. So, if a TCP ACK is not received within a reasonable Retransmission Time-Out (RTO), then it will be assumed that the data was lost and a re-transmission will be initiated. Each RTO is computed after a new estimated RTT between the sender and receiver is specified, as well as the variance in this round trip time [134]. Note that ACKs arriving back at the sender arrive at intervals approximately equal to the intervals at which the data packets arrived at the sender.

Another characteristic of TCP is flow control. Flow control is a technique whose primary purpose is to properly match the transmission rate of sender to that of the receiver in order to avoid having the sender send data faster than what the receiver can handle. TCP uses a sliding window flow control protocol. The receiver specifies in every response a rwnd size, the amount of additionally received data that it is willing to buffer for the connection. The sending host can send segments only up to that amount of rwnd size before it must wait for an ACK and window update from the receiving host. If data queued by the sender reaches a point where data sent will exceed the receiver's advertised window size, the sender must halt transmission and wait for further acknowledgements and an advertised window size that is greater than zero before resuming.

Although flow control has similar diagnostics with congestion control, they are not the same since the latter's primarily concern is to sustain overloading the network. Congestion occurs when routers are overloaded with traffic that causes their queues to build up and eventually overflow, leading to high delays and packet losses. Therefore, when losses are detected, besides retransmitting the lost packet, TCP also reduces its transmission rate, allowing router queues to drain. Subsequently, it gradually increases its transmission rate so as to gently probe the network's capacity. In order to control the transmission rate, the protocol maintains a Congestion Window (cwnd), which is an estimate of the number of segments that can be in transmit without causing congestion. The initial size of the cwnd is usually a single segment of size up to the MSS but can also be more as mentioned in [135]. New segments are only sent if allowed by both this window and the receiver's advertised window. It is important for the transmission to be at a high enough rate to ensure good performance, but also to protect against overwhelming the network or receiving host. The way that the TCP protocol handles the cwnd depends on the implementation of the TCP variant but the typical protocol will be using one of the following algorithms, also defined in [136]: i) TCP Slow Start, ii) Congestion Avoidance, iii) Fast Retransmit, and iv) Fast Recovery.

The original TCP congestion avoidance algorithm [137] includes both *TCP Tahoe* and *TCP Reno* extensions. Since then, many other alternative algorithms have been introduced that engage the congestion in a more or less aggressive and systematic manner. Some of the most commonly used variants are *TCP Vegas* [138, 133], *FAST TCP*[139], *TCP New Reno* [121, 140, 141], *TCP Hybla* [142], *TCP CUBIC* [143], and *Compound TCP* [144].

Attribute	Value
Version/Flavour	New Reno
Maximum Segment Size (bytes)	Auto-Assigned
Receive Buffer (bytes)	4,096 - 655,350
Receive Buffer Adjustment	None
Receive Buffer Usage Threshold	0.0
Delayed ACK Mechanism	Segment/Clock Based
Maximum ACK Delay (sec)	0.200
Slow-Start Initial Count (MSS)	1
Fast Retransmit	Enabled
Duplicate ACK Threshold	3
Fast Recovery	New Reno
Window Scaling	Enabled
Selective ACK (SACK)	Disabled
ECN Capability	Disabled
Segment Send Threshold	Byte Boundary
Active Connection Threshold	Unlimited
Nagle Algorithm	Disabled
Karn's Algorithm	Enabled
Timestamp	Disabled
Initial Sequence Number	Auto Compute
Retransmission Thresholds	Attempts Based
Initial RTO (sec)	3.0
Minimum RTO (sec)	1.0
Maximum RTO (sec)	64
RTT Gain	0.125
Deviation Gain	0.25
RTT Deviation Coefficient	4.0
Timer Granularity (sec)	0.5
Persistence Timeout (sec)	1.0
Connection Information	Do Not Print

Table 5.10: TCP configuration for OPNET simulations

The *TCP New Reno* is the TCP extension of interest since this is being used for the TCP traffic for our simulation runs. For congestion control, the *TCP New Reno* applies the current mechanisms of *TCP Reno*, Slow Start, Congestion Avoidance and Fast Retransmit, and it modifies the Fast Recovery. During congestion avoidance, receipt of four back-to-back identical ACKs causes the sender to perform Fast Retransmit and to enter in a Fast Recovery mode. For the Fast Retransmit mechanism, the sender retransmits the lost segment, sets the slow start threshold (ssthresh) to half the current cwnd (cwnd/2) and sets cwnd equal to the new ssthresh size plus three (3) segments. Upon entering Fast Recovery, the sender continues to increase the cwnd by one (1) segment for each subsequent received duplicate ACK. Within the old *TCP Reno*, if the sender receives a non-duplicate ACK, it starts a window deflation and cancels Fast Recovery. The resolution of *TCP New Reno* corrects this behaviour by distinguishing between a "full" ACK and a "partial" ACK, depending if all or part of segments that were outstanding at the start of Fast Recovery have been acknowledged. Unlike *TCP Reno*, window deflation and congestion avoidance instance will only arise with the reception of a "full" ACK. Otherwise, the TCP protocol retransmits the segment next in sequence based on the "partial" ACK, and reduces the congestion window by one less (-1) than the number of segments acknowledged.

Note that for every simulation run, the selected TCP flavour for the OPNET scenarios is *TCP New Reno*, thus the wireless nodes that carry TCP traffic have been configured with the following common properties shown in Table 5.10. Most of the parameters are set by OPNET's default value, except "Window Scaling" and "Receiver Buffer" where respectively we have chosen to expand the window size in view of high bandwidth network and vary the receiver's buffer according to the scenario's characteristics or objectives. Information about the definitions and description of the listed TCP attributes can be gathered from OPNET's documentation [63], more specifically from the section labelled as *TCP Model User Guide*.

5.6 Cause of the TCP Problem

So now that we have a better understanding of the procedure that the TCP protocol follows and bearing in mind that the scheme was designated for single packet transmissions, we can go back to the previous *Scenario 2* and find the cause of the problem. We set new simulation runs for a range of TCP window sizes, beginning with a size of $4,096$ bytes and increase by a growth factor of two (2) till we reach $655,350$ bytes. Figure 5.4 and Figure 5.5 present the goodput and maximal delay results for the single TCP flow, the traffic generated by the IF application. Both graphs follow a similar pattern, starting from the beginning and up to a certain point we observe that by enabling the DCA algorithm, there is a negative impact over the performance measurements.

More precisely, at the beginning (TCP rwnd $= 4,096$ bytes) the goodput and peak delay results for the TCP traffic while DCA is disabled is equal to 0.666 Mb/s and 0.09950 sec, respectively. When DCA is activated, the goodput slightly drops to 0.636 Mb/s but the delay rises to 0.16677 sec. This negative behaviour is visible even at the aforementioned low rwnd (TCP rwnd $= 65,535$

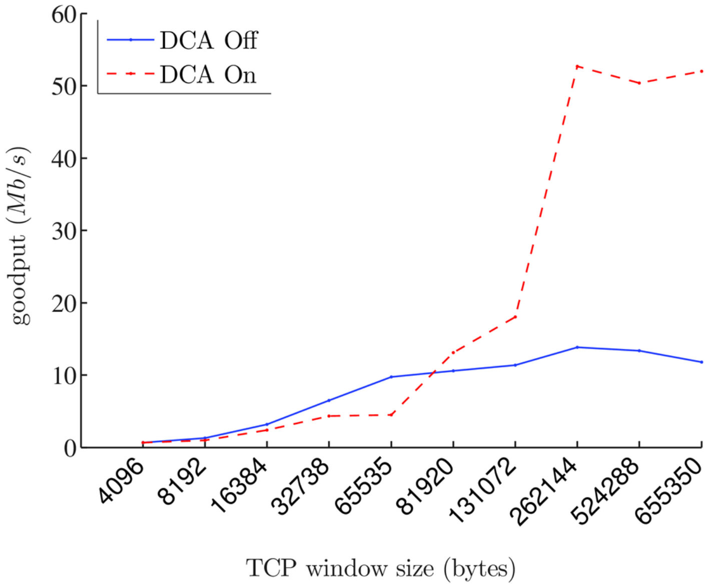


Figure 5.4: Goodput for Scenario 2 of TCP for various TCP window sizes with and w/o DCA

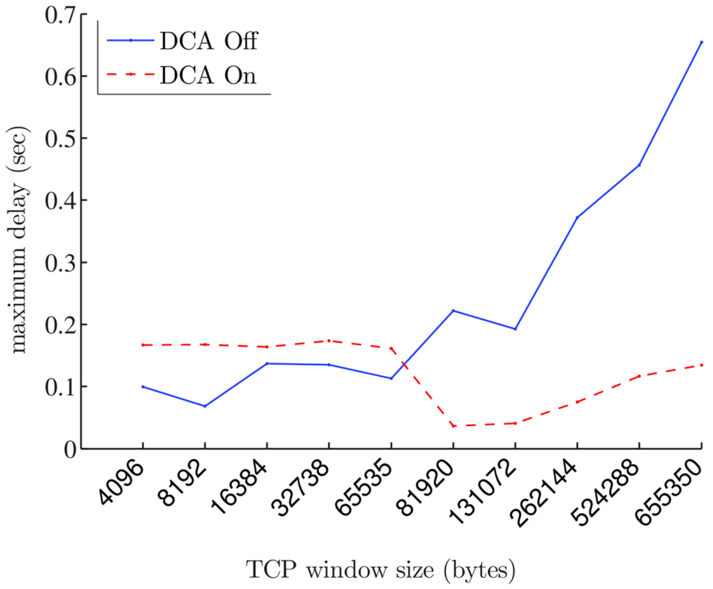


Figure 5.5: Maximal delay for Scenario 2 of TCP for various TCP window sizes with and w/o DCA

bytes), where goodput decreases from 9.714 Mb/s to 4.494 Mb/s and delay grows from 0.11326 sec to 0.16125 sec. For the consecutive simulation run (TCP rwnd = $81,920$ bytes), the resulting behaviour diverges. Now, when DCA is activated, the goodput results increases from 10.554 Mb/s to 13.116 Mb/s and peak delay subsides from 0.22219 sec to 0.03618 sec. Finally, for TCP rwnd = $655,350$ bytes, we see a massive increase in the performance of DCA as the goodput increases from 11.796 Mb/s to 51.999 Mb/s and the delay extensively drops from 0.65408 sec to 0.13416 sec.

Investigating further the divergence, we have discovered that the point that the graphs change behaviour is the rwnd step increase from $70,079 \rightarrow 70,080$, where we closely observe that $70,080$ is equal to $48 \times 1,460$, where $1,460$ bytes is the MSS attribute and 48 the value of the DCA triggering parameter σ. This finding assures us that there is definitely a negative association between DCA and TCP traffic. So, in order to trace out the TCP anomaly and find the cause for performance degradation, we need to simplify the scenario by designing a use case with a single TCP flow. Figure 5.6 illustrates the layout of a scenario with a single TCP flow with two (2) STAs, a sender (STA 1) and a receiver (STA 0). The data traffic is generated by an Internet File application which is classified as BE traffic and its characteristics can be retrieved from Table 5.11.

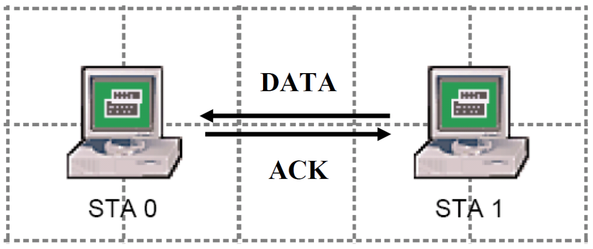
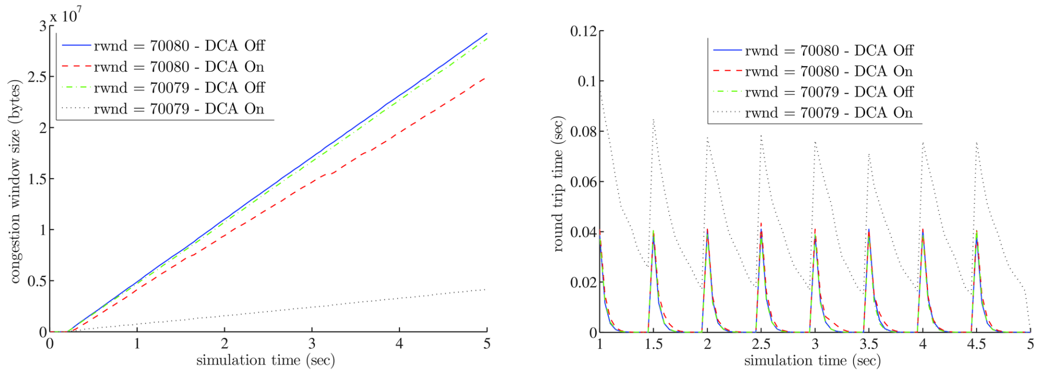


Figure 5.6: Layout of the single TCP flow scenario

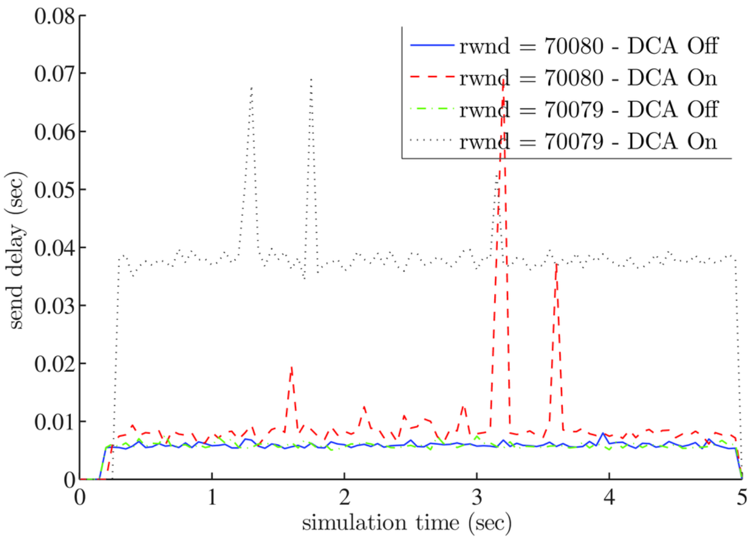
STA Name	Dest. STA	Role	Mean Rate	Rate Distrib.	MSDU	Delay
STA 1	STA 0	Internet File Transfer	120 Mbps	Constant, TCP	1,500 B	N/A

Table 5.11: Configuration of the single TCP flow scenario

As a first step, we will access OPNET's Discrete Event Simulation (DES) statistics of TCP delay, cwnd size and RTT calculations. In OPNET, TCP delay refers to the time (in sec) required to transmit buffered TCP data, computed as the time difference from the instance the TCP process started refraining from sending data due to small congestion window to the time when the data is sent out. In addition, the RTT is defined as a mean estimation of the round-trip delay, and is calculated based on the current measurement of the round-trip time and the smoothed mean deviation estimator of the previous and the current value. Figure 5.7 show the aforementioned performance measurements for a set of simulations with DCA enabled or disabled and the parameter of the TCP rwnd set at 70, 080 and 70, 079 bytes.



(a) Congestion Window vs. Sim. Time (b) RTT vs. Sim. Time



(c) TCP Send Delay vs. Sim. Time

Figure 5.7: OPNET results for single TCP flow scenario

If we assume a steady state transmission, no channel errors and without any congestion, the RTT of segments shall be maintaining a constant rate. Then, based on this assumption, we can also claim that the time to transmit a number of segments and receive TCP ACKs back will have a steady rate too. The responsive TCP ACK for each set of segments is very important because it will have an effect on the cwnd update. The cwnd behaviour was observed for the cases with or without DCA, for both buffer size, combinedly (see Figure 5.7a). It was found that without DCA algorithm the rate of change of the cwnd with respect to the simulation time is higher. This means that in a 5 second simulation time the cwnd was updated more often and so its size increased considerably. Also, it is noted that a huge difference arises between the two TCP cwnd results for both TCP rwnd when DCA is enabled. One can observe that at the end of the simulation run, for a TCP buffer up to $70,080$ bytes the correspondent cwnd is around $24,900,000$ bytes while for a buffer size of $70,079$ bytes only achieves an insubstantial $4,100,000$ bytes.

The discrepancy in the cwnd maximum size is explained by observing and comparing the update rate of both cases. In the DCA enabled case the cwnd was updated 552 times per second while without DCA the cwnd was updated $3,901$ times per second. This implies that during a simulation time of five (5) seconds the case without DCA increased the cwnd to such a value that more data was sent to the receiver thus increasing the throughput of the system. Each time an update occurs the cwnd is increased by $1,460$ bytes that is one (1) segment at a time. The delay that is introduced by the DCA algorithm will effectively have a repercussion on the cwnd update rate. The latter can also been seen in Figure 5.7b. The round trip delay for each bunch of segments has been increased for when compared to the normal channel access method.

So far, we have established that by deferring the channel access delay causes a drawback for the TCP traffic. Both RTT and cwnd comparative results verify this argument. However, we haven't pointed out if the problem is mainly originated by the delay of the responding acknowledgements from the receiver's side or does the sender take a huge responsibility in this performance degradation as well. Figure 5.7c shows the time required for the TCP layer to buffer a number of segments equal to the receiver's rwnd and have that data transmitted. Thus, it includes TCP refraining, MAC channel access delay and the delay from the Frame Aggregation procedure due to DCA algorithm. During a five (5) seconds simulation run, we observe that for both TCP buffer sizes the mean TCP send delay is around 0.056 sec. Once DCA is included, the mean outcomes increase, for rwnd = $70,080$ bytes the mean delay reaches around 0.089 sec and for rwnd = $70,079$ bytes escalates to a deferment of 0.359 sec. In conclusion, there is definitely an additional delay over the data transmission from the TCP initiator's side.

The original OPNET model for this new HT standard had to be significantly amended in order to provide additional monitoring information. Flags and extra variables were introduced in the code so as to be able to study how the DCA is reacting with different scenario attributes and parameters. As we've mentioned earlier, the DCA algorithm has three main way to trigger a formed frame in the aggregate buffer to acquire channel access permission. In general, the three triggers are, an aggregate size threshold (σ), an interval constraint starting from the initial packet in the buffer due to delay requirements (τ), and a time-out period due to a burst factor which forecasts the inter arrival time of the packets (γ). Note that γ is not really a triggering condition but rather a variable that updates another conditional parameter which will be the focus in more detail in Chapter 7, nevertheless within this chapter we consider it as a trigger. By setting the OPNET model in debugging mode, we were able to study the triggering mechanisms over the single TCP flow scenario and produce the results in Table 5.12.

TCP Buffer: 70,080 bytes				**Triggers**			
Traffic	**From**	**To**	**No. Packets**	**Total**	γ	τ	σ
DATA	1	0	33,153	700	13	0	687
ACK	0	1	11,665	728	728	0	0

TCP Buffer: 70,079 bytes				**Triggers**			
Traffic	**From**	**To**	**No. Packets**	**Total**	γ	τ	σ
DATA	1	0	5,485	127	127	0	0
ACK	0	1	2,091	154	154	0	0

Table 5.12: Triggering trend for different TCP buffer sizes

The results that we are interested in are the triggering types and the total number of packets that were transmitted. Based on the results, as an initial observation, we can determine that there are no τ triggers involved, something that is consistent with the QoS rules since the traffic in thus study is a BE flow and has no delay requirements. Continuing, we distinguish that for the STA 0 (the responder) and in both cases the DCA algorithm was only triggered by γ, 728 times for rwnd $= 70,080$ bytes and 154 times for rwnd $= 70,079$ bytes. Again, this behaviour is rational since the responder, in a perfect environment, only transmits acknowledgements (usually a single TCP ACK) to verify the successfully reception of the transmitted segments; within the TCP protocol the amount of data that has been sent but not yet acknowledged is known as Flight Size. This last observation although it shows a particular divergence, shall not really be incited as the main cause of the whole TCP problem since the performance degradation is a follow up behaviour of a problem caused by another source. The main difference between the two TCP buffer sizes is the triggering type of σ at STA 1, while for rwnd $= 70,080$ bytes there are 687 triggers issued, for rwnd $= 70,079$ bytes there are none except 127 events of the γ triggering type. Note that, the TCP buffer for the first case can fill up exactly 48 segments which is the precise equivalent to the DCA's number of packet threshold (σ), however for the second case there is a single segment difference as it only fills up to 47 segments at the TCP buffer. So, for the 48 segment threshold there is an immediate response for less segments, while the bulk has to wait till it is triggered by γ. This waiting period is the additional deferment that causes the TCP problem in study and is resulting in less transmitted packets.

This situation can also been illustrated in Figure 5.8 and results in the scenario where both the TCP layer and the MAC layer are waiting for data from each other. The TCP is waiting to get acknowledgements for a number of segments that had already been sent, before it can continue with the next rwnd while at the MAC layer the DCA is waiting for following segments to come from the upper layer before some of the other conditions can trigger the channel access stage. This leads to a point where both layers are dependent on each other to proceed and interlocked in a waiting period where they cannot do anything, hence the long delay results. Consequently, although we previously validated that DCA increases the channel efficiency for high priority flows when it comes down to the TCP traffic with small window buffers there is a dead-lock issue which needs to

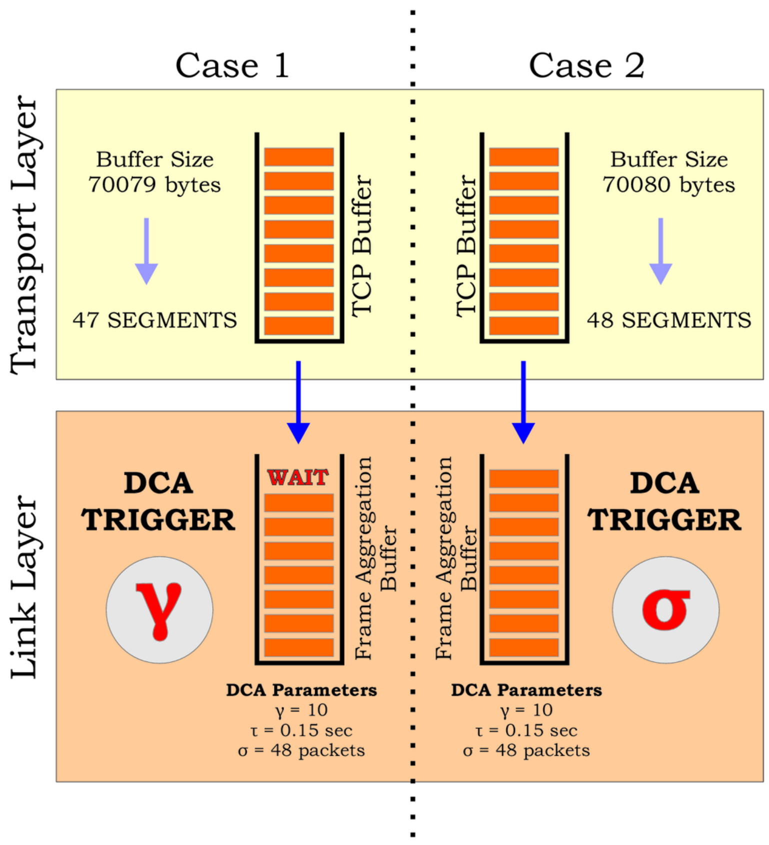


Figure 5.8: TCP and DCA behaviour

be resolved.

5.7 TCP Problem Over the Other Scenarios

So far, we've identified and described the problem of the TCP protocol in conjunction with the proposed DCA algorithm for a specific HT WLAN. The initial observation of the degrading performance was done over the overloaded WLAN scenario called *Scenario 2* and in order to determine the cause of the problem, a single TCP flow scenario was created. However, to establish our hypothesis that DCA may become deficient for smaller rwnd also with other WLANs, we need to provide the resulting behaviour for the rest of the usage case scenarios shown in previous chapters, e.g. for *Scenario 1*, *Scenario 4* and *Scenario 6*.

Scenario	TCP rwnd	DCA	UP	Off. Load		Goodput		Avg. Aggr.	Avg. Delay	PLR
1	655,350	Off	BE	83.524	31	59.362	7.36	26.725	0.36558	N/A
			VI		50.448		49.93	8.93	0.02217	0
			VO		2.076		2.075	1.19	0.00271	0
		On	BE	83.524	31	83.154	33.831	27.98	0.04525	N/A
			VI		50.448		47.247	13.66	0.04451	2.33
			VO		2.076		2.076	2.72	0.00684	4.86
	65,535	Off	BE	83.524	31	56.849	4.636	19.25	0.05536	N/A
			VI		50.448		50.139	6	0.01374	0
			VO		2.076		2.075	1.17	0.00257	0
		On	BE	83.524	31	57.199	4.809	24.48	0.06772	N/A
			VI		50.448		50.314	11.22	0.04122	2.78
			VO		2.076		2.076	2.37	0.00555	4.71

Table 5.13: DCA results with large and small rwnd for Scenario 1

By testing DCA with *Scenario 1* for small and large rwnd, we acquire the simulation results of Table 5.13. So, for a total OL of 83.524 Mb/s and rwnd set to $655,350$ bytes, the total goodput for a network with DCA disabled and enabled is 59.362 Mb/s and 83.154 Mb/s, respectively. Thus, we observe a high increase in performance close to the administered application services. On the other hand, when TCP rwnd takes a lower value of $65,535$ bytes then the system's performance, for both available cases of DCA, stays low. The TCP flow achieves throughputs of 4.636 Mb/s and 4.809 Mb/s for DCA disabled and enabled, respectively. Although the end goodput result rises slightly, the ~ 0.35 Mb/s increase cannot be considered justifiable for a DCA enhancement over low TCP buffer sizes since it is obvious that a deferment will only cause a MAC delay twice the period without DCA and with a PLR increase.

Scenario	TCP rwnd	DCA	UP	Off. Load		Goodput		Avg. Aggr.	Avg. Delay	PLR
4	655,350	Off	BE	460.176	451.024	62.061	52.909	42.95	0.63252	N/A
			VI		8		8.001	2.66	0.00766	0
			VO		1.152		1.151	1.13	0.00522	0.2
		On	BE	460.176	451.024	86.075	76.964	42.82	0.38581	N/A
			VI		8		7.958	19.64	0.05273	0
			VO		1.152		1.152	1.82	0.00932	0.3
	65,535	Off	BE	460.176	451.024	58.69	49.569	38.70	0.11170	N/A
			VI		8		7.969	2.69	0.00772	0
			VO		1.152		1.152	1.13	0.00529	0.2
		On	BE	460.176	451.024	77.494	68.476	35.9	0.07168	N/A
			VI		8		7.867	17.57	0.05135	0
			VO		1.152		1.151	1.62	0.00824	0.2

Table 5.14: DCA results with large and small rwnd for Scenario 4

In Table 5.14, we can observe the simulation outcomes for *Scenario 4*. Similar to the evaluation of the DCA performance at a previous chapter, we ascertain that when DCA is established, the network's performance increases effectively by 38.7% with the only downside being a negligible rise to the VO's PLR and some accumulation to VI's delay. As the outcomes of PLR and mean MAC delays remain below the QoS requirements while throughput increases, the DCA enhancement can be characterised as effective for the case of receiver's TCP buffer size equal to $655,350$ bytes. Diversely from above, when the rwnd is set to a lower value, the scenario's overall goodput still improves with a gain of 32.04%, in detail it increases from 58.69 Mb/s to 77.494 Mb/s. However, these results shall not be considered to definitively determine the efficiency of DCA with lower rwnd since the underlying behaviour doesn't explicitly originate from the DCA operation solely but from the scenario's set-up as well. The Large Enterprise scenario accommodates a wide number of TCP flows and this congestion may act unintentionally to the network's traffic control realization. The latter will be evidently supported over the next chapters where by modifying the channel access delay conditions, we can achieve even higher results.

Scenario	TCP rwnd	DCA	UP	Off. Load		Goodput		Avg. Aggr.	Avg. Delay	PLR
6	655,350	Off	BE	64.88	20	47.878	7.857	53.73	1.11206	N/A
			VI		42		37.144	40.24	0.23886	59.22
			VO		2.88		2.8783	1.44	0.00863	2
		On	BE	64.88	20	60.583	15.966	58.13	0.38268	N/A
			VI		42		41.738	27.26	0.04036	0
			VO		2.88		2.88	2	0.01091	3.4
	65,535	Off	BE	64.88	20	49.415	10.5	30.96	0.23790	N/A
			VI		42		36.042	39.92	0.29321	65.89
			VO		2.88		2.871	1.44	0.00886	2.3
		On	BE	64.88	20	61.45	16.933	40.23	0.11176	N/A
			VI		42		41.641	25.15	0.03761	0
			VO		2.88		2.876	1.97	0.01081	3

Table 5.15: DCA results with large and small rwnd for Scenario 6

Analogous, Table 5.15 yields OPNET's simulation results for *Scenario 6*. Here, we determine that from both rwnd situations, a performance improvement holds for the DCA method by a gain of 26.52% and 24.37% towards the initial goodput measurements for rwnd set to $655,350$ bytes and $65,535$ bytes, respectively. In spite of the improvement over the total goodput, the dominate advance over the network's performance applies to the PLR of the VI traffic. The system's apparent crowding with the majority of the STAs belonging to the highest AC, prevails upon lower ACs so the VI flows fetch PLR rates of 59.22% and 65.89%, unacceptable and deficient for video streaming. By introducing DCA, a better channel utilization takes place with flows applying the aggregation methods effectively, resulting in diminishing the PLR to zero ratings. Similar to previous scenario, first findings show that a lower rwnd configuration won't affect the simulation runs, yet this reaction mainly occurs because of the scenario's congestion and not of DCA.

5.8 Summary

In the previous chapter, we've identified issues arising from the poor interaction of the EDCA prioritized channel access mechanism defined in the IEEE 802.11e amendment and the frame aggregation mechanisms proposed in the latest HT standard. Using the original DCA algorithm with static parameters, we've seen that these issues can be addressed successfully. Through extensive simulation runs, we've demonstrated that DCA and its extensions can provide great fairness over lower ACs by deferring the transmission for all flows, including high ACs. Results show that for all contenting entities great improvement over the channel utilization and the total throughput while still obeying all QoS requirements.

Although we've seen that DCA increases the channel efficiency for high priority flows when it comes down to the TCP traffic with small window buffers there is a dead-lock issue which needs to be resolved. The alteration of the wireless channel contention by delaying the channel access granting request may prove to be diminishing for networks similar to *Scenario 1* and *Scenario 2* when STAs apply low TCP receiving buffers. However, testing over other overcrowded scenarios, e.g. *Scenario 4* and *Scenario 6*, have shown that DCA and low rwnd users may not after all affect overall performance since other characteristics such as congestion may become controlling factors. Anyhow, the TCP protocol and the MAC scheduling mechanism are independent and transparent to each other. An ideal situation would have been a mean of communication between these two layers in order to adjust attributes like rwnd according to QoS requirements. Since in reality this is not possible, we need to apply other solutions such as dynamically over time adapting DCA's parameters over the prominent TCP behaviour.

Chapter 6

Adaptive DCA

In previous chapters, we have seen that the new enhancements of the latest IEEE 802.11 amendment for HT STAs are considered valuable for reaching high data rate targets, since they mitigate transmission overheads by concatenating multiple data units into single frames and acquire higher channel bandwidth by using innovative channel transmission techniques. On the other hand, the set of QoS methods defined in IEEE 802.11e and handled by the EDCA mechanism seem to limit overall performance since delay-sensitive applications can be concerned with high importance and preeminent low priority flows. A STA with high priority traffic defers, on average, for less period than a STA with low priority traffic, so the number of data packets assigned in each aggregated frame turns out low, resulting the wireless medium to be utilized insufficiently. We demonstrated that this abominable behaviour can be resolved by introducing the DCA algorithm which impels STAs into further deferring in a way that it allows throughout buffering process more packets to arrive allowing the end aggregate size to accumulate, resulting in network's performance improvement. However, although DCA increases the channel efficiency when it comes down to the TCP traffic with small window buffers there are dead-lock waiting issues with deleterious outcome to the overall throughput.

Within this chapter, we will be introducing an extension of DCA, named as Adaptive DCA (ADCA). This proposed algorithm tends to dynamically alter one of DCA's parameter according to the current TCP transmission window size. Further simulation results will show the efficiency and competence of our innovative proposal towards the network's performance measurements.

6.1 Rethinking of DCA

The initial DCA algorithm as proposed in [19], it includes three main attributes that basically control and dictate its behaviour. The identifiers τ, σ and γ are preset and considered important for the determination of the algorithm's decisions. If we have previous knowledge of the network's layout, infrastructure and STAs service demands, we may be able to adjust the values of these parameters for each node in order to achieve maximum performance from the system. Still, within a WLAN the high rate of topology changes due to mobility or environmental factors [145] [146], may prove the predefined parameter solution impractical for a long-term situation.

The first two main attributes, τ and σ, determine the maximal waiting time for packets and the utmost allowable number of packets in the aggregation buffer, respectively, before aggregation is triggered. Despite the fact that both of these values define the behaviour of the DCA; once a value is set there is no flexibility, in the current implementation, to adapt the dynamics of the algorithm towards the incoming traffic flows. The last of the three DCA attributes, γ, which value determines the ratio of the inter-arrival time to the MAC channel access delay, it is considered a more flexible quantity even though it's set as constant as well. The reasoning behind this flexibility is that since the MAC channel access intervals will be deferring according to the STA's traffic flow bursts and WLAN's congestion conditions, in order to maintain a consistent transmission flow, DCA maintains records of the current traffic flow burst via the T_B variable and dynamically adapts the triggering mechanism concerning receiving and sending packets.

So, the way the current DCA is implemented is too rigid and there is no flexibility in adapting the parameters to any of the environmental factors, even if these are the incoming traffic flows, further QoS constraints or the eccentric characteristics of the MAC channel access scheduler. The only adaptability that currently provide is an adjusting mechanism over a specific traffic burst in order to avoid operational idling. An additional improvement would be to be able to adapt dynamically the aggregate size threshold (σ) based on feedback mechanisms within the DCA but at the same time keep transparency from other layers.

6.2 TCP Window Sizes

The latest Internet standard track protocol of TCP Congestion Control from Internet Engineering Task Force (IETF), a document known as *RFC 5681* [147], describes in detail the TCP's four intertwined congestion control algorithms, specifies how TCP should begin transmission after a relatively long idle period, and also discusses various acknowledgement generation methods. More importantly it defines state variables and describes the usage of them in order to maintain congestion management.

The set of state variables that are added to the TCP per-connection state and are of interest to the following proposal, are listed below:

- Congestion Window (cwnd): This value limits the amount of data a TCP can send.
- Receiver Window (rwnd): The most recently advertised receiver window.
- Flight Size: The amount of data that has been sent but not yet acknowledged.

Note that cwnd is a sender-side limit on the amount of data the sender can transmit into the network before receiving an ACK. As we've mentioned previously, every TCP protocol flavour has implemented a distinct function how to manage congestion and on how to surpass a situation like this. So, the operation of congestion control defers from each implementation but all focus on what values shall cwnd set to. On the other hand, the receiver's rwnd is a receiver-side limit on the amount of outstanding data and is based on the physical aspects of the TCP buffer length or the OS that the network is based on. By any means, these two state variables govern the data transmission rate and at any given time, a TCP transmission must not send data with a sequence number higher than the sum of the highest acknowledged sequence number and the minimum of cwnd and rwnd. The last state variable of interest is the Flight Size as it represents the number of segments that the sender has transmitted during a specific TCP state but not yet acknowledged by the receiver. The term of *FlightSize* can also be seen in TCP's protocol implementation and it has been proven to be an important variable since bounds the ssthresh parameter to be assigned with a value not larger than the current rwnd, which it could had been larger if cwnd variable was used instead.

The single TCP flow scenario set in Chapter 5 established as cause of the DCA & TCP problem the differentiation between receiver's TCP buffer size, declared in the rwnd attribute, and the number of packets formed in the DCA buffer before a transmission trigger is initiated, defined by σ triggering condition. So, if the expected number of packets stated by σ was bigger than the number of segments announced by the rwnd, then DCA could cause deficient results. A possible solution is to try to comprise both sizes or keep σ less than rwnd.

6.3 Set σ Equal to Receiver Window

Let's assume that there are no boundaries between the layers of the OSI reference model and a concept of cross-layer communication exists. Then, one layer is permitted to access the data of another layer, exchange information and enable interaction. Under these circumstances, the size of the receiver's TCP buffer can be known to the MAC layer via a simple feedback mechanism.

So, now that the rwnd size and the TCP segment length are available, we can compute the recommended number of packets that wait in the DCA buffer before the scheduler initiates a channel access procedure. Using *Scenario 2*, we derive some new simulation runs for a range of TCP window sizes, beginning with a size of $8,192$ bytes and increase by a growth factor of 2 till we reach $655,350$ bytes but this time we also adjust σ triggering mechanism, as shown in Table 6.1. Note, that the total number of packets may not exceed 64, the maximal allowable packets in a frame as defined in IEEE 802.11n, also as segment size we've used $1,460$ bytes. The rest of the DCA parameters are set to $\gamma = 10$ and $\tau \leq \frac{1}{2}$ maximal delay (e.g. $\tau_{BK\&BE} = 0.15$ ms, $\tau_{VI} = 0.1$ ms and $\tau_{VO} = 0.008$ ms).

Figure 6.1 illustrates the goodput results for the TCP flow of *Scenario 2*. The comparison between the cases of DCA disabled and enabled has already been

rwnd (bytes)	8,192	16,384	32,738	65,535	81,920	131,072	262,144	524,288	655,350
σ (packets)	5	11	22	44	56	64	64	64	64

Table 6.1: Respective σ values for each rwnd

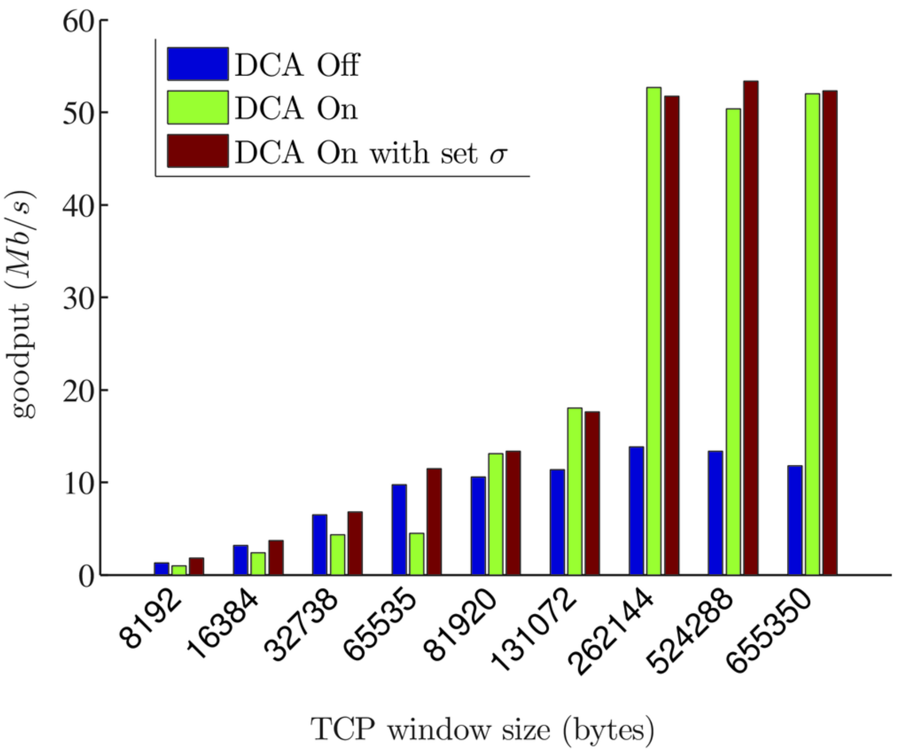


Figure 6.1: Goodput for TCP traffic vs rwnd for Scenario 2 with set σ

discussed before. The conclusion was that DCA will in general provide better results but before a certain point there is a negative impact over the performance and that can also be show over the bar graph. Now, the latest additional case of where DCA algorithm is enabled but the parameter σ has already been set with a value equal to the number of segments fit in a TCP rwnd, show an improvement throughout the simulation runs, although is still limited due to the small TCP bandwidth. Even for instances with low rwnd, the outcome exceeds the two other cases. More precisely, the numerical results of rwnd $= 65,535$ bytes show that the IF traffic of the BE flow achieves a throughput of 9.714 Mb/s, 4.494 Mb/s and 11.451 Mb/s, respectively for each DCA case. Reasoning behind the low throughputs for smaller rwnd sizes is the difference between the rwnd and the present cwnd. In an optimum environment with no congestion, without corrupted TCP segments and RTT within the delay boundaries, the number of segments on every TCP transmission step will be increasing until it reaches the maximal allowable size of the advertised rwnd. Nevertheless, before the transmission size reaches rwnd, the cwnd will be starting from a rather small number and on every step will be adding one or more segments. Hence, the TCP Flight Size not always equals rwnd, especially at the beginning of the stage. In conclusion, being able to adjust the triggering parameter σ, DCA will be definitely improve performance even for TCP traffic with low rwnd but the magnitude of that improvement is determined from the attribute setting mechanism.

6.4 Adapting σ Towards TCP Flight Size

A first challenge towards the proposed solution was to set DCA's parameter σ equal to TCP rwnd. If a cross-layer communication existed, this could have easily be done through exchange of information over different layers. However, in a real case scenario, the layers of an OSI reference model are transparent and independent from each other. Therefore, a realistic solution will require a mechanism that would be able to determine or more preferably guess a TCP Window Size within the MAC layer only. A quick resolution is to increment the aggregate size threshold, σ, gradually until the Flight Size of the TCP flow concurs, in this way we can always guarantee that $\sigma \leq$ Flight Size. The traffic flow burst variable, T_B, will then be used as a condition to determine if σ has reached the maximal value of the TCP Flight Size. This very simple dynamic adjustment of DCA's parameters is shown in Figure 6.2.

So, the DCA parameter σ initializes with a starting low value and every time DCA's mechanism is triggered this value will be either incremented or decreased analogous. If the trigger was caused by σ itself, it means that the number of packets in the aggregation buffer is more likely less or equal than the number of segments in a TCP Flight Size, therefore next σ will be increased by a unit. Otherwise, if the IAT trigger was initiated then the algorithm assumes that the size of the packets in the buffer queue has exceeded the TCP Window Size and there are no more segments to be received from the above layer. Once the first

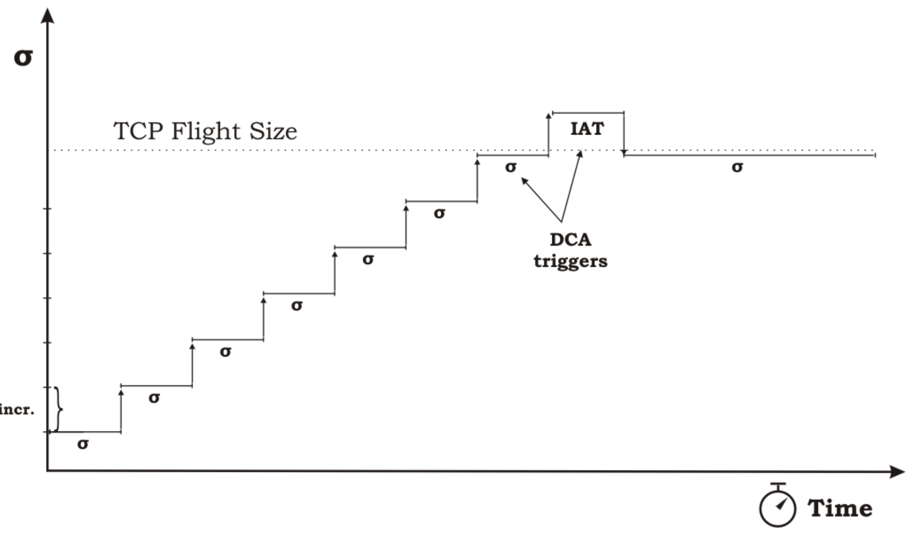


Figure 6.2: An example of DCA with adaptive σ

IAT trigger appears, the algorithm will assume that the prior σ value was also the maximal buffer size for TCP transmissions and so reduces it by a single packet and keeps that value throughout a specified period. The IAT trigger, stands for Inter-Arrival Time and it describes the true condition of the current interval timed from the last received packet, T_L, has exceeded the last recorded traffic burst duration, T_B. The conditional statement describing IAT triggering is denoted in Equation 6.1.

$$(t - T_L > T_B) \rightarrow IAT \tag{6.1}$$

The operation of ADCA outlines a similar mechanism of that of a standard process that TCP protocol follows. Taking as an example TCP's slow-start mode, the transmitter initiates a transmission of some low number segments, the initial TCP Flight Size. After a successful transmission, which will be determined by a preceded TCP ACK from the receiver, the next TCP cwnd will be accumulated by one or more segments. The increments of the number of segments has been defined differently in each TCP extension but it logically increments exponentially. The size of the window buffer may vary dynamically depending on the timeouts or retransmissions that may occur, this is why this process is also known as a sliding window protocol. So, for every further successfully transmissions the TCP Flight Size increases till it reaches receiver's rwnd. Thereafter, the sliding window remains stable or increases slightly by a single packet. For any network traffic congestion issues the following number of segments drops back to the initial point. Nevertheless, transparency remains throughout the layers and so the application will still be offering data for transmission to TCP without being aware of any of these fluctuations. So, the TCP window size varies dynamically depending on the network traffic and the buffer length of ADCA follows a similar behaviour but depending on the DCA's triggering mechanisms. When both processes follow a similar pattern, the achieving performance outcomes can be proven optimal.

So, to investigate further the performance of the latter solution, the HT model in OPNET had to be modified accordingly. For that reason, adjustments were carried out within DCA algorithm's functions and a decisional controller for σ, that will increase its value as described above, was implemented. Again, another set of simulations were performed over *Scenario 2* with exactly the same settings as before for comparison purposes.

The results of the simulation runs for DCA with σ dynamically increasing towards an estimated TCP Flight Size can be seen in Figure 6.3. From the bar graphs we observe that for lower rwnd buffer sizes, the latest enhancement surpass previous results. Specifically, for a rwnd buffer size equal to $16,384$ bytes the TCP throughput for the IF flow is 10.761 Mb/s, an increase of around 338.4% and 454.6% when DCA is disabled and enabled in its initial form, respectively. Likewise, when the receiver's advertise window is $65,535$ bytes, the throughput gain is around 3.9 and 8.3 times more than the previous cases. However, as the rwnd sizes start to largely increase the advancement over the original DCA algorithm diminishes. So, for buffer sizes above $262,144$ bytes the DCA with adapted σ is bounded to a throughput of ~ 43 Mb/s where for unchanged DCA the outcome reaches up to ~ 52 Mb/s.

Overall, the results reveal potential advancements over a range of TCP buffer sizes especially when these are rather small. So, the experimental dynamic adjustment of σ depending over other traffic characteristics is the right way for advancement to the solution. However, the end algorithm requires some additional polishing as specific characteristics of the TCP behaviour ought to be taken into

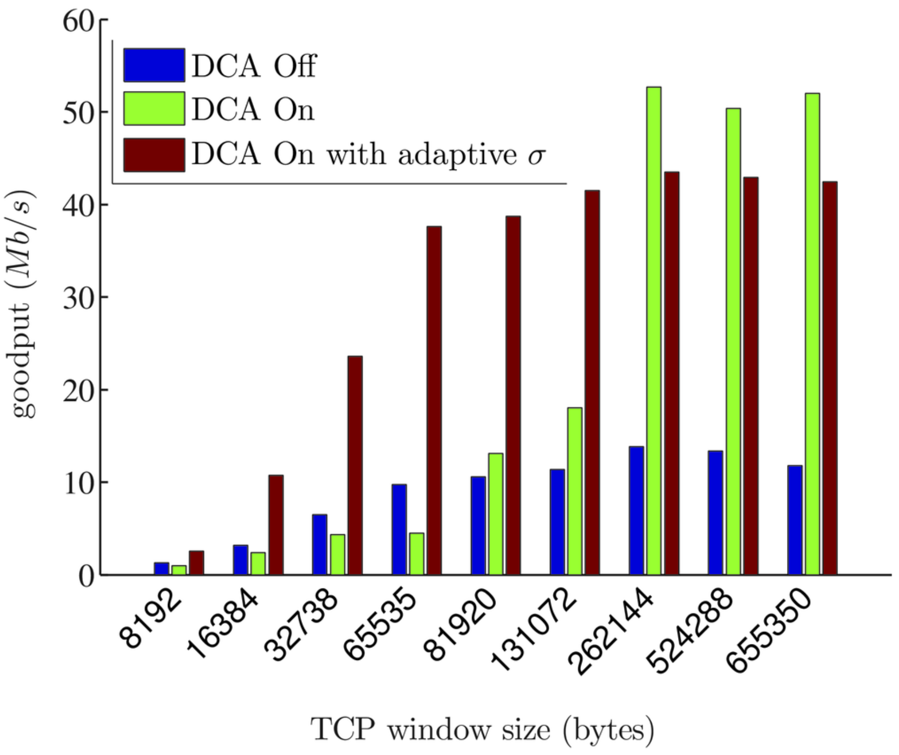


Figure 6.3: Goodput for TCP traffic vs rwnd for Scenario 2 with adaptive σ

consideration and these may possibly improve the results even further. A simple factor that hasn't been examined is the TCP congestion control which its function alters in time the cwnd depending on the TCP's protocol policy that it follows. So, the flexibility of the DCA over dynamically changing its parameters needs to follow additional rules.

6.5 Design of Adaptive DCA

The next challenge was to make the DCA fully adaptive over its aggregate size threshold attribute, σ. The proposed solution is to use the IAT as a condition to be able to increase or decrease the aggregate size threshold but at the same time make the σ be able to stabilise for a particular period of transmissions before attempting to alter its value. The new proposed algorithm will have an adaptive σ attribute and two marginal values, a minimum value (σ_{min}) which within our model we set as a minimum aggregate size four (4) packets, and a maximal value (σ_{max}) that can be set at any value up to the allowable maximal packets in an aggregated frame (64 packets). The adjusting σ will stabilise to any value below σ_{max} and that should yield an optimum TCP throughput. Therefore, σ won't have to be known and be set prior DCA's operation but it will be computed dynamically.

The proposed ADCA algorithm is built on the basic concept of the original DCA algorithm but some supplementary functions, attributes and conditions are introduced so as to make the system perform better under various changing traffic behaviour. Figure 6.4 illustrates the flowchart of current DCA algorithm including the conditional triggering mechanism. So far, the algorithm has three pre-set attributes, τ, σ and γ, and only the latter is taking the role of adjusting the behaviour of DCA according to the conditional IAT trigger. The squared red section points out the part of the transition diagram where changes will take place for the new proposed ADCA algorithm. Additionally the circled characters, *'A'*, *'B'*, *'C'* and *'D'*, denote connections between the algorithm's input and output transition flows of the legacy DCA and new ADCA and will be used as reference points within the description of the latter. The main objective of ADCA is to find a value where the queue is most likely to be at steady state, considering that the TCP connection doesn't have many imbalances.

The new proposed DCA enhancement characterised by its adaptability will be based on the two main conditions, the inter-arrival time trigger (IAT) and the current aggregate size threshold (σ). The flowchart in Figure 6.5 outlines the modifications of ADCA that were made over the original DCA algorithm.

On this basis, the mechanism of ADCA will perform the following course of actions. At start, every queue buffer will initialize its aggregate size threshold with σ_{min} as commencing value. Then, σ will be increasing each time, triggered by the aggregate size threshold condition and this will provide a measure of improving the aggregate size gradually and dynamically. So, the next σ value is updated by a predefined variant but without exceeding the maximum allowable size, σ_{max}. On other hand, if IAT condition is triggered, meaning that there was an excessive waiting time at the MAC, current aggregate size, σ, will initially decrease its threshold by the predefined fluctuation value and if the following

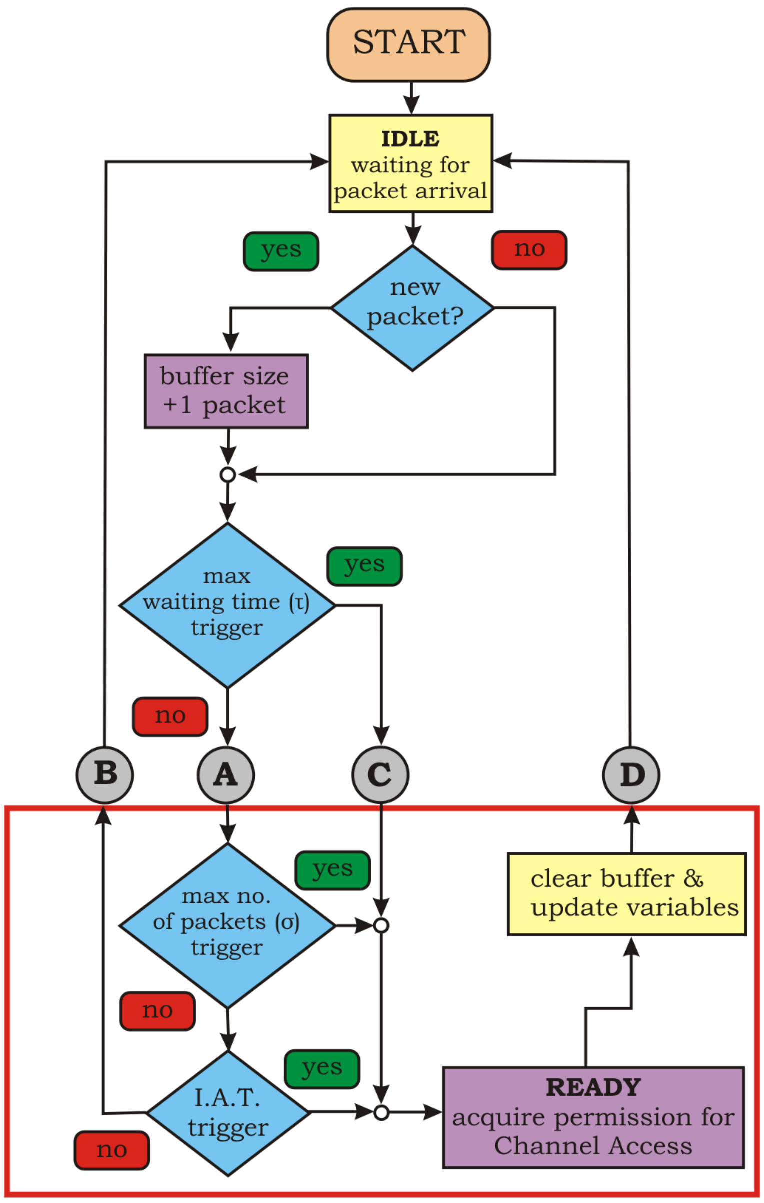


Figure 6.4: The flowchart of the original DCA algorithm

triggering behaviour continues then the threshold shall be dropped down to the preset minimum value, σ_{min}. Eventually, the flutter will stabilize with the number of packets in the buffer queue being similar to the number of segments sent from the TCP layer.

Nevertheless, it is possible for the algorithm to develop an oscillatory be-

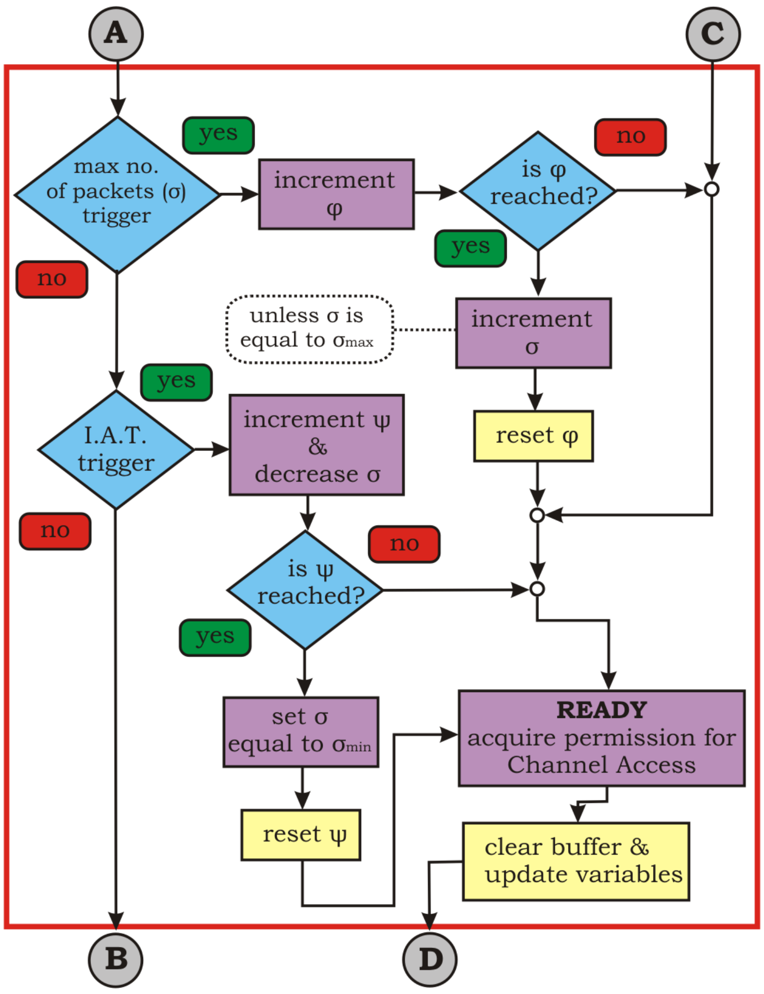


Figure 6.5: The flowchart of the proposed Adaptive DCA algorithm

haviour where the triggering will bounce from the aggregate size threshold condition to the burst factor condition. In order to balance such an attitude, we also introduce some oscillation controllers which establish the number of times the individual triggers occur. We use two oscillation counters, psi (ψ) and phi (ϕ) for the IAT and σ triggering conditions, respectively. When the σ is stabilised, we will keep it in steady-state for a definite amount of packet formed transmissions, ϕ oscillator controller, and then try again to increment the aggregate size threshold. Likewise, in order to avoid cropping the buffer size down to σ_{min} just from a single IAT trigger occurrence and definitely affect the overall performance, ψ

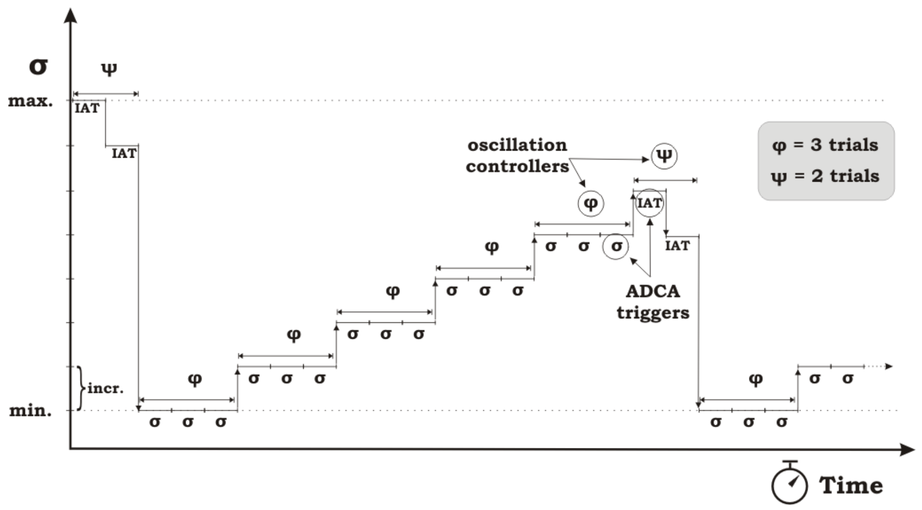


Figure 6.6: Simple example of Adaptive DCA over time

oscillator is introduced. The algorithm becomes flexible and withstands possible glitches within a certain preset number of trials. Figure 6.6 shows how σ varies over time and also indicates the importance of ADCA's oscillator factors. The examples assigns the set $(\phi, \psi) = (3, 2)$ to the oscillator controllers.

Table 6.2 shows the triggering trend for a TCP buffer size of $65, 535$ bytes when using ADCA over a single TCP flow scenario. In Chapter 5, the legacy DCA fired too many IAT alerts and the reasoning behind it was the size mismatch between rwnd and σ. Nevertheless, ADCA resolves that issue and from the results derives the factual evidence that the proposed algorithm has a double benefit towards the sender with the aggregated segments and the receiver having consequent ACK responses. In the table, we denote as IAT triggering condition the attribute γ since this is the DCA parameter that updates its rates from. With ADCA the majority of the triggers occur due to maximum number of packets in buffer (σ). Specifically, for the DATA flow and for the ACKs there were 756 and 776 σ triggers out of total 771 and 807, respectively. While with any rwnd size over the legacy DCA, we show that the receiver's ACK responses always actuate channel access action via the IAT condition. The outnumbered IAT triggers are mainly caused during the adjustment phase or when TCP traffic causes irregularities but still are considered negligible.

TCP Buffer: 65,535 bytes				**Triggers**			
Traffic	**From**	**To**	**No. Packets**	**Total**	γ	τ	σ
DATA	1	0	32,854	771	15	0	756
ACK	0	1	4,886	807	31	0	776

Table 6.2: Triggering trend for TCP buffer size 65,535 bytes

In conclusion, such a variable and adaptable buffer size threshold, σ, allows the MAC layer to proceed with the ADCA operation without the need to have a priori knowledge of the flight size or what type of flow will be received from the upper layers. The ADCA can work for both UDP and TCP protocols and each buffer queue from each AC will maintain individual values for its attributes. Last but not least, an exact pattern of behaviour for TCP traffic doesn't exist as the TCP mechanism reacts differently at every step while considering many internal and external factors. However, a close representation within the ADCA methods should provide optimum results.

6.6 Performance Evaluation of Adaptive DCA

In this chapter, we proposed an extension of DCA with an adaptive aggregated size threshold (σ), thus was given the name ADCA. We resolved the issue of the degrading performance over small rwnd due to the opposing queue sizes of the TCP and MAC layers. The solution to the problem of balancing both of these queue sizes so that the TCP protocol won't have to wait for a responsive ACK and at the same time DCA would form an aggregate ready for transmission to create an algorithm that would follow closely TCP actions and dynamically adapt its conditional triggering mechanisms. In this section, we evaluate its performance thoroughly by using the predefined use case scenarios known in this document as *Scenario 2*, *Scenario 1*, *Scenario 4* and *Scenario 6* and for the set parameters we used: $\sigma_{max} = 48$ packets, $\sigma_{min} = 10$ packets, $\sigma_{incr} = 2$ packets, $\phi = 5$ times and $\psi = 2$ times.

The results of TCP traffic for *Scenario 2* for various rwnd sizes can be seen in Figure 6.7. Each simulation run displays the TCP throughput for the BE traffic for different rwnd sizes. Very much like previous proposed attempts, the product

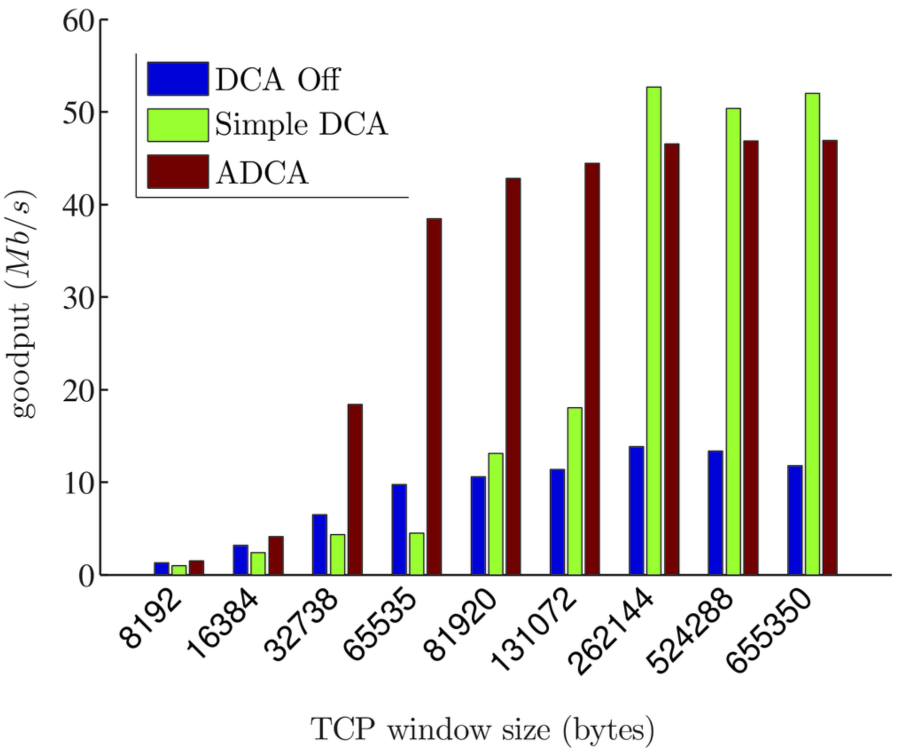


Figure 6.7: Goodput for TCP traffic vs rwnd for Scenario 2 with ADCA

of the latest ADCA function surpasses the outcome of IEEE 802.11n HT standard without applying any further amendments and that of original DCA algorithm. The augmentation of the performance is most obvious within the range of $32,738$ – $131,072$ of the independent variable rwnd. It is interesting to mention that for a rwnd equal to $65,535$ and with DCA and ADCA applied, the throughputs are 4.494 Mb/s and 38.454 Mb/s, respectively, an increase of around 855.7%. The latter proves the efficiency of how a deferment before accessing the wireless channel improves channel utilization but only if algorithm has been designed rationally.

More numerical results of *Scenario 2* can also be seen in Table 6.3. It provides the goodput and average delay outcomes of all accommodate traffic for low and high rwnd when ADCA is disabled and enabled. For all HDTV flows, the offered load is transmitted fully in every single case and while having slightly longer delays than those without ADCA are still delivered well below the allowed maximal

ADCA	TCP rwnd	Traffic	Offered Load	Goodput	Avg. Delay
Off	655,350	HDTV	19.2	19.197	0.00100
		HDTV	24	23.994	0.00115
		Internet File	120	11.796	0.39693
On	655,350	HDTV	19.2	19.194	0.00661
		HDTV	24	23.952	0.00824
		Internet File	120	46.89	0.10278
Off	65,535	HDTV	19.2	19.2	0.00087
		HDTV	24	23.997	0.00100
		Internet File	120	9.714	0.03194
On	65,535	HDTV	19.2	19.122	0.00495
		HDTV	24	23.97	0.00595
		Internet File	120	38.454	0.00775

Table 6.3: Numerical results for Scenario 2 with ADCA for low and high TCP window sizes

delay of 200 ms. For the BE traffic, ADCA provides a better channel utilization and results in a throughput of 46.89 Mb/s and 38.454 Mb/s for high and low rwnd, respectively. Out of 163.20 Mb/s total offered load, the new algorithm achieves 90.04 Mb/s and 81.55 Mb/s total goodput, a MAC efficiency of 55.17% and 49.96% for each rwnd case.

Scenario	TCP rwnd	ADCA	UP	Off. Load		Goodput		Avg. Aggr.	Avg. Delay	PLR
1	655,350	Off	BE	83.524	31	59.362	7.357	26.725	0.36558	N/A
			VI		50.448		49.93	8.931	0.02217	0
			VO		2.076		2.075	1.187	0.00271	0
		On	BE	83.524	31	80.021	27.634	24.97	0.08034	N/A
			VI		50.448		50.312	9.133	0.03382	2.1
			VO		2.076		2.075	2.62	0.00654	2.7
	65,535	Off	BE	83.524	31	56.849	4.636	19.245	0.05536	N/A
			VI		50.448		50.139	5.995	0.01374	0
			VO		2.076		2.075	1.17	0.00257	0
		On	BE	83.524	31	73.59	21.186	17.605	0.01598	N/A
			VI		50.448		50.329	7.982	0.03131	2
			VO		2.076		2.075	2.511	0.00643	1

Table 6.4: ADCA results with large and small rwnd for Scenario 1

For simulating ADCA with *Scenario 1* for small and large rwnd, we acquire the results of Table 6.4. So, for a total OL of 83.524 Mb/s and rwnd set to $655,350$ bytes, the total goodput for a network with ADCA disabled and enabled is 59.362 Mb/s and 80.021 Mb/s, respectively. Before, we tested simple DCA with a lower TCP rwnd of $65,535$ bytes and the outcome was diminishing. However, with ADCA the system's total throughput performance reaches up to 73.59 Mb/s well above previous results of DCA's mechanism. The formed frames have increased even for the higher ACs, resulting in a better channel utilization. Within this scenario we can justify channel access delay as a method for improvement even for various TCP traffic characteristics.

In Table 6.5, we can look up the simulation outcomes for *Scenario 4*. Similar

Scenario	TCP rwnd	ADCA	UP	Off. Load		Goodput		Avg. Aggr.	Avg. Delay	PLR
4	655350	Off	BE	460.176	451.024	62.061	52.909	42.95	0.63252	N/A
			VI		8		8.001	2.66	0.00766	0
			VO		1.152		1.151	1.13	0.00522	0.2
		On	BE	460.176	451.024	81.989	72.887	40.77	0.42764	N/A
			VI		8		7.95	11.48	0.02882	1
			VO		1.152		1.152	1.74	0.00893	0.7
	65535	Off	BE	460.176	451.024	58.69	49.569	38.70	0.11170	N/A
			VI		8		7.969	2.69	0.00772	0
			VO		1.152		1.152	1.13	0.00529	0.2
		On	BE	460.176	451.024	79.415	70.304	38.4	0.07592	N/A
			VI		8		7.962	10.93	0.02890	0.8
			VO		1.152		1.150	1.71	0.00880	0.4

Table 6.5: ADCA results with large and small rwnd for Scenario 4

to the evaluation of the DCA performance at a previous chapter, we ascertain that when a form of DCA is established, the network's performance increases effectively with the only downside being a negligible rise on the average delay. As the total goodput increases by 32.11% and 35.32% for high and low rwnd, respectively, so does the average delay of VI by ~ 3.7 times more for both cases. Nevertheless, as the outcomes of PLR and mean MAC delays remain below the QoS requirements, the DCA enhancement can be characterised as effective. It is interesting how the aggregated sizes for the VI flows increase from 2.66 and 2.69 packets to 11.48 and 10.93 for the correspondent high and low rwnd sizes.

Scenario	TCP rwnd	ADCA	UP	Off. Load		Goodput		Avg. Aggr.	Avg. Delay	PLR
6	655,350	Off	BE	64.88	20	47.878	7.857	53.73	1.11206	N/A
			VI		42		37.144	40.24	0.23886	59.22
			VO		2.88		2.8783	1.44	0.00863	2
		On	BE	64.88	20	56.833	12.17	55.32	0.83852	N/A
			VI		42		41.728	16.49	0.02454	0
			VO		2.88		2.875	1.95	0.01028	2.2
	65,535	Off	BE	64.88	20	49.415	10.5	30.96	0.23790	N/A
			VI		42		36.042	39.92	0.29321	65.89
			VO		2.88		2.871	1.44	0.00886	2.3
		On	BE	64.88	20	60.959	16.167	39.07	0.14825	N/A
			VI		42		41.914	24.9	0.05686	0.2
			VO		2.88		2.877	1.99	0.01067	2.9

Table 6.6: ADCA results with large and small rwnd for Scenario 6

Again, the results in Table 6.6 for *Scenario 6* demonstrate an overall improvement. TCP throughput increases from 7.86 Mb/s to 12.17 Mb/s when high rwnd is used and 10.5 Mb/s to 16.17 Mb/s, otherwise. The key role of ADCA can be highlighted when comparing the PLRs for the VI flow. In the case of rwnd = $655,350$ bytes the PLR for a normal case is at the unacceptable rate of 59.22%, while for rwnd = $65,535$ the PLR increases more at 65.89%. But once ADCA is enabled all packets can be received successfully with negligible loss ratio, PLR $\rightarrow\sim 0$. Additionally, a decrease of the VI flow occurs but this is normal since ADCA has stabilized the EDCA prioritization and now that VO flow use the resources with fairness, the VI have more channel access and consequently a significant drop on the PLRs.

6.7 Summary

In most cases, frame aggregation adheres due to the EDCA scheduler's priority mechanism, resulting in the network's poor overall performance. Using the original DCA algorithm with static parameters, we've seen that these issues can be addressed successfully, however when various TCP windows sizes are inspected further issues arise. By incrementing the aggregate size threshold, σ, gradually until the flight size of the TCP flow is reached, we eliminate any TCP problems and MAC deadlocks. The static DCA is too rigid and there is no flexibility in dynamically adjusting the parameters. Our proposed adaptive DCA administers the contingency to incorporate adaptability. The simulation results evinced that the ADCA operation with various TCP window sizes over different scenarios improves the system performance significantly as compared with systems abstaining delayed channel access. The impact of the proposed modified DCA enhancement over the network's performance is perceived as outstanding and the architecture of this innovative idea can be considered as a guide for the design of future HT standards.

Chapter 7

Selective DCA

In previous chapters, we have demonstrated that over a HT network that is compliant with the IEEE 802.11n standard, the achieved throughput performance can be bounded to lower rates. The DCA solution aims to resolve the poor channel utilization and the low efficiency that high priority stations adhere due to shorter waiting times allocated by the EDCA function. The algorithm operates over the MAC layer and based on the traffic characteristics, it delays the packets from being transmitted by postponing the channel access request. As a result, the average aggregate size of a high priority flow increases and consequently so is the channel efficiency and the end overall performance. However, in some situations we have noticed that further deferring has a negative impact with applications that are using the TCP/IP protocol rather the UDP transport protocol. A possible resolution is a traffic awareness feature that will allow the algorithm to distinguish any data transferred over TCP and if found necessary to override the additional MAC delay that DCA applies. But before we are able to classify the incoming traffic, first we need to come upon with the conditional elements that the selection process will be based on. A simple empirical analysis of the traffic flow behaviour can provide a distinguished pattern that the selection process can be based on.

Within this chapter, we will be introducing another extension of DCA, named as Selective DCA (SDCA). This proposed algorithm will be examining the behaviour of the incoming traffic flows, based on characteristics of inter-arrival intervals, packet lengths, short recordings of previous packets, assigned destination addresses, etc., and accordingly will be deciding if the MAC aggregate buffer shall be deferred further or not. Further simulation results validate the efficiency and competence of our innovative selection process towards the network's performance measurements.

7.1 UDP & TCP Usage Over Large Networks

The necessity of network traffic monitoring and analysis is growing dramatically with the increasing network usage demands. Wireless technology has become ubiquitous to computer networks located in small and broad areas, so it is increasingly important to understand trends in the usage of these networks specifically. A study dated back to 2004, shows that over a three (3) years period there were dramatic increases in usage, and changes in the applications and devices used on the authors' campus wireless network [148]. The WLAN was initially dominated by WWW traffic but the latest traces presented significant increases in P2P, streaming multimedia, VoIP, and IF traffic. Also, the proportion of heavy users on our WLAN remained static, despite the shift from early adopters to a more general population.

A more recent study shows that even though there is a growth over the network's flows, there is no clear evidence that the ratio between TCP and UDP transfers has increased or decreased [149]. Again, the ratio is rather dependent on application popularity and, consequently, on user choices. However, the majority of traffic volume is dominated by the TCP protocol in contrast with the UDP which is ahead with the number of flows, mostly from P2P applications. A traffic flow is defined as a sequence of packets sent from a particular source to a particular destination that the source desires to label as a flow. Also, a flow could consist of all packets in a specific transport connection or a media stream [150]. The analysis of the network traffic trace in [151], summarizes that the byte count and packet count of TCP traffic is much larger than those of UDP traffic, while the flow count of TCP traffic is two times smaller than UDP traffic. More specific, the proportions, out of all flows, packets and length, for the TCP traffic are 34%, 93% and 98% and for the UDP traffic are 63%, 6% and 2%, respectively. Note, that after establishing a TCP connection (using the three-way handshake) the communication between the hosts is bi-directional, thus two flows are created. On the other hand, the communication of the applications using the UDP protocol as mean of transportation are unicast, so only single flows exist. Last but not least, UDP transport provides Best-Effort delivery but in terms of QoS the majority of low priority BE traffic use TCP instead.

7.2 TCP-Aware DCA

So far, we have demonstrated DCA's increase on the system's overall performance but over low TCP buffer sizes the results may be contrary to the expectations. A solution to the issue is to enhance DCA with a TCP awareness functionality, meaning that an agent will identify TCP flows over the incoming traffic. This measure will allow the DCA algorithm to operate as normal until a packet arrives from the upper layers that includes a TCP header. In this case, DCA will initiate the channel access procedure for the packets that are currently formed in the aggregate buffer. For any following TCP packets, DCA will defer no further the transmission and a zero-waiting mode will be taking place. The functionality of TCP-Aware DCA is illustrated in Figure 7.1.

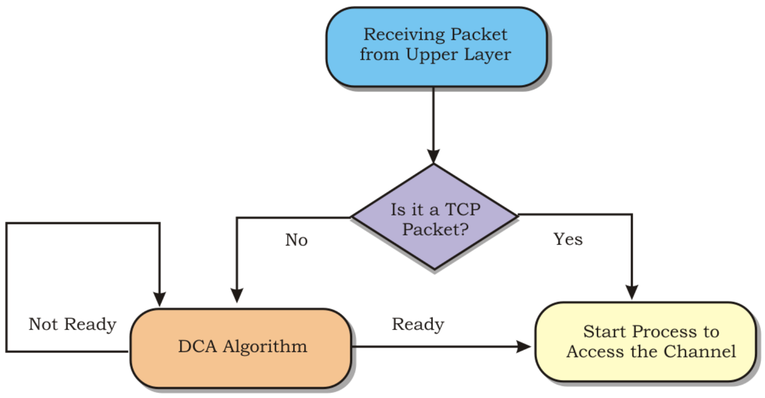

Receiving Packet from Upper Layer
Is it a TCP Packet?
No
Yes
Not Ready
DCA Algorithm
Ready
Start Process to Access the Channel

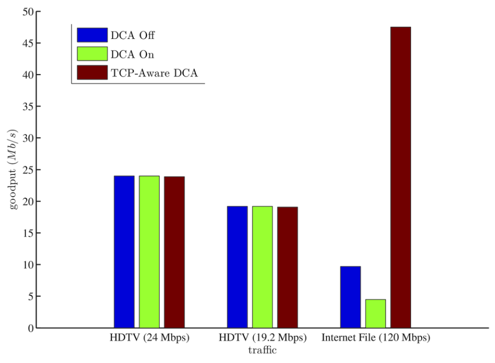
Figure 7.1: Flowchart of TCP-Aware DCA

The challenge in this chapter is how to let the MAC layer know that a data unit that arrives from the upper layer is TCP or UDP. If we assume that there are no boundaries between the layers of the OSI reference model and a concept of cross-layer communication exists, then one layer is permitted to access the data of another layer, exchange information and enable interaction. A physical example to that solution is to amend the IEEE 802.1D standard [70] and provide an additional support for the MAC service that will state the transport protocol. This procedure is similar to the *"Maintenance of QoS – Priority Mapping"* but it uses a mapping table for the transport protocol instead. The information from the IP layer can be retrieved via the "Protocol" element of the IPv4 (or IPv6) header.

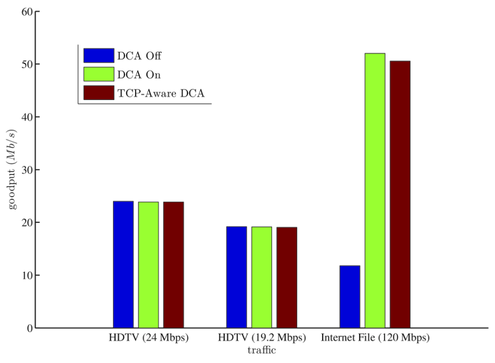
A preliminary study regarding an enhanced DCA algorithm that is able to determine TCP instances, was introduced by the author of this Thesis in his MSc Dissertation [152]. The simulation results of the study were retrieved using OPNET together with the HT model but altered accordingly. The original model had to be modified at the beginning of the DCA's method, so instead of checking if the incoming packet is the first arrived to queue's buffer, we just inquire if it is a TCP packet. If yes, we send it straight for transmission along with other packets that may have waited in the aggregated buffer. In OPNET, a special data type called ICI can contain fields for user-defined parameters to be shared by multiple entities in the network and consecutively these are referenced in calls to Kernel Procedures (KPs) from within process models [63].

An ICI becomes associated with an interrupt if a process initializes the ICI prior to taking the action that causes the interrupt. ICIs are dynamic simulation entities, since they are created and destroyed as needed during the execution of a process. The KP *op_ici_create()* is used to create ICIs, based on a specified format. An ICI format, which has been created using the ICI Format Editor, determines the list of attribute names and data types supported by the newly created data structure type. After an ICI is created it returns a pointer which is used to reference this ICI in most KPs. ICIs have been created in almost every layer of the 802.11n OPNET model, so we modified an already existing one, by adding extra information about the transport protocol in use and move it along to the MAC layer and act accordingly.

Similar with previous chapter, for the evaluation purpose of the newly proposed TCP-Aware DCA algorithm, simulation runs over *Scenario 2* were designed and carried out. Figure 7.2 and Figure 7.3 display the results in bar charts for the goodput and mean delay for all participating applications. The figures also show a comparison between scenarios with small (65, 535 bytes) and large (655, 350 bytes) and DCA enabled, disabled or enabled with TCP-Aware agent.

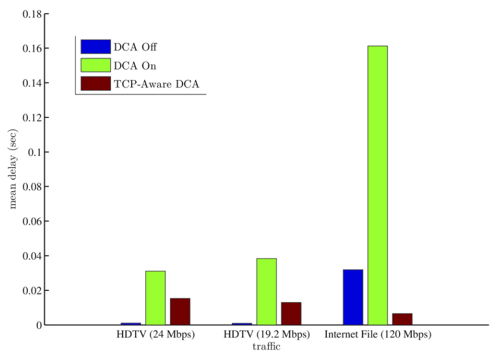


(a) TCP rwnd set to 65,535 bytes

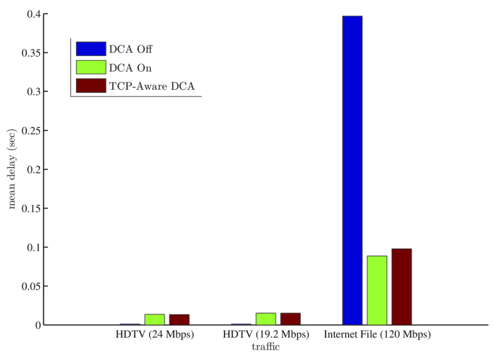


(b) TCP rwnd set to 655,350 bytes

Figure 7.2: Goodput for low and high rwnd using TCP-Aware DCA



(a) TCP rwnd set to 65,535 bytes



(b) TCP rwnd set to 655,350 bytes

Figure 7.3: Average Delay for low and high rwnd using TCP-Aware DCA

The goodput results for the HDTV in both TCP rwnd cases are equal and do correspond to the initial offered load of 24 Mb/s and 19.2 Mb/s, respectively. The mean delay for the two cases that DCA is enable has increased expectedly as we have already mentioned that by applying further deferment endorses to a better channel utilization. However, with TCP-Aware DCA enabled, the delay is much smaller for the low TCP rwnd and at the same time the throughput for the TCP application surpasses all other cases. Specifically, the throughput when DCA is enabled achieved 9.71 Mb/s, when disabled is 4.49 Mb/s and when enabled conjointly with TCP-Aware reaches up to 47.496 Mb/s. For larger TCP rwnd the enhanced DCA works equally well with the original algorithm and it is logical that there are no significant differences between them.

In conclusion, the enhanced DCA algorithm with TCP traffic awareness is more effective than the simple algorithm if you consider that realistically a WLAN device may advertise a smaller TCP window size than $655,350$ bytes. However, the implementation of a cross layer communication between MAC with other layers and the required amendments over existing protocols in real-life are convoluted.

7.3 Design of Selective DCA

A standard IEEE 802.11 device follows the seven-layered OSI Model, thus we must overrule any cross-layer assumptions and settle with the aspect that layer functions and elements are not specified and should be transparent to other layers. The only information that the MAC layer receives from the upper layer, except the actual payload, is the Type of Service (ToS). So, aa a provisional idea will be to exclude the use of DCA from BE flows since large portion of these application utilize the TCP protocol. Then, similar with TCP-Aware, this can be easily implemented by just placing the DCA process into a conditional statement which will intermittently check if the incoming packets are of BE type. Then again, sometimes there are BE flows that are using UDP connections and other real-time applications (AC_VI or AC_VO) that are connected trough TCP links. Therefore, this approach may not be realistic either.

Recent research on Internet traffic classification algorithms has yielded a flurry of proposed approaches for distinguishing types of traffic based on transport layer ports, host behaviour, and flow features [153]. Traditional methods of traffic flow classification relied on the well-known ports registered with Internet Assigned Numbers Authority (IANA) to represent a specific application. However, emerging popular application such as those that support P2P file sharing, started to use arbitrary port numbers in order to hide their identity, so solely based on port numbers alone could lead to inaccurate assumptions [154, 155, 156]. Another more reliable approach is deep packet inspection, a work-around that was mainly used by commercial tools [157, 158] which inspect packet payloads for specific string patterns of known applications [159, 160, 161, 162], provided that the packets are not encrypted. Nevertheless, it is costly, processor and bandwidth resource-thirsty and causes tremendous privacy and legal concerns. Two proposed traffic classification approaches that avoid payload inspection and tend to be more popular are: i) host-behaviour-based, which takes advantage of information regarding "social interaction" of hosts [163, 155, 159], and ii) flow features-based, which classifies based on flow duration, number and size of packets per flow, and inter-packet arrival time [164, 165, 166, 167, 168, 169].

Taking in consideration the aforementioned traffic classification algorithms, we came to the conclusion that the approach with the most prospects is the flow feature-based. So, in order to come up with a potential resolution, we had to identify a pattern for the TCP flows using the three (3) types of classification: a) flow duration, b) number and size of packets per flow, and c) inter-packet arrival time. The inter-packet arrival time was identified from the trace files collected from various OPNET simulations. The traces showed that the TCP segments, assigned in a single queue buffer, were arriving at the MAC layer in a homogeneous Poison process with a constant rate. The period from the point where the first segment arrived at the MAC layer until a second full segment appeared, from the same TCP process, was observed to be precisely 10 μs. In addition, we found that most of the TCP segments complied with the MSS and MTU rules, so we were able to determine the expected packet (segment) length of the payload arriving from the higher layers. Figure 7.4 illustrates the encapsulation method that occurs over each layer with the additional headers and tails. Therefore, for a segment size equal to $1,460$ bytes and following the encapsulation process, the packet that will arrive to the MAC layer will be including a TCP header of 20 bytes, consequently will have a total size of $1,480$ bytes. Also, packets smaller than 44 bytes are mostly TCP's associated packets, such as ACK, SYN, FIN or RST. Last but not least, TCP flows start with a three-way handshake and terminate with a four-way handshake (or a time-out), so we can identify a signature for the flow duration. In conclusion, all these clues are the foundations to design a cognitive agent that will select the incoming flows.

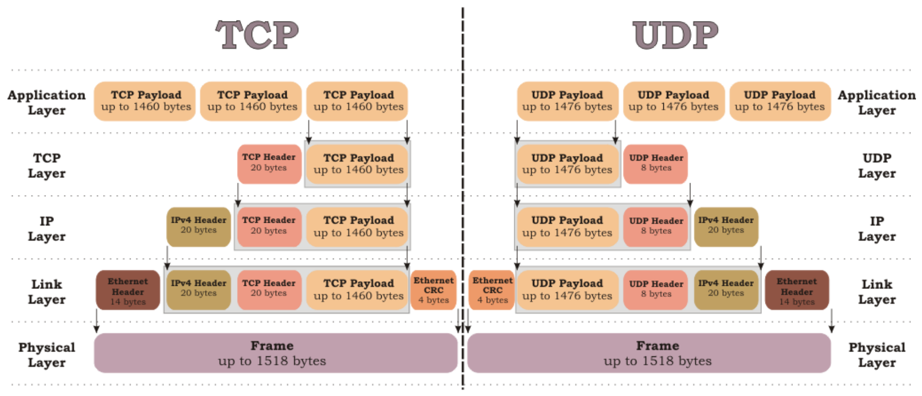


Figure 7.4: Encapsulation Process

The new proposed enhanced DCA method is called SDCA and is fundamentally based on the TCP-Aware with only difference the conditional decisions of determine the type of flow. The new algorithm will try to distinguish the flows by observing its features. So, each AC buffer collects discrete information for each recipient and reviews the flows separately. Within the function there is a constant variable which defines the level of tolerance, meaning that there could be a margin of deviation on the TCP packet rate. In order to decrease the level of misconception, the function also checks the packet sizes. It is known that when a TCP connection establishes there is a three-way handshake, a negotiation between the two nodes where they shared information with specific segments with no data but just the headers. The size of these segments can easily be determined (40 – 44 bytes long) and audited at the beginning and end of the transaction. As a result, whenever a packet shows up at the MAC with length less than 44 bytes then the function increases its level of awareness.

The accuracy of the function's decisions has been tested with a set of traffic patterns and found that if there are any misjudgements on the flow type, there won't be any negative development over the SDCA performance. Let us assume an example where both UDP and TCP belong in the same AC and share the same receiver. Then, the function may not be able to distinguish any distinctness between them two as the next TCP packet will be compared with the last arrived UDP traffic, or vice versa. Nevertheless, the packets in queue increases exponentially and DCA triggers before the interlock situation discussed earlier occurs. Also, it is unusual to receive UDP traffic with a 10 μs inter-arrival rate and in a series of TCP packets, it will definitely have at least two consequent TCP packets. So, once the first packet initiates a transmission sequence, every other following packet (UDP or TCP) that arrives during that time will be transmitted too. Finally, because the nature of TCP traffic is generally BE flows, it will ordinarily have a long AIFS and Back Off timer so it helps the aggregates to increase further without the need of SDCA.

7.4 Test-Bed for Packet Analyser

In order to verify our hypothesis that the inter-segment arrival rate at MAC for TCP flows is within the region of 10 μs, we examined the packet traffic of a small Ad-hoc network. The purpose of these experiments was to establish that aforementioned assumptions, derived from OPNET's trace files, can also be reflected to real networks. We consider a set-up consisting of two (2) laptop computers, choosing as general OS Linux-based platform Ubuntu 8.10 (Intrepid Ibex) [170], and including the following wireless interfaces: i) a Netgear WG511T adapter card [171] and ii) a USRobotics USR5410 adapter card [172] . The STAs operated in a 22MHz frequency band around 2.462GHz, were located in the same room resulting in a high Signal-to-Noise Ration (SNR) between nodes and no hidden terminals. The channel is error-prone, thus few inaccuracies are expected. A schematic of the test-bed set-up is shown in Figure 7.5.

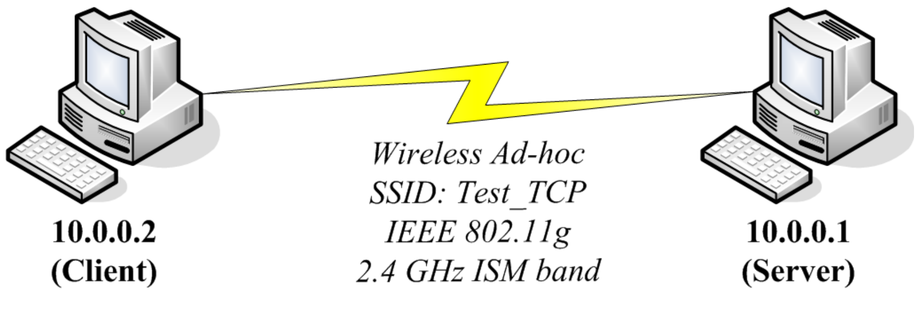


Figure 7.5: A schematic of the test-bed set-up

As we will discuss in detail, our measurements focus on two traffic scenarios. First, we consider UDP traffic from one STA (Client) to the other STA (Server) of constant packet length with Poisson distributed inter-departure times at different rates, depending the application in study taken from *Scenario 2*. Subsequently, we consider the same configuration parameters but with TCP traffic instead. Note that the applications will be checked for both types of traffic although in real scenarios this is not the case, however to validate our results we consider all possible cases.

The traffic was generated using the Distributed Internet Traffic Generator (D-ITG) [173, 174, 175], which allowed us to statistically characterize parameters such as Inter-Departure Times (IDTs) and packet length. For example, in order to generate the traffic for the UDP traffic for Case 1, as displayed in Table 7.1, we instruct D-ITG to generate UDP packets at a rate of $1,600$ packets per second with constant inter-departure times and a constant packet length of $1,458$ bytes. Note that the listed packet size, refers to the packet length of the transmitted frame, consequently when we calculate the packet length for D-ITG, we consider the expected payload length at the Application Layer, thus excluding additional headers.

Case	Flow Name	Protocol	Offered Load	Packet Size	Packets/Sec	IDT
1	HDTV + Futuristic Audio	UDP	19.2 Mb/s	1,500B	1,600	0.000625 sec
2	HDTV + PCM 5.1 Audio	UDP	24 Mb/s	1,500B	2,000	0.0005 sec
3	Internet File	TCP	120 Mb/s	1,500B	10,000	0.0001 sec

Table 7.1: Test-bed traffic generation parameters

Besides the statistics provided by the wireless interface, we used *nstat* to gather IP, UDP and TCP statistics aggregated across all interfaces, so as to check for unexpected network activity during the tests. We have also operated a customized Packet Analyser (sniffer) to record detailed logs of all packets sent and received by the wireless interfaces during each test. Sniffers [176] are programs used to read packets that travel across the network at various levels of the OSI layer. The custom implemented Packet Analyser was able to retrieve specific interface statistics, such as number of packets received and transmitted, inter-arrival times, packet lengths, etc., and histograms, like signal noise levels. The Listing 7.1 displays a snippet of one of the TCP traces. The sniffer's trace files were examined off-line on a case by case basis, so a flow behavioural pattern for TCP and UDP cases could be derived. In order to calculate mean, minimum and maximum values for the statistics and examine the TCP and UDP traces, a specialised parser was implemented as well. Both source codes for the Packet Analyser and the Parser can be found in Appendix Source Codes of Packet Analyser and Parser.

```
01:03:52.143554 IP 10.0.0.2.38221 > 10.0.0.1.9000: S 419399396:419399396(0) win 5840
01:03:52.144533 IP 10.0.0.1.9000 > 10.0.0.2.38221: S 3728503593:3728503593(0) ack 419399397 win 5792
01:03:52.144585 IP 10.0.0.2.38221 > 10.0.0.1.9000: . ack 1 win 92
01:03:52.144636 IP 10.0.0.2.38221 > 10.0.0.1.9000: P 1:2(1) ack 1 win 92
01:03:52.145999 IP 10.0.0.1.9000 > 10.0.0.2.38221: . ack 2 win 91
01:03:52.146741 IP 10.0.0.1.9000 > 10.0.0.2.38221: P 1:2(1) ack 2 win 91
01:03:52.146763 IP 10.0.0.2.38221 > 10.0.0.1.9000: . ack 2 win 92
01:03:52.147124 IP 10.0.0.2.38221 > 10.0.0.1.9000: P 2:34(32) ack 2 win 92
01:03:52.148390 IP 10.0.0.1.9000 > 10.0.0.2.38221: P 2:7(5) ack 34 win 91
01:03:52.184451 IP 10.0.0.2.38221 > 10.0.0.1.9000: . ack 7 win 92
01:03:52.201978 IP 10.0.0.2.8998 > 10.0.0.1.8997: S 422983504:422983504(0) win 5840
01:03:52.203473 IP 10.0.0.1.8997 > 10.0.0.2.8998: S 3729080509:3729080509(0) ack 422983505 win 5792
01:03:52.203525 IP 10.0.0.2.8998 > 10.0.0.1.8997: . ack 1 win 92
01:03:52.208791 IP 10.0.0.2.8998 > 10.0.0.1.8997: P 1:1435(1434) ack 1 win 92
01:03:52.208986 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 1435:2883(1448) ack 1 win 92
01:03:52.210538 IP 10.0.0.1.8997 > 10.0.0.2.8998: . ack 1435 win 136
01:03:52.210580 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 2883:4331(1448) ack 1 win 92
01:03:52.210602 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 4331:5779(1448) ack 1 win 92
01:03:52.211198 IP 10.0.0.1.8997 > 10.0.0.2.8998: . ack 2883 win 181
01:03:52.211226 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 5779:7227(1448) ack 1 win 92
01:03:52.211248 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 7227:8675(1448) ack 1 win 92
01:03:52.213064 IP 10.0.0.1.8997 > 10.0.0.2.8998: . ack 4331 win 227
01:03:52.213092 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 8675:10123(1448) ack 1 win 92
01:03:52.213112 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 10123:11571(1448) ack 1 win 92
01:03:52.215352 IP 10.0.0.1.8997 > 10.0.0.2.8998: . ack 5779 win 272
01:03:52.215379 IP 10.0.0.2.8998 > 10.0.0.1.8997: P 11571:13019(1448) ack 1 win 92
01:03:52.215412 IP 10.0.0.2.8998 > 10.0.0.1.8997: P 13019:14341(1322) ack 1 win 92
01:03:52.216928 IP 10.0.0.1.8997 > 10.0.0.2.8998: . ack 7227 win 317
01:03:52.216955 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 14341:15789(1448) ack 1 win 92
01:03:52.216977 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 15789:17237(1448) ack 1 win 92
01:03:52.218180 IP 10.0.0.1.8997 > 10.0.0.2.8998: . ack 8675 win 362
01:03:52.218207 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 17237:18685(1448) ack 1 win 92
01:03:52.218228 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 18685:20133(1448) ack 1 win 92
01:03:52.219005 IP 10.0.0.1.8997 > 10.0.0.2.8998: . ack 10123 win 408
01:03:52.219031 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 20133:21581(1448) ack 1 win 92
01:03:52.219042 IP 10.0.0.2.8998 > 10.0.0.1.8997: P 21581:23029(1448) ack 1 win 92
01:03:52.220559 IP 10.0.0.1.8997 > 10.0.0.2.8998: . ack 11571 win 453
01:03:52.220610 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 23029:24477(1448) ack 1 win 92
01:03:52.220627 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 24477:25925(1448) ack 1 win 92
01:03:52.221040 IP 10.0.0.1.8997 > 10.0.0.2.8998: . ack 13019 win 498
01:03:52.221066 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 25925:27373(1448) ack 1 win 92
01:03:52.221080 IP 10.0.0.2.8998 > 10.0.0.1.8997: . 27373:28821(1448) ack 1 win 92
01:03:52.221741 IP 10.0.0.1.8997 > 10.0.0.2.8998: . ack 14341 win 543
```

Listing 7.1: Sample of Sniffer's trace file

Simulation Results (Client ↔ AP)						
Flow	HDTV (19.2 Mb/s)		HDTV (24.0 Mb/s)		IF (120.0 Mb/s)	
IAT $\leq$ 10 μs	**UDP**	4.32%	**UDP**	2.64%	UDP	2.87%
	TCP	89.4%	TCP	94.92%	**TCP**	89.92%

Table 7.2: Ad-hoc test-bed results for inter-packet arrival expectancy

For the experiments, we processed five (5) repetitions for each case and then provided as an output the average calculated result from all runs. Table 7.2 displays the ratio for the segments arrived at the MAC layer within the 10 μs ($\pm$ small deviation) interval over the total number of segments. Although the actual HDTV and IF flows are transmitted over the UDP and TCP protocols, respectively, we've also examined cases with opposite transport protocols with the same traffic characteristics (e.g. inter-arrival times, offered loads) for the sake of confidence. For the UDP test, we can observe that percentage of traffic arriving within that specific interval is only 4.32% and 2.64% for the HDTV flows, and 2.87% for the IF. For the TCP traffic, the statistics provided by the wireless interface are proving our initial hypothesis to a huge extend. A large portion of the total segments was recognised as TCP traffic even for flows that have smaller rates, like the HDTV. The results for the TCP traffic over both HDTV and IF, were 89.4%, 94.92% and 89.92%, respectively.

7.5 Performance Evaluation of SDCA with OPNET

In this chapter, we introduced another extension of DCA, named as SDCA. This proposed algorithm will be examining the behaviour of the incoming traffic flows and determine according to the flow pattern if the traffic belongs to TCP or UDP protocols. In the case of a TCP packet arrival, the algorithm will function similar to the operation of TCP-Aware DCA. Within this section, we will evaluate closely the performance of SDCA through various simulations using OPNET. The design and choice of network architecture for each scenario corresponds to a home, a large enterprise and a hot spot environment. Therefore, for the home scenario, we use previously defined *Scenario 1* and *Scenario 2* while for the large enterprise and hot spot layout, we follow usage models *Scenario 4* and *Scenario 6*, respectively. All scenarios use TCP *New Reno* with the receiver's window buffer set at $65,535$ and $655,350$ bytes.

The numerical results of *Scenario 2* in Table 7.3 provide the goodput and average delay outcomes of all accommodate traffic for low and high rwnd when SDCA is disabled and enabled. For all HDTV flows, the offered load is transmitted fully in every single case and while having slightly longer delays than those without SDCA are still delivered well below the allowed maximal delay of 200 ms. For the BE traffic, ADCA provides a better channel utilization and results in a

SDCA	TCP rwnd	Traffic	Offered Load	Goodput	Avg. Delay
Off	655,350	HDTV	19.2	19.197	0.00100
		HDTV	24	23.994	0.00115
		Internet File	120	11.796	0.39693
On	655,350	HDTV	19.2	19.074	0.01488
		HDTV	24	23.904	0.01341
		Internet File	120	50.343	0.09721
Off	65,535	HDTV	19.2	19.2	0.00087
		HDTV	24	23.997	0.00100
		Internet File	120	9.714	0.03194
On	65,535	HDTV	19.2	19.062	0.01490
		HDTV	24	23.994	0.01294
		Internet File	120	47.517	0.00638

Table 7.3: SDCA results with large and small rwnd for Scenario 2

throughput of 50.343 Mb/s and 47.517 Mb/s for high and low rwnd, respectively. Out of 163.20 Mb/s total offered load, the new algorithm achieves 93.32 Mb/s and 90.57 Mb/s total goodput, an efficiency of 57.18% and 55.5% for each rwnd case.

Scenario	TCP rwnd	SDCA	UP	Off. Load		Goodput		Avg. Aggr.	Avg. Delay	PLR
1	655,350	Off	BE	83.524	31	59.362	7.357	26.73	0.36558	N/A
			VI		50.448		49.93	8.93	0.02217	0
			VO		2.076		2.075	1.19	0.00271	0
		On	BE	83.524	31	76.946	24.836	28.79	0.05401	N/A
			VI		50.448		50.036	13.38	0.04389	4.33
			VO		2.076		2.074	2.7	0.00661	2.71
	65,535	Off	BE	83.524	31	56.849	4.636	19.25	0.05536	N/A
			VI		50.448		50.139	6	0.01374	0
			VO		2.076		2.075	1.17	0.00257	0
		On	BE	83.524	31	81.944	29.764	22.87	0.03192	N/A
			VI		50.448		50.104	13.61	0.04409	3.4
			VO		2.076		2.075	2.7	0.00651	2.8

Table 7.4: SDCA results with large and small rwnd for Scenario 1

Table 7.4 shows the computed results for *Scenario 1* for SDCA. Notice the differences between the values for SDCA enabled with a system set with regular IEEE 802.11n complied operation without any enhancements. The mean aggregate sizes for AC_VI and AC_VO traffic flows with TCP rwnd set to the lower value, grow notably on average from 6.00 packets to 13.61 packets and from 1.17 packets to 2.69 packets, respectively. Hence, as the channel utilization for the higher ACs increases we would assume that the overall goodput must be improved significantly too. Actually the performance data unquestionably proves this conjecture since the system's overall goodput for a realistic TCP buffer size of $65,535$ bytes, boosts from 56.85 Mb/s to 81.94 Mb/s which is a 44.13% increase. Furthermore, all multimedia packets, while having slightly longer delays than those without SDCA, are still delivered well below the allowed maximal delay (200 ms). For example, the mean delays for the AC_VI packets are 44 ms that is around 22% of the delay boundary. Furthermore, we can observe an increase to the PLR ratios but nevertheless this scenario proves SDCA effective of improving the system's goodput with no QoS suffering.

Scenario	TCP rwnd	SDCA	UP	Off. Load		Goodput		Avg. Aggr.	Avg. Delay	PLR
4	655350	Off	BE	460.176	451.024	62.061	52.909	42.95	0.63252	N/A
			VI		8		8.001	2.66	0.00766	0
			VO		1.152		1.151	1.13	0.00522	0.2
		On	BE	460.176	451.024	85.149	76.091	42.2	0.40677	N/A
			VI		8		7.906	18.24	0.05222	0
			VO		1.152		1.151	1.79	0.00923	0.7
	65535	Off	BE	460.176	451.024	58.69	49.569	38.70	0.11170	N/A
			VI		8		7.969	2.69	0.00772	0
			VO		1.152		1.152	1.13	0.00529	0.2
		On	BE	460.176	451.024	83.076	74.013	39.89	0.07549	N/A
			VI		8		7.91	19.60	0.05303	0
			VO		1.152		1.151	1.74	0.00889	0.3

Table 7.5: SDCA results with large and small rwnd for Scenario 4

Scenario	TCP rwnd	SDCA	UP	Off. Load		Goodput		Avg. Aggr.	Avg. Delay	PLR
6	655,350	Off	BE	64.88	20	47.878	7.857	53.73	1.11206	N/A
			VI		42		37.144	40.24	0.23886	59.22
			VO		2.88		2.8783	1.44	0.00863	2
		On	BE	64.88	20	57.903	13.298	57.97	0.70561	N/A
			VI		42		41.728	25.54	0.03705	0
			VO		2.88		2.878	1.92	0.01030	2.1
	65,535	Off	BE	64.88	20	49.415	10.5	30.96	0.23790	N/A
			VI		42		36.042	39.92	0.29321	65.89
			VO		2.88		2.871	1.44	0.00886	2.3
		On	BE	64.88	20	61.165	16.438	41.79	0.19522	N/A
			VI		42		41.849	24.88	0.03655	0
			VO		2.88		2.877	1.92	0.01030	2.1

Table 7.6: SDCA results with large and small rwnd for Scenario 6

Table 7.5 and Table 7.6 show the simulation results for SDCA for *Scenario 4* and *Scenario 6*, respectively. Let us start describing the outcomes for the TCP buffer sizes equal to $655,350$ bytes. We observe that for the BE traffic flows for for *Scenario 4*, there is an increase in the overall goodput from 62.06 Mb/s to 85.15 Mb/s and similar to for *Scenario 6* from 47.88 Mb/s to 57.90 Mb/s. Especially, for the TCP flows in *Scenario 4*, the throughput escalates with an extraordinary raise of 43.8%. Furthermore, the expected PLR for VI flows is 0% in both scenarios and only for VO flows is 0.7% and 2.1%, respectively, but yet again is less than the allowed maximal PLR of 5%. Also, SDCA's key role in increasing multimedia performance can be noted when comparing the PLRs values for VI applications in *Scenario 6*. With SDCA set to disable, the network fails to deliver in time 59.22% of the total video flows, on the other hand when SDCA is applied, all packets are received successfully and the PLR drops to negligible rates. For the case of a low TCP rwnd the PLR is much higher and yet when SDCA kicks off the results show no drops. All multimedia flows meet their QoS provisions even though we choose to defer further the channel access scheduling mechanism. This is because SDCA manages to increase the aggregate sizes for high priority flows and hence uses the wireless medium efficiently. More specific, in *Scenario 4* the VI flows' aggregate frame sizes have gone up from 2.69 (SDCA Off) to 19.60 (SDCA on) packets and there is a similar augmentation for the VO frames too. However, we can still see a decrease of the aggregated size of the VI flows in *Scenario 6* but this is normal since SDCA has stabilized the 802.11e's probabilistic priority mechanism and both multimedia can fairly share the wireless medium effectively, on that account the immediate drop over the PLR ratings.

7.6 Summary

By using a simple function that analyses and records the flow of packets arriving at the MAC layer we are able to specify the packet's type of transportation protocol that uses. A simple empirical analysis of the traffic flow behaviour can provide a distinguished pattern that the selection process can be based on. The efficiency of the proposed operation has been demonstrated in a real test-bed and the outcomes have shown relative collateral results. A small marginal differentiation over the comparisons between the test-bed and the modelled simulation runs shall be linked to environmental conditions, as the test-bed operated in a error-prone channel.

We use more than one scenario to evaluate our proposal and prove that the SDCA algorithm improves the system performance significantly. Based on the above results, we can claim that the SDCA fixes the significantly negative performance impact by the poor interaction between EDCA and IEEE 802.11n and in addition it can effectively confine the TCP problem for smaller TCP window buffer sizes as well.

Chapter 8

Conclusions

The latest IEEE 802.11n standards attains rates of more than 100 Mb/s by introducing innovative enhancements at the PHY and MAC layer, e.g. MIMO and Frame Aggregation, respectively. However, in this thesis we demonstrated that frame aggregation's performance adheres due to the EDCA scheduler's priority mechanism and consequently resulting in the network's poor overall performance. Short waiting times for high priority flows into the aggregation queue resolves to poor channel utilization. A Delayed Channel Access algorithm was designed to intentionally postpone the channel access procedure so that the number of packets in a formed frame can be increased and so will the network's overall performance. However, in some cases, the DCA algorithm has a negative impact on the applications that utilize the TCP protocol, especially the when small TCP window sizes are engaged. So, the TCP process starts to refrain from sending data due to delayed acknowledgements and the overall throughput drops. The main objectives to this research work is to introduce innovative resolutions to the problem.

8.1 Research Outcomes

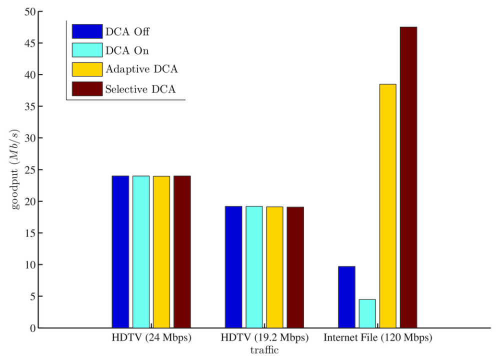
We started the thesis by indicating the throughput limitations of the archetype IEEE 802.11 standard and how these can be overcome with the use of Frame Aggregation. Then we described in detail the Frame Aggregation mechanism and validate the beneficial advancements of this new MAC enhancement over the networks overall performance through extensive simulations using a HT model in OPNET. We then developed an analytical model to explicitly illustrate the impact of Frame Aggregation, which from the mathematical analysis we derived the assumption that for networks with low traffic intensity, we need to intentionally defer the channel access procedure in order to introduce additional packets in the aggregate buffer. However, a trade-off choice has to be taken into account regarding the question which viewpoint is more important, minimizing the waiting time or maximizing utilization, an algorithm which would be able to dynamically control the aggregation buffer may be valuable.

Afterwards, we discussed and demonstrated the luck of performance improvement of IEEE 802.11n in conjunction with IEEE 802.11e prioritization mechanisms. Each AC is parametrised with different set of values, so higher priority traffic has certain parameters to allow it to gain access to the channel earlier and more often than the lower priority traffic. Based on computer model analysis, we demonstrated the magnitude of impact that high ACs have over lower ACs. Most of the simulated use case scenarios had been chosen from TGn's *Usage Models* document [68], in addition of a customized home scenario called as *Scenario 2*, and all were tested thoroughly over various conditions. Next, we introduced DCA, an algorithm that provides great improvement over the total throughput by deferring the transmission for all flows, including high ACs, while still obeying all QoS requirements. Results showed that for all contenting entities, there was a huge gain over the channel utilization and the total throughput as the algorithm repeatedly allows newly packets to tail the aggregate buffer queue till certain conditions, predefined in the DCA mechanism, are met.

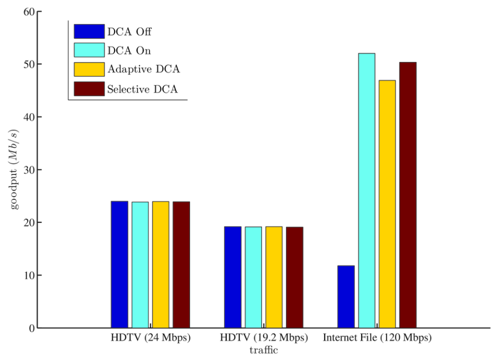
Later, we tested the original DCA mechanism with various TCP Window sizes since previous results were set with the suggestive BDP value and not with a more realistic value. The results affirmed that the reduction of the rwnd in conjunction with the DCA functionality produced unfavourable consequences over the network's performance measurements. The source of the problem, it was found to be a dead-lock situation caused by the buffering queues integrated in the MAC and TCP layer. The operational actions of both queues let their mechanisms to wait for a certain packet before it carried on; the MAC layer was expecting an extra segment from the upper layers while the TCP protocol was holding for an acknowledgement.

Then, we went on adapting the basic DCA and extending it further in order to provide support for both UDP and TCP protocols. So, we first considered an adapting sizing algorithm that is based on certain measurements of the current and previous packet traffic, and feedback from DCA's triggering mechanism. And our second approach was to classify with the aid of a cognitive agent the type of transportation protocol that flows use based on duration, number and size of packets per flow, and inter-packet arrival time. We designed and implemented two distinct enhancements for DCA, known as Adaptive DCA and Selective DCA. Each has a different operational approach but the end objective is the same. The effectiveness of these algorithms was demonstrated via extensive simulations and experimental measurements. For the test-bed experiments a Packet Analyser was implemented in C language.

Figure 8.1 and Figure 8.2 summarise the results of ADCA and SDCA for *Scenario 2* with TCP Window size set to $65,535$ bytes and $655,350$ bytes. For comparison reasons, the performance results of original DCA and HT model without an extensions have been included too. From the goodput results of the consisting applications for large TCP rwnd, we observe the benefits of the additional deferment on channel access, illustrated by the DCA based cases. On the other hand, for small TCP rwnd, the original DCA presents an unfavourable behaviour by dropping the TCP throughout from 9.71 Mb/s to 4.49 Mb/s, while for ADCA increases at 38.454 Mb/s and for SDCA rises even further at 47.517 Mb/s. Similar behaviour show the mean delay outcomes, with the original DCA algorithm falling behind with higher delays for the TCP flows. It is important to mention that with the incorporation of any DCA extension, some flows will encounter additional mean delays. For example the HDTV traffic with 24 Mb/s and TCP Window size set to $655,350$ bytes, the derived mean delay is 0.00115 sec, 0.01342 sec, 0.00824 sec and, 0.01341 sec, given in the same order as displayed. So, there is a huge increase in the time that the packets wait in the aggregation buffer which in both DCA and SDCA cases has ascents up to 11.6 times. In conclusion, we can be certain of choosing which DCA extension is preferable to utilize for best improvements, ADCA or SDCA. Both enhancements operate in a different manner but have the same objectives and any one of them may surpass the other's performance in a different case scenario.

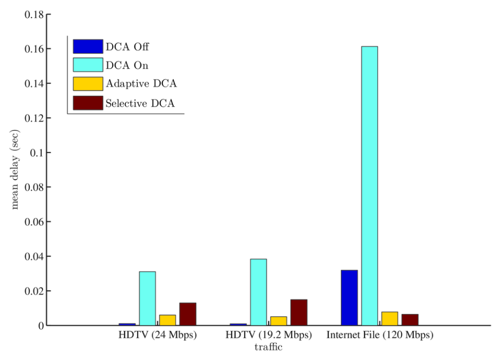


(a) TCP rwnd set to 65,535 bytes

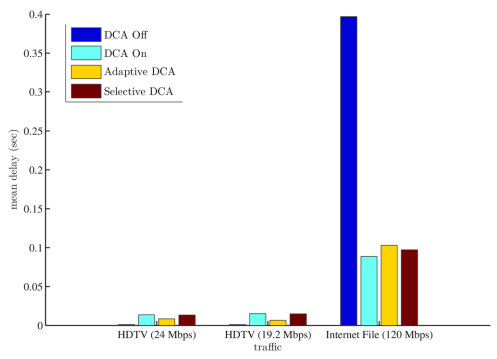


(b) TCP rwnd set to 655,350 bytes

Figure 8.1: Goodput for low and high rwnd using various DCA enhancements



(a) TCP rwnd set to 65,535 bytes



(b) TCP rwnd set to 655,350 bytes

Figure 8.2: Average Delay for low and high rwnd using various DCA enhancements

8.2 Future Research Directions

Now that the activities of 802.11n standard have been finalized and IEEE 802.11 wireless cards are available, we can deploy a real test-bed and retrieve performance measurements for various scenarios. In addition, the open-source community has devoted huge efforts to provide Linux based firmwares, such as carl9170 [177], ar9170 [178] and brcm80211 (by Broadcom) [179]. Open Source firmwares are suitable for providing the easiest approach to implement an experimental method while at the same time supporting a great number of functionalities within the framework of the respective hardware platform used.

The proposed ADCA and SDCA can be extended even further. Both algorithms, in their own manner, are trying to speculate the presence of a TCP flow and determine its parameters, e.g. TCP Window size. For the SDCA extension, we propose a basic but still innovative flow-features based classification mechanism with evaluation experiments showed positive results. Additional classifying methods have been introduced, such as Support Vector Machine (SVM) algorithm, which achieves a 98.0% accuracy on every trace and application[153]. The basic principle of SVM is to construct the optimal separating hyperplane, which maximizes the distance between the closest sample data points in the (reduced) convex hulls for each class, in an n-dimensional feature space[180]. Therefore, DCA is a subject of extended research that could easily improve performance even further by implementing new approaches.

The initial DCA algorithm was firstly introduced as a concept in *TGnSync Proposal Technical Specification Document* [181]. TgnSync was one of the main industry associations that took part in the development of the IEEE 802.11n standard. The standard's specification document had to undergo through many changes and various balloting sessions before eventually reach its final form as we know it today. Usually during a standardization process, there are multiple issues that to have to be addressed and negotiated between the Task Group's members, mainly technical differences but also licensing of intellectual property which then could lead to endless conflicts. Nevertheless, when TGnSync and another consortium, known as WWiSE, collaborated together, the DCA proposal was dropped from their joint specifications document. The reasoning is unknown but we assume that DCA design was on its initial phase and hadn't been researched thoroughly. Evidence of the issues that the algorithm imposes is the TCP Window size problem, also discussed in this text. However, the evolution of the HT wireless broadband networks still continues with the emerging development of IEEE 802.11ac and IEEE 802.11ad. Both end specifications will aim to enable multi-station WLAN communication at multi-gigabit speeds. Frame aggregation will still be a key technology for this future standards and so can the inclusion of well defined DCA algorithm.

Final but not least, since energy-related considerations are gaining popularity in wireless networks, especially for mobile devices, there is a tremendous interest in energy efficiency. The main transmission technique in 802.11n is utilizing the MIMO technology, enabling the use of multiple sending and receiving antennas with the objective of providing high rates but resulting in higher power consumption too. Early studies have shown that enhanced frame aggregation schemes that increase channel utilization while supporting robust frame delivery, can also reduce the energy cost for wireless devices [182]. The improvements from the DCA operation can also be compared with possible gains over the power consumption as well.

Appendix A

Usage Models

A general definition for the term *"Usage Model"* is given within IEEE's *Usage Models* documentation for the emerging 802.11n amendment [68]. According to 802.11 TGn, usage model is a specification of one or more applications and environments from which a simulation scenario can be created once the traffic patterns of the applications are known. A use case is a description of how end users uses an application, such as HDTV, video streaming, internet transfer, VoIP and etc. and how these users are deployed over the system. In general, usage models are created to cover various market-based use-cases and intend to support the definitions of network simulations that will allow 802.11 TGn to evaluate performance of various proposals in terms of network throughput and goodput, delay, packet loss and other metrics.

The following usage models are enumerated according to [68] and brief descriptions and definitions are provided.

A.1 Model 1 - Residential

The first scenario represents an indoor (room to room) residential network with several HT devices. Wireless connectivity has been spread over a residential platform for a long time now, a distinguished example is the use of cordless telephones that can provide the flexibility to move around the house and have conversations on the phone with minimum jitter. Nowadays, more and more home wireless devices are being used and further more are being developed for the near future. By introducing higher data rates and QoS, users will be able to view SDTV and HDTV anywhere in the house and simultaneous talk on their VoIP telephones, surf on the Internet, listening to MP3 music that is stored on a central wireless unit or even playing games on-line via their wireless consoles.

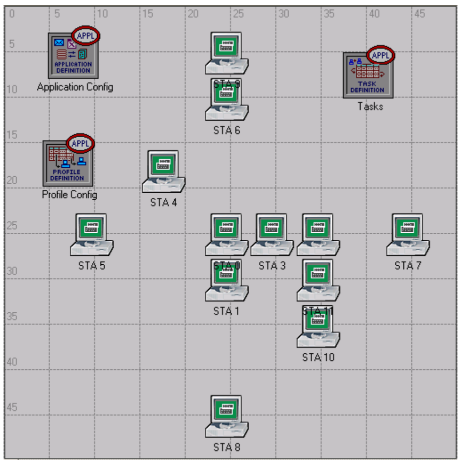


Figure A.1: Spatial distribution in OPNET for Usage Model 1

STA Name	Role	Dest. STA	Mean Rate	Rate Distrib.	MSDU	Delay	Application
AP	Access Point	STA 1	19.2 Mbps	Constant, UDP	1,500 B	200 ms	HDTV
		STA 3	24 Mbps	Constant, UDP	1,500 B	200 ms	HDTV
		STA 4	4 Mbps	Constant, UDP	1,500 B	200 ms	SDTV
		STA 4	1 Mbps	TCP	300 B		Internet File
		STA 7	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
		STA 8	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
		STA 9	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
		STA 10	2 Mbps	UDP	512 B	200 ms	Internet Streaming
		STA 11	0.128 Mbps	UDP	418 B	200 ms	MP3 Audio
STA 1	HDTV Display	AP	60 kbps	Constant, UDP	64 B	100 ms	VoD Control Channel
STA 3	HDTV Display	AP	60 kbps	Constant, UDP	64 B	100 ms	VoD Control Channel
STA 4	SDTV Display, Gaming & Printing	STA 10	30 Mbps	Constant, TCP	1,500 B		Local File Transfer
STA 5	Video Phone	STA 6	0.5 Mbps	Constant, UDP	512 B	100 ms	Video
STA 6	Video Phone & Internet Upload	STA 5	0.5 Mbps	Constant, UDP	512 B	100 ms	Video
STA 7	VoIP Phone	AP	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
STA 8	VoIP Phone	AP	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
STA 9	VoIP Phone	AP	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
STA 10	Video Console & Internet Entertainment	AP	1 Mbps	Constant, UDP	512 B	50 ms	Console to Internet
STA 11	Video Gaming Controller	STA 10	0.5 Mbps	Constant, UDP	50 B	16 ms	Controller to Console

Table A.1: Role and configuration for each STA for Usage Model 1

The main role that each device possesses during this scenario can be found in Table A.1. Some of these stations may operate more than one application depending on their functionalities. The complexity of the scenario's configuration increases while we consider direct links between STAs and with the AP who also acts as a flow coordinator. The spatial distribution for the stations over a residential plot of 20m range is shown in Figure A.1.

A.2 Model 4 - Large Enterprise

With more than 70 percent of enterprises upgrading or deploying WLANs in the 21st century, comprehensive management of the wireless network is a top priority to limit burgeoning operational costs. Business is one of the major customers for the wireless manufacturing. Therefore, an additional attention and concern comes over the surface regarding over how this new technology is going to affect enterprise market.

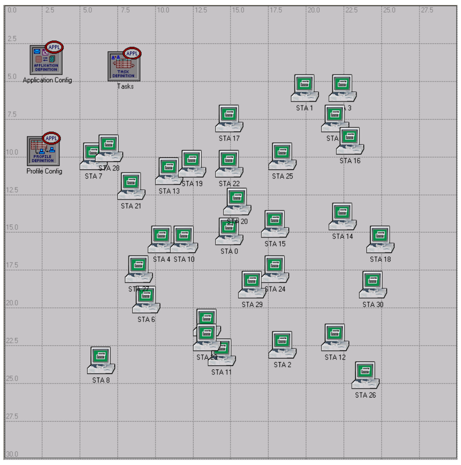


Figure A.2: Spatial distribution in OPNET for Usage Model 4

The following scenario illustrates a large enterprise network (Fig. A.2). It holds 30 stations which are randomly deployed over an indoor area of 300 square meters (m^2). It has an infrastructure topology and its AP is located in the middle as it covers all stations in range. A considerable difference from the residential scenario is that we are expecting from the AP to participate more as a bridge

STA Name	Role	Dest. STA	Mean Rate	Rate Distrib.	MSDU	Delay	Application
AP	Access Point	STA 1	1 Mbps	TCP	300 B		Internet File
		STA 2	1 Mbps	TCP	300 B		Internet File
		STA 3	1 Mbps	TCP	300 B		Internet File
		STA 4	1 Mbps	TCP	300 B		Internet File
		STA 5	1 Mbps	TCP	300 B		Internet File
		STA 6	10 Mbps	TCP	300 B		Internet File & DLing Large Email Attachments
		STA 7	1 Mbps	Constant, UDP	512 B	100 ms	Video Conferencing
		STA 8	1 Mbps	Constant, UDP	512 B	100 ms	Video Conferencing
		STA 9	2 Mbps	UDP	512 B	200 ms	Internet Streaming & MP3 Audio
		STA 10	2 Mbps	UDP	512 B	200 ms	Internet Streaming & MP3 Audio
		STA 11	30 Mbps	TCP	1,500 B	200 ms	Local File Transfer
		STA 12	30 Mbps	TCP	1,500 B	200 ms	Local File Transfer
		STA 13	30 Mbps	TCP	1,500 B	200 ms	Local File Transfer
		STA 14	30 Mbps	TCP	1,500 B	200 ms	Local File Transfer
		STA 15	30 Mbps	TCP	1,500 B	200 ms	Local File Transfer
		STA 16	30 Mbps	TCP	1,500 B	200 ms	Local File Transfer
		STA 17	30 Mbps	TCP	1,500 B	200 ms	Local File Transfer
		STA 18	30 Mbps	TCP	1,500 B	200 ms	Local File Transfer
		STA 19	30 Mbps	TCP	1,500 B	200 ms	Local File Transfer
		STA 20	30 Mbps	TCP	1,500 B	200 ms	Local File Transfer
		STA 25	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
		STA 26	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
		STA 27	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
		STA 28	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
		STA 29	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
		STA 30	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
STA 1	Web Browsing	AP	0.256 Mbps	TCP	64 B		Clicking Web Links
STA 2	Web Browsing	AP	0.256 Mbps	TCP	64 B		Clicking Web Links
STA 3	Web Browsing	AP	0.256 Mbps	TCP	64 B		Clicking Web Links
STA 4	Internet File	AP	5 Mbps	TCP	1,000 B		UPLing Internet File
STA 5	Internet File	AP	10 Mbps	TCP	1,500 B		UPLing Internet File
STA 6	Web Browsing	AP	0.256 Mbps	TCP	64 B		Clicking Web Links
STA 7	Video Conferencing	AP	1 Mbps	Constant, UDP	512 B	100 ms	Video Conferencing
STA 8	Video Conferencing	AP	1 Mbps	Constant, UDP	512 B	100 ms	Video Conferencing
STA 9 - 20	Sink Node	AP					None
STA 21 - 24	Local File Transfer Source	AP	30 Mbps	TCP	1,500 B		Local File Transfer
STA 25 - 30	VoIP Phone	AP	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP

Table A.2: Role and configuration for each STA for Usage Model 4

with the backbone wired network and not just as a coordinator. The applications used in this scenario are representing user requests that could be running over a normal enterprise network (Table A.2).

A.3 Model 6 - Hot Spot

The hotspots are locations with public wireless access points where allow connections from mobile computers, such as a laptop, a tablet or a smart phone to use the Internet through the wireless medium. Hotspots are often found near restaurants, train stations, airports, cafs, libraries and other public places and usually are free to access. The level of simplicity for a user to associate and connect in such an environment may result to a headache for the network administrator as it is rather difficult to manually control the traffic.

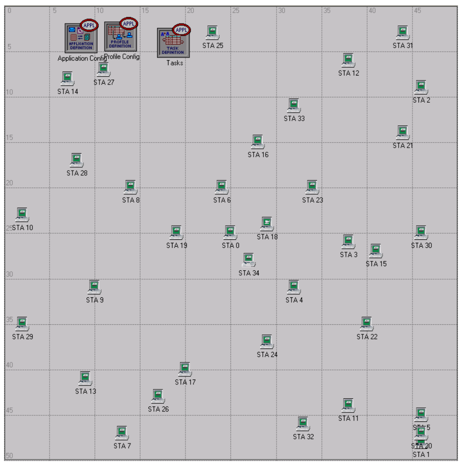


Figure A.3: Spatial distribution in OPNET for Usage Model 6

Because of the importance of such a network deployment, the following scenario represents a Hotspot situation. Again the roles that each station exercises are given from the Table A.3. Until recently, clients were mainly interested into internet file transfers, web browsing and email messaging are included, but nowadays VoIP has developed into a very useful application since it provides cheap global telephony. Thus, a large number of VoIP users had also to be considered in this scenario.

The sketch (Fig. A.3) displayed here, shows the exact location of each correlated STAs which are also fixed deployed over the area given by the specifications. In a real-life scenario there should be frequent associations and de-associations from new and existed clients respectively. However, in our case we assume that

STA Name	Role	Dest. STA	Mean Rate	Rate Distrib.	MSDU	Delay	Application
AP	Access Point	STA 1 - 10	2 Mbps	TCP	300 B		Internet File Transfer
		STA 11 - 14	2 Mbps	UDP	512 B	200 ms	Mid Quality Audio & Video Streaming
		STA 15 - 17	8 Mbps	UDP	512 B	200 ms	High Quality Audio & Video Streaming
		STA 18 - 19	5 Mbps	UDP	1,500 B	200 ms	SDTV Broadcast
		STA 20 - 34	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP
STA 20 - 34	VoIP Phone	AP	0.096 Mbps	Constant, UDP	120 B	30 ms	VoIP

Table A.3: Role and configuration for each STA for Usage Model 6

all STAs remain connected with the Independent Basic Service Set (IBSS) for the whole simulation period and there are no new STAs ask permission for association.

A.4 Model 17 - Point-to-Point High Throughput Goodput Test

In this scenario we will be testing the efficiency of aggregation over two HT stations, both operating in the 20 MHz channel range. The first station has a compound data source providing 100 Mb/s offered load using the UDP protocol and an MSDU size of $1,500$ bytes as seen in Table A.4.

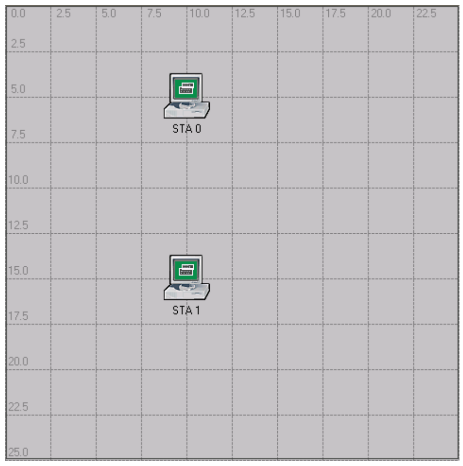


Figure A.4: Spatial distribution in OPNET for Usage Model 17

STA Name	Role	Dest. STA	Mean Rate	Rate Distrib.	MSDU	Delay	Application
AP	HT AP						Sink Node
STA	HT Source	AP	100 Mbps	Constant, UDP	1,500 B		Traffic Generator

Table A.4: Role and configuration for each STA for Usage Model 17

Note, that unlike other UDP sources, these UDP sources have no time-out values specified. Basically, in order to perform plain throughput analysis with OPNET, we use a simple node model that consists from two basic processes, the source which generates compound traffic and the sink that destroys received traffic. Obviously, additional processes for the MAC and PHY layers will be essential. Finally, the stations are placed over a distance of 10 meters and they are in a LoS arrangement as can been seen from Figure A.4.

A.5 Model 18 - Point-to-Point Legacy Throughput Test

The following scenario illustrated in Figure A.5 describes the occasion where a legacy device chooses to utilize an IEEE 802.11n compliant AP. In order to review the consequences of this particular situation regarding the network's and nodes' performance, we need to keep a simple structure that includes only the two participating nodes. Again, both operate over a 20 MHz wide channel, using the same parameters as before and they are 10 metres away. Table A.5 presents the parameters of each STAs set up within the scenario. Our main concern over this simulation is to determine if the AP can be able to operate sufficiently with a wireless node that follows the legacy IEEE 802.11 standard.

STA Name	Role	Dest. STA	Mean Rate	Rate Distrib.	MSDU	Delay	Application
AP	HT AP						Sink Node
STA	Legacy Source	AP	100 Mbps	Constant, UDP	1,500 B		Traffic Generator

Table A.5: Role and configuration for each STA for Usage Model 18

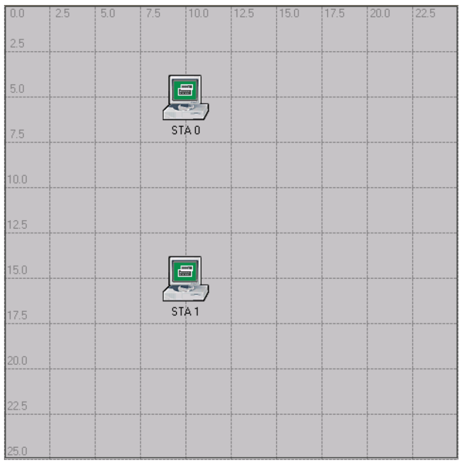


Figure A.5: Spatial distribution in OPNET for Usage Model 18

A.6 Model 19 - Point-to-Point Legacy Sharing Throughput Test

Within this model case, we are going to investigate a scenario where two separate STAs, one HT and one STA belonging to the legacy standard, concurrently operate with a HT AP. This scenario examines the effect, if any, of the multi-rate fairness problem caused by a STA with lower PHY rate. Previous studies show that a co-existence of two diverse data rate STAs will reduce the performance of the upper date rate down to the network's bottleneck, also known as multi-rate fairness problem. For example, it has been demonstrated that an IEEE 802.11g based network will achieve less throughput, in many cases halved, when an IEEE 802.11b compliant shares the same resources with other IEEE 802.11g applicable nodes [73, 71].

STA Name	Role	Dest. STA	Mean Rate	Rate Distrib.	MSDU	Delay	Application
AP	HT AP						Sink Node
STA 1	Legacy Source	AP	100 Mbps	Constant, UDP	1,500 B		Traffic Generator
STA 2	HT Source	AP	100 Mbps	Constant, UDP	1,500 B		Traffic Generator

Table A.6: Role and configuration for each STA for Usage Model 19

To simulate this, we place both STAs beside the AP over equivalent distances as illustrated in Figure A.6 and configure them with the exact same parameters

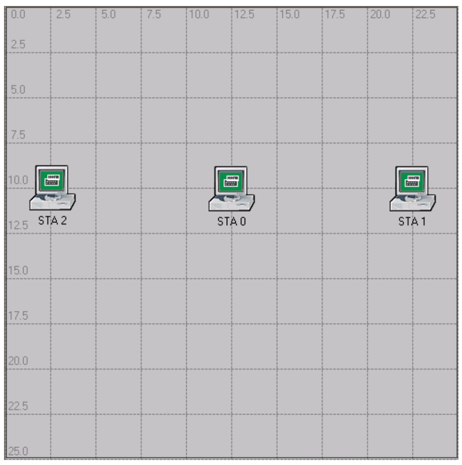


Figure A.6: Spatial distribution in OPNET for Usage Model 19

shown in Table A.6. The AP acts as a data information recipient while the adjacent wireless nodes send compound traffic of similar packet but utilizing heterogeneous transmission techniques.

Appendix B

Source Codes of Packet Analyser and Parser

B.1 Packet Analyser

```
/*
   **************************************************************

 *
 * Purpose: A simple packet analyser application
 * Author:  D. Skordoulis
 * Version: 0.8
 * Date:    03-Feb-09
 *
 **************************************************************
    */

#include <stdio.h>    // standard I/O (e.g. printf)
#include <stdlib.h>   // standard library (eg. exit)
#include <string.h>   // manipulate strings
#include <errno.h>    // error reporting mechanism

#include <sys/socket.h> // declaration of socket constants,
    types and functions
#include <sys/types.h>  // needed for bind function
#include <sys/time.h> // needed for the time function
#include <sys/ioctl.h>  // I/O control - ifreq struct
```

```c

#include <linux/ip.h> // IP hdr structure
#include <linux/tcp.h>  // TCP hdr structure
#include <linux/udp.h>  // UDP hdr structure
#include <net/if.h>   // contains the interface's name length
#include <netinet/in.h> // IPPROTO declarations

#include <features.h> // for the glibc version number
#if __GLIBC__ >= 2 && __GLIBC_MINOR >= 1
#include <netpacket/packet.h> // packet structures
#include <net/ethernet.h>   // the L2 protocols
#else
#include <asm/types.h>
#include <linux/if_packet.h>
#include <linux/if_ether.h>   // the L2 protocols
#endif

/* Define symbolic constant declarations */
#define MAXBUFFSIZE 1514  // Ethernet Frame Length (also consider 1518(1522) or 2048)
#define IF_NAME "wlan0"   // Network's Interface Name (other eth0 & ath0)
#define MAC_ADDR_LEN 6    // Length of the MAC address
#define DEBUG 0       // Debug Mode (0 = false / 1 = true)

/* Declare new data types
typedef unsigned char MAC[MAC_ADDR_LEN];  // a character structure for the MAC address

/* Function that prints a number in HEX format */
void print_in_HEX(unsigned char *element, char dlm, int len) {
  while (len--) {
    printf("%.2X%c", *element, (len != 0 ? dlm : '\0'));
    element++;
  }
}

/* Main Function */
int main(int argc, char *argv[]) {
  // Initialize variables
  int sockfd, numbytes, count, seconds;
```

```
struct timeval curr_time;
struct sockaddr_ll my_addr, rcv_addr;
struct ifreq ifr;
struct ethhdr *eth_hdr;
struct iphdr *ip_hdr;
struct tcphdr *tcp_hdr;
struct udphdr *udp_hdr;
socklen_t addr_len;
FILE *fp;
char *outputFilename = NULL;
void* buffer = (void*)malloc(MAXBUFFSIZE); // Buffer for
   ethernet frame

/* For LAN Interfaces
  192.168.0.2 - 00:0c:76:fb:18:34
  192.168.0.4 - 08:00:46:cf:1c:ad

  MAC sta_1 = {0x00, 0x0C, 0x76, 0xFB, 0x18, 0x34};
  MAC sta_2 = {0x08, 0x00, 0x46, 0xCF, 0x1C, 0xAD};
*/

/* For WLAN Interfaces
  10.0.0.1 - 00:07:CA:03:0F:D7
  10.0.0.2 - 00:09:5b:c5:00:81 - atheros
  10.0.0.2 - 00:C0:49:53:98:48 - acx
*/

// Assign the MAC address of the interfaces in use
MAC sta_1 = {0x00, 0x07, 0xCA, 0x03, 0x0F, 0xD7};
//MAC sta_2 = {0x00, 0x09, 0x5B, 0xC5, 0x00, 0x81};
MAC sta_2 = {0x00, 0xC0, 0x49, 0x53, 0x98, 0x48};


// Create the Socket
if((sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL)
   )) == -1) { // not ETH_P_IP
  perror("socket");
  exit(1);
}
if (DEBUG) printf("Socket created\n");

// Obtaining interface index
strncpy(ifr.ifr_name, IF_NAME, 16);
```

```
if(ioctl(sockfd, SIOCGIFINDEX, &ifr) == -1) {
  perror("IF index");
  exit(1);
}
if (DEBUG) printf("Interface_ID ::: %d\n", ifr.
   ifr_ifindex);

// Assign the values to the sockaddr structure
memset(&my_addr, '\0', sizeof(struct sockaddr_ll));
my_addr.sll_family = AF_PACKET;
my_addr.sll_protocol = htons(ETH_P_ALL);
my_addr.sll_ifindex = ifr.ifr_ifindex;
addr_len = sizeof(struct sockaddr_ll);

if (DEBUG) printf("Address Length (sockaddr_ll) ::: %d\n"
   , addr_len);

// Bind the interface
if (bind(sockfd, (struct sockaddr*) &my_addr, addr_len)
   == -1) { // casting &my_addr
  perror("bind");
  printf("Binding Error ::: %s\n", strerror(errno));
  exit(1);
}
if (DEBUG) printf("Socket bounds to device ::: %d\n",
   my_addr.sll_ifindex);
if (DEBUG) printf("MAXBUFFSIZE ::: %d\n", MAXBUFFSIZE);

count = 0;

// Set memory allocation for receiver's address and timer
memset(&rcv_addr, '\0', sizeof(struct sockaddr_ll));
memset(&curr_time, '\0', sizeof(struct timeval));

if (argc > 1) {
  outputFilename = argv[1]; // Output the results
}

numbytes = 0;

// Loop indefinitely for incoming transmission
while(1) {
  // Retrieve length of packet at MAC layer
```

```
    if ((numbytes = recvfrom(sockfd, buffer, MAXBUFFSIZE ,
       0, (struct sockaddr*) &rcv_addr, &addr_len)) == -1)
       {
      perror("recvfrom");
      exit(1);
    }

    // Obtaining socket's timestamp
    if (ioctl(sockfd, SIOCGSTAMP, &curr_time) == -1) {
      perror("Current Time");
      exit(-1);
    }

    // Open file stream to output packets' information
    if (outputFilename != NULL) {
      fp = fopen(outputFilename, "a");
      if (fp == NULL) {
        perror("Can't open output file");
        exit(1);
      }
    }

    // Receiving and Ethernet packet
    if (numbytes > sizeof(struct ethhdr)) {

      eth_hdr = (struct ethhdr*) buffer;

      if ((memcmp(eth_hdr->h_dest, sta_1, MAC_ADDR_LEN) *
          memcmp(eth_hdr->h_source, sta_1, MAC_ADDR_LEN) ==
              0) &&
        (memcmp(eth_hdr->h_dest, sta_2, MAC_ADDR_LEN) *
          memcmp(eth_hdr->h_source, sta_2, MAC_ADDR_LEN) ==
              0) &&
        htons(eth_hdr->h_proto) == ETH_P_IP) {
        count++;
        if (outputFilename != NULL) {
          fprintf(fp, "%5d\t%5d\t", count, numbytes);
        }
        else {
          printf("%5d\t%5d\t", count, numbytes);
        }

        seconds = (curr_time.tv_sec) % 86400;
        if (seconds < 0) seconds += 86400;
        if (outputFilename != NULL) {
          fprintf(fp, "%02d:%02d:%02d.%06u\t", seconds / 3600, (seconds % 3600) / 60,
            seconds % 60, (u_int32_t) curr_time.tv_usec);
        }
        else {
          printf("%02d:%02d:%02d.%06u\t", seconds / 3600, (seconds % 3600) / 60,
            seconds % 60, (u_int32_t) curr_time.tv_usec);
        }

        eth_hdr = (struct ethhdr*) buffer;
        //print_in_HEX(eth_hdr->h_dest, ':', 6); printf("\t"); // 6 B - Dest. Address
        //print_in_HEX(eth_hdr->h_source, ':', 6); printf("\t"); // 6 B - Source Address
        //printf("0x%.4x\t", htons(eth_hdr->h_proto));  // 2 B - Ethernet Protocol

        if (ntohs(eth_hdr->h_proto) == ETH_P_IP) {
          if (numbytes >= sizeof(struct ethhdr) + sizeof(struct iphdr)) {
            ip_hdr = (struct iphdr*) (buffer + sizeof(struct ethhdr));
            if (outputFilename != NULL) {
              fprintf(fp, "%s\t", inet_ntoa(ip_hdr->saddr));
              fprintf(fp, "%s\t", inet_ntoa(ip_hdr->daddr));
              fprintf(fp, "%d\t", ip_hdr->protocol);
            }
            else {
              printf("%s\t", inet_ntoa(ip_hdr->saddr));
              printf("%s\t", inet_ntoa(ip_hdr->daddr));
              printf("%d\t", ip_hdr->protocol);
            }
            // Packet uses UDP protocol
            if (ip_hdr->protocol == IPPROTO_UDP) {
              udp_hdr = (struct udphdr*) (buffer + sizeof(struct ethhdr) +
                ip_hdr->ihl*4);
              if (outputFilename != NULL) {
```

```c
              fprintf(fp, "%d\t", ntohs(udp_hdr->source));
              fprintf(fp, "%d\t", ntohs(udp_hdr->dest));
            }
            else {
              printf("%d\t", ntohs(udp_hdr->source));
              printf("%d\t", ntohs(udp_hdr->dest));
            }
          }
          // Packet uses TCP protocol
          else if (ip_hdr->protocol == IPPROTO_TCP) {
            tcp_hdr = (struct tcphdr*) (buffer + sizeof(struct ethhdr) +
              ip_hdr->ihl*4);
            if (outputFilename != NULL) {
              fprintf(fp, "%d\t", ntohs(tcp_hdr->source));
              fprintf(fp, "%d\t", ntohs(tcp_hdr->dest));
            }
            else {
              printf("%d\t", ntohs(tcp_hdr->source));
              printf("%d\t", ntohs(tcp_hdr->dest));
            }
          }
        }
        else {
          /* printf("!!! Not a full IP header !!!\n"); */
        }
      }
      else {
        /* printf("!!! Not an IP header !!!\n"); */
      }
      if (outputFilename != NULL) {
        fprintf(fp, "\n");
      }
      else {
        printf("\n");
      }
    }
    // Close output file buffer
    if (outputFilename != NULL) {
      fclose(fp);
    }
    }
    else {
      printf("!!! Packet size too small !!!");
    }
  }
  // Close all network sockets
  close(sockfd);
  if (DEBUG) printf("Socket closed\n");
  return 0;
}
```

B.2 Parser

```
/*
   **************************************************************

 *
 * Purpose: A parser that reads and processes the exported
    files from
 *        listener
 * Author:  D. Skordoulis
 * Version: 0.1
 * Date:    23-Feb-09
 *
 **************************************************************
    */

#include <stdio.h>     // standard I/O (e.g. printf)
#include <stdlib.h>    // standard library (eg. exit)
#include <string.h>    // manipulate strings
#include <errno.h>     // error reporting mechanism

#include <time.h>    // needed for the time function
#include <math.h>    // needed for mathematical functions

// Definitions
#define LINE_MAX_LENGTH 100
#define S_IP "10.0.0.2"
#define D_IP "10.0.0.1"
#define S_PORT "8998"
#define D_PORT "8997"

```

```
int display_file(char*);

/* Main Function */
int main(int argc, char *argv[]) {

  int i;
  char *file_name_input = NULL;
  char *file_name_output = NULL;
  char line[LINE_MAX_LENGTH];
  int lcount;
  double rate;
  double time_var;

  FILE *file_input;
  FILE *file_output;

  // Input arguments from the CLI that define the file and
     requested results
  if (argc < 5) {// 0: command, 1: -i, 2: fileA, 3: -r, 4:
     rate, 5: -t, 6: variation
    printf("usage: %s [-i input_file] [-r rate] [-v
       variation]\n", argv[0]);
    return 1;
  }

  for (i = 1; i < argc; i++) {
    if (argv[i][0] == '-')
    switch (argv[i][1]) {
      case 'i':
        file_name_input = argv[i+1];
        //printf("%s, with size %d\n",file_name_input,
           sizeof(file_name_input));
        file_name_output = malloc(strlen(file_name_input) +
            10);
        strncpy(file_name_output, file_name_input, strlen(
           file_name_input) - 4);
        strcat(file_name_output, "_stats.txt");
        //printf("%s, with size %d\n",file_name_output,
           sizeof(file_name_output));
        break;
      case 'r':
        rate = atof(argv[i+1]);
        break;
      case 'v':
        time_var = atof(argv[i+1]);
        break;
    }
  }

  // Open file streams for the input and output results
  file_input = fopen(file_name_input, "r");
  file_output = fopen(file_name_output, "w");

  if (file_input == NULL) {
    printf("Error opening %s: %s (%u)\n", file_name_input,
         strerror(errno), errno);
    return 1;
  }

  else if (file_output == NULL) {
    printf("Error creating %s for writing: %s (%u)\n",
       file_name_output,
         strerror(errno), errno);
    return 1;
  }

  char str_time[16], str_ip[2][16], str_port[2][6];
  int no, bytes, prot;

  while (fgets(line, LINE_MAX_LENGTH, file_input) != NULL)
     {
    sscanf(line, "%d%d%s%s%s%d%s%s", &no, &bytes, str_time,
        str_ip[0],
         str_ip[1], &prot, str_port[0], str_port[1]);
    if ((strcmp(str_ip[0], S_IP) == 0) && (strcmp(str_ip
       [1], D_IP) == 0) &&
      (strcmp(str_port[0], S_PORT) ==0) && (strcmp(str_port
         [1], D_PORT) == 0)
      ) {
      lcount++;
      fprintf(file_output, "%d\t%d\t%s\t%s\t%s\t%d\t%s\t%s\
         n", no, bytes, str_time,
          str_ip[0], str_ip[1], prot, str_port[0], str_port
             [1]);
    }
  }
```

```

fclose(file_input);
fclose(file_output);

file_input = fopen(file_name_output, "r");
char *str_time_[lcount];
int count = 0;
while (fgets(line, LINE_MAX_LENGTH, file_input) != NULL)
   {
  str_time_[count] = malloc(16);
  sscanf(line, "%*d%*d%s%*s%*s%*d%*s%*s", str_time_[count
     ]);
  //printf("%s\n", str_time_[count]);
  count++;
}
fclose(file_input);

struct timeval times[lcount];
struct tm tm;

int cc = 0;

for (; cc < lcount; cc++) {
  strptime(str_time_[cc], "%H:%M:%S", &tm);
  times[cc].tv_sec = (tm.tm_hour * 3600) + (tm.tm_min *
     60) + (tm.tm_sec);
  sscanf(str_time_[cc], "%*9s%d", &times[cc].tv_usec);
}

struct timeval times_diff[lcount-1];

for (cc = 1; cc < lcount; cc++) {
  /* Perform the carry for the later subtraction by
     updating y. */
  if (times[cc].tv_usec < times[cc-1].tv_usec) {
    int nsec = (times[cc-1].tv_usec - times[cc-1].tv_usec
       ) / 1000000 + 1;
    times[cc-1].tv_usec -= 1000000 * nsec;
    times[cc-1].tv_sec += nsec;
  }
  if (times[cc].tv_usec - times[cc-1].tv_usec > 1000000)
     {
    int nsec = (times[cc].tv_usec - times[cc-1].tv_usec)
```

```
       / 1000000;
    times[cc-1].tv_usec += 1000000 * nsec;
    times[cc-1].tv_sec -= nsec;
  }
  /* Compute the time remaining to wait. tv_usec is
     certainly positive. */
  times_diff[cc-1].tv_sec = times[cc].tv_sec - times[cc
     -1].tv_sec;
  times_diff[cc-1].tv_usec = times[cc].tv_usec - times[cc
     -1].tv_usec;
}

struct timeval time_diff_max, time_diff_min;
int equal_rate = 0;
int rate_10 = 0;
int betw_diff = 0;
double time_diff_total = 0;

for (cc = 0; cc < lcount-1; cc++) {
  if (cc == 0) {
    time_diff_max = times_diff[cc];
    time_diff_min = times_diff[cc];
  }
  else if ((times_diff[cc].tv_sec > time_diff_max.tv_sec)
      ||
       (times_diff[cc].tv_sec == time_diff_max.tv_sec &&
        times_diff[cc].tv_usec > time_diff_max.tv_usec))
           {
    time_diff_max = times_diff[cc];
  }
  else if ((times_diff[cc].tv_sec < time_diff_min.tv_sec)
      ||
       (times_diff[cc].tv_sec == time_diff_min.tv_sec &&
        times_diff[cc].tv_usec < time_diff_min.tv_usec))
           {
    time_diff_min = times_diff[cc];
  }
  if (times_diff[cc].tv_sec == 0) {
    if (fabs(times_diff[cc].tv_usec - (rate * 1000000)) <
        1E-9) {
      equal_rate++;
    }
    if (times_diff[cc].tv_usec - ((rate + time_var) *
         1000000) <= 1E-9 &&
        times_diff[cc].tv_usec - ((rate - time_var) *
           1000000) >= (-1)*1E-9 ) {
        betw_diff++;
      }
      if (times_diff[cc].tv_usec - 10 <= 0) {
        rate_10++;
      }
    }
    time_diff_total += ((times_diff[cc].tv_sec * 1000000) +
        times_diff[cc].tv_usec);
  }

  // Structured output results
  file_output = fopen(file_name_output, "w");

  fprintf(file_output, "Filename - Input: %s\n",
     file_name_input);
  fprintf(file_output, "No. of Packets: %d\n", lcount);
  fprintf(file_output, "Total time: %6.6f\n",
     time_diff_total / 1000000);
  fprintf(file_output, "Avg. time: %6.6f\n", (
     time_diff_total / 1000000) / (lcount -1));
  fprintf(file_output, "MAX IAT: %d.%06d\n", time_diff_max.
     tv_sec, time_diff_max.tv_usec);
  fprintf(file_output, "MIN IAT: %d.%06d\n", time_diff_min.
     tv_sec, time_diff_min.tv_usec);
  fprintf(file_output, "Equal Rate (%f): %3.2f%% (%d
     packets)\n", rate, ((equal_rate / (double)(lcount-1))
     *100), equal_rate);
  fprintf(file_output, "Within (+/-) Variation (%f): %3.2f
     %% (%d packets)\n", time_var, ((betw_diff / (double)(
     lcount-1))*100), betw_diff);
  fprintf(file_output, "Below or Equal 10 usec: %3.2f%% (%d
      packets)\n", ((rate_10 / (double)(lcount-1))*100),
     rate_10);

  fclose(file_output);

  display_file(file_name_output);

  return 0;
}
```

References

[1] D. Skordoulis, Q. Ni, H. Chen, A. 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, 2008.

[2] D. Skordoulis, Q. Ni, and C. Zarakovitis, "A Selective Delayed Channel Access (SDCA) for the High-Throughput IEEE 802.11n," in *Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE.* IEEE, 2009, pp. 1–6.

[3] D. Skordoulis, Q. Ni, G. Min, and K. Borg, "Adaptive Delayed Channel Access for IEEE 802.11n WLANs," in *Circuits and Systems for Communications, 2008. ICCSC 2008. 4th IEEE International Conference on.* IEEE, 2008, pp. 167–171.

[4] D. Skordoulis, Q. Ni, U. Ali, and M. Hadjinicolaou, "Analysis of Concatenation and Packing Mechanisms in IEEE 802.11n," in *Proceedings of the 6th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNET07)*, 2007.

[5] D. Akhmetov, S. Shtin, and A. Stephens, "MAC Detailed Design - Documentation," Intel Corporation, Tech. Rep., 2004.

[6] IEEE Computer Society LAN MAN Standards Committee, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," in *ANSI/IEEE Std 802.11, 1999 Edition (R2003)*, 2003.

[7] Broadband Radio Access Networks Project, "Radio Equipment and Systems (RES): High Performance Radio Local Area Network (HIPERLAN): Functional Specification," *European Telecommunications Standards Institute*, vol. 6, p. 921, 1995.

[8] V. Jones, R. DeVegt, and T. Jerry, "Interest for Higher Data Rates (HDR) extension to 802.11a," *IEEE 802.11n Working Group Document - IEEE 802.11-02/081r0*, 2002.

[9] J. Rosdahl, "Draft Project Authorization Request (PAR) for High Throughput Study Group," *IEEE 802.11n Working Group Document - IEEE 802.11-02/798r2*, 2003.

[10] IEEE Computer Society LAN MAN Standards Committee, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-speed Physical Layer in the 5 GHZ Band," in *IEEE Std 802.11a-1999 (Supplement to IEEE Std 802.11-1999)*, 1999.

[11] Y. Xiao and J. Rosdahl, "Throughput and Delay Limits of IEEE 802.11," *Communications Letters, IEEE*, vol. 6, no. 8, pp. 355–357, 2002.

[12] IEEE Computer Society LAN MAN Standards Committee, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements," in *ANSI/IEEE Std 802.11e, 2005 Edition*, 2005.

[13] ——, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amedment 5: Enhancements for Higher Throughput," in *ANSI/IEEE Std 802.11n, 2009 Edition*, 2009.

[14] M. Visser and M. El Zarki, "Voice and Data Transmission Over an 802.11 Wireless Network," in *Personal, Indoor and Mobile Radio Communications, 1995. PIMRC'95. Sixth IEEE International Symposium on*, vol. 2. IEEE, 1995, pp. 648–652.

[15] Y. Tian, K. Xu, and N. Ansari, "TCP in Wireless Environments: Problems and Solutions," *Communications Magazine, IEEE*, vol. 43, no. 3, pp. S27–S32, 2005.

[16] G. Xylomenos, G. Polyzos, P. Mahonen, and M. Saaranen, "TCP Performance Issues over Wireless Links," *Communications Magazine, IEEE*, vol. 39, no. 4, pp. 52–58, 2001.

[17] G. Xylomenos and G. Polyzos, "TCP and UDP Performance over a Wireless LAN," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 1999, pp. 439–446.

[18] D. Leith and P. Clifford, "TCP Fairness in 802.11e WLANs," in *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, vol. 1. IEEE, 2005, pp. 649–654.

[19] C. Liu and A. Stephens, "Delayed Channel Access for IEEE 802.11e Based WLAN," in *Communications, 2006. ICC'06. IEEE International Conference on*, vol. 10. IEEE, 2006, pp. 4811–4817.

[20] C. Villamizar and C. Song, "High Performance TCP in ANSNET," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 5, pp. 45–60, 1994.

[21] L. Green, K. Balmy, and M. Emmelmann, "Theoretical Throughput Limits (Substantive Standard Draft Text)," *802.11 TGt Wireless Performance Prediction Task Group doc. 06/928r2*, 2006.

[22] J. Jun, P. Peddabachagari, and M. Sichitiu, "Theoretical Maximum Throughput of IEEE 802.11 and its Applications," in *Network Computing and Applications, 2003. NCA 2003. Second IEEE International Symposium on*. IEEE, 2003, pp. 249–256.

[23] Y. Xiao, "Packing Mechanisms for the IEEE 802.11n Wireless LANs," in *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, vol. 5. IEEE, 2004, pp. 3275–3279.

[24] Y. Kim, S. Choi, K. Jang, and H. Hwang, "Throughput Enhancement of IEEE 802.11 WLAN via Frame Aggregation," in *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, vol. 4. IEEE, 2004, pp. 3030–3034.

[25] R. Choudhury, Y. Chen, and S. Emeott, "Performance Analysis of Data Aggregation Techniques for Wireless LAN," in *Proceedings of IEEE Globecom*. Citeseer, 2006, pp. 1958–1962.

[26] Y. Lin and V. Wong, "Frame Aggregation and Optimal Frame Size Adaptation for IEEE 802.11n WLANs," in *Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE*. IEEE, 2006, pp. 1–6.

[27] T. Selvam and S. Srikanth, "A Frame Aggregation Scheduler for IEEE 802.11n," in *Communications (NCC), 2010 National Conference on*. IEEE, 2010, pp. 1–5.

[28] B. Ginzburg and A. Kesselman, "Performance Analysis of A-MPDU and A-MSDU Aggregation in IEEE 802.11n," in *Sarnoff Symposium, 2007 IEEE*. IEEE, 2007, pp. 1–5.

[29] J. Hu, G. Min, and M. Woodward, "Analysis and Comparison of Burst Transmission Schemes in Unsaturated 802.11e WLANs," in *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*. IEEE, 2007, pp. 5133–5137.

[30] ——, "Modeling of IEEE 802.11e Contention Free Bursting Scheme with Heterogeneous Stations," in *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007. MASCOTS'07. 15th International Symposium on*. IEEE, 2007, pp. 88–94.

[31] N. Taher, Y. Ghamri-Doudane, and B. El Hassan, "Transmission Time Analysis and Modeling in 802.11e Contention Free Burst Mode," in *Global Information Infrastructure Symposium, 2007. GIIS 2007. First International*. IEEE, 2007, pp. 191–194.

[32] T. Suzuki, A. Noguchi, and S. Tasaka, "Effect of TXOP-Bursting and Transmission Error on Application-Level and User-Level QoS in Audio-Video Transmission with IEEE 802.11e EDCA," in *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on.* IEEE, 2006, pp. 1–7.

[33] S. Selvakennedy, "The Impact of Transmit Buffer on EDCA with Frame-Bursting Option for Wireless Networks," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on.* IEEE, 2004, pp. 696–697.

[34] T. Li, Q. Ni, D. Malone, D. Leith, Y. Xiao, and T. Turletti, "Aggregation with Fragment Retransmission for Very High-Speed WLANs," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 2, pp. 591–604, 2009.

[35] Q. Ni, T. Li, T. Turletti, and Y. Xiao, "AFR Partial MAC Proposal for IEEE 802.11n," *IEEE 802.11n Working Group Document - IEEE 802.11-04/0949r00*, 2004.

[36] S. Rashwand and J. Misic, "IEEE 802.11e EDCA Under Bursty Traffic-How Much TXOP Can Improve Performance," *Vehicular Technology, IEEE Transactions on*, vol. 60, no. 3, pp. 1099 – 1115, 2011.

[37] F. Peng, H. Alnuweiri, and V. Leung, "Analysis of Burst Transmission in IEEE 802.11e Wireless LANs," in *Communications, 2006. ICC'06. IEEE International Conference on*, vol. 2. IEEE, 2006, pp. 535–539.

[38] H. Zimmermann, "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection," *Communications, IEEE Transactions on*, vol. 28, no. 4, pp. 425–432, 1980.

[39] I. Tinnirello and S. Choi, "Efficiency Analysis of Burst Transmissions with Block ACK in Contention-Based 802.11e WLANs," in *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, vol. 5. Ieee, 2005, pp. 3455–3460.

[40] H. Lee, I. Tinnirello, J. Yu, and S. Choi, "Throughput and Delay Analysis of IEEE 802.11e Block ACK with Channel Errors," in *Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference on.* IEEE, 2007, pp. 1–7.

[41] A. Balachandran, G. Voelker, P. Bahl, and P. Rangan, "Characterizing User Behavior and Network Performance in a Public Wireless LAN," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1. ACM, 2002, pp. 195–205.

[42] M. Balazinska and P. Castro, "Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network," in *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services.* ACM, 2003, pp. 303–316.

[43] A. Jardosh, K. Ramachandran, K. Almeroth, and E. Belding-Royer, "Understanding Link-Layer Behavior in Highly Congested IEEE 802.11b Wireless Networks," in *Proceedings of the 2005 ACM SIGCOMM.* ACM, 2005, pp. 11–16.

[44] T. Nakajima, Y. Utsunomiya, Y. Nishibayashi, T. Tandai, T. Adachi, and M. Takagi, "Compressed Block Ack, an Efficient Selective Repeat Mechanism for IEEE 802.11n," in *Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on*, vol. 3. IEEE, 2005, pp. 1479–1483.

[45] V. Frost and B. Melamed, "Traffic Modeling for Telecommunications Networks," *Communications Magazine, IEEE*, vol. 32, no. 3, pp. 70–81, 1994.

[46] P. Kenis and V. Schneider, "Policy Networks and Policy Analysis: Scrutinizing a New Analytical Toolbox," *Policy Networks: Empirical Evidence and Theoretical Considerations*, pp. 25–59, 1991.

[47] R. Paul and D. Balmer, *Simulation Modelling.* Chartwell-Bratt, 1993.

[48] D. Vose, *Quantitative Risk Analysis: Guide to Monte Carlo Simulation Modelling.* John Wiley and Sons, 1996.

[49] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heidemann, P. Huang, *et al.*, “Improving Simulation for Network Research,” 1999.

[50] X. Xian, W. Shi, and H. Huang, “Comparison of OMNET++ and Other Simulator for WSN Simulation,” in *Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference on.* IEEE, 2008, pp. 1439–1443.

[51] G. Lucio, M. Paredes-Farrera, E. Jammeh, M. Fleury, and M. Reed, “OPNET Modeler and Ns-2: Comparing the Accuracy of Network Simulators for Packet-Level Analysis Using a Network Testbed,” *WSEAS Transactions on Computers*, vol. 2, no. 3, pp. 700–707, 2003.

[52] OPNET Technologies Inc. OPNET Modeler. [Online]. Available: http://www.opnet.com/solutions/network_rd/modeler.html

[53] X. Chang, “Network Simulations with OPNET,” in *Simulation Conference Proceedings, 1999 Winter*, vol. 1. IEEE, 1999, pp. 307–314.

[54] R. Baldwin, N. Davis IV, and S. Midkiff, “Implementation of an IEEE 802.11 Wireless LAN model using OPNET,” in *Proceedings of OPNETWORK*, vol. 98, 1998.

[55] W. Xi, T. Whitley, A. Munro, and M. Barton, “Modeling and Simulation of MAC for QoS in IEEE 802.11e Using OPNET Modeler,” *Networks &J Protocols Group, CCR, Department of Electrical & Electronic Engineering, University of Bristol*, 2006.

[56] C. Li, T. Tsuei, and H. Chao, “Evaluation of Contention-Based EDCA for IEEE 802.11e Wireless LAN,” *Journal of Internet Technology*, vol. 5, no. 4, pp. 429–434, 2004.

[57] J. Song and L. Trajkovic, “Enhancements and Performance Evaluation of Wireless Local Area Networks,” 2003. [Online]. Available: http://www.ensc.sfu.ca/~ljilja/papers

[58] J. Ransbottom and N. Davis IV, “Packet Aggregation through a Wireless LAN using OPNET,” 2003.

[59] S. Mujtaba *et al.*, “TGnSync Proposal MAC Results,” *IEEE 802.11n Working Group Document - IEEE 802.11-04/0892r0*, 2004.

[60] A. Stephens, Y. Morioka, T. Adachi, D. Akhmetov, and S. Shtin, “TGn Joint Proposal MAC Results,” *IEEE 802.11n Working Group Document - IEEE 802.11-05/1266r1*, 2006.

[61] A. Stephens, D. Akhmetov, and S. Shtin, “802.11 TGn Intel Simulation Model,” *IEEE 802.11n Working Group Document - IEEE 802.11-08/0740r0*, 2008.

[62] D. Akhmetov, S. Shtin, and A. Stephens, “MPS High-Level Design - Documentation,” Intel Corporation, Tech. Rep., 2004.

[63] OPNET Modeler, *Product Documentation - Release 10.5*, OPNET Technologies Inc, 2004.

[64] D. Akhmetov and S. Shtin, “A Method for 802.11e QoS Functionality Implementation,” Intel Corporation, Tech. Rep., 2006.

[65] M. Gaudel, “Testing can be formal, too,” *TAPSOFT’95: Theory and Practice of Software Development*, pp. 82–96, 1995.

[66] G. Bernot, M. Gaudel, and B. Marre, “Software Testing Based on Formal Specifications: a Theory and a Tool,” *Software Engineering Journal, IET*, vol. 6, no. 6, pp. 387–405, 1991.

[67] D. Akhmetov, S. Shtin, and A. Stephens, “PHY Detailed Design - Documentation,” Intel Corporation, Tech. Rep., 2004.

[68] A. Stephens *et al.*, “IEEE P802.11 Wireless LANs: Usage Models,” *IEEE 802.11n Working Group Document - IEEE 802.11-03/802r23*, May 11 2004.

[69] C. Wang, E. Au, R. Murch, W. Mow, R. Cheng, and V. Lau, “On the Performance of the MIMO Zero-Forcing Receiver in the Presence of Channel Estimation Error,” *Wireless Communications, IEEE Transactions on*, vol. 6, no. 3, pp. 805–810, 2007.

[70] IEEE Computer Society LAN MAN Standards Committee, "IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges," in *ANSI/IEEE Std 802.1D-1998 Edition (R2004)*, 2004.

[71] L. Li, M. Pal, and Y. Yang, "Proportional Fairness in Multi-rate Wireless LANs," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE.* IEEE, 2008, pp. 1004–1012.

[72] G. Tan and J. Guttag, "Time-based Fairness Improves Performance in Multi-rate WLANs," in *Proceedings of the annual conference on USENIX Annual Technical Conference.* USENIX Association, 2004, pp. 23–23.

[73] G. Cantieni, Q. Ni, C. Barakat, and T. Turletti, "Performance Analysis under Finite Load and Improvements for Multirate 802.11," *Computer Communications*, vol. 28, no. 10, pp. 1095–1109, 2005.

[74] N. Bailey, "On Queueing Processes with Bulk Service," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 80–87, 1954.

[75] F. Downton, "Waiting Time in Bulk Service Queues," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 256–261, 1955.

[76] M. Wilson, "A Historical View of Network Traffic Models," Washington University in St. Louis, Tech. Rep., 2008.

[77] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *Networking, IEEE/ACM Transactions on*, vol. 2, no. 1, pp. 1–15, 1994.

[78] P. Olivier and N. Benameur, "Flow Level IP Traffic Characterization," *Networks, IEEE*, vol. 10, p. 11, 2001.

[79] W. Willinger, V. Paxson, and M. Taqqu, "Self-similarity and Heavy Tails: Structural Modeling of Network Traffic," *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, vol. 23, pp. 27–53, 1998.

[80] B. Mandelbrot, "Self-similar Error Clusters in Communication Systems and the Concept of Conditional Stationarity," *Communication Technology, IEEE Transactions on*, vol. 13, no. 1, pp. 71–90, 1965.

[81] R. Adelson, "Compound Poisson Distributions," *Operational Research*, vol. 17, no. 1, pp. 73–75, 1966.

[82] H. Heffes and D. Lucantoni, "A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance," *Selected Areas in Communications, IEEE Journal on*, vol. 4, no. 6, pp. 856–868, 1986.

[83] G. Min, J. Hu, and M. Woodward, "Performance Modelling and Analysis of the TXOP Scheme in Wireless Multimedia Networks with Heterogeneous Stations," *Wireless Communications, IEEE Transactions on*, vol. 10, no. 12, pp. 4130–4139, 2011.

[84] R. Jain and S. Routhier, "Packet Trains – Measurements and a New Model for Computer Network Traffic," *Selected Areas in Communications, IEEE Journal on*, vol. 4, no. 6, pp. 986–995, 1986.

[85] A. Erramilli, M. Roughan, D. Veitch, and W. Willinger, "Self-Similar Traffic and Network Dynamics," *Proceedings of the IEEE*, vol. 90, no. 5, pp. 800–819, 2002.

[86] G. Bolch, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications.* Wiley-Blackwell, 2006.

[87] M. Chaudhry and U. Gupta, "Modelling and Analysis of M/G[a, b]/1/N queue - A simple alternative approach," *Queueing Systems*, vol. 31, no. 1, pp. 95–100, 1999.

[88] M. Chaudhry, B. Madill, and G. Briere, "Computational Analysis of Steady-State Probabilities of M/G[a, b]/1 and Related Nonbulk Queues," *Queueing Systems*, vol. 2, no. 2, pp. 93–114, 1987.

[89] M. Chaudhry and J. Templeton, *A First Course in Bulk Queues.* John Wiley&Sons, 1983.

[90] K. Hirasawa, "Numerical Solutions of Bulk Queues via Imbedded Markov Chain," *Elec. Eng. Jpn*, vol. 91, pp. 127–136, 1971.

[91] U. Bhat, *An Introduction to Queueing Theory: Modeling and Analysis in Applications.* Birkhauser, 2008.

[92] H. Gold and P. Tran-Gia, "Performance Analysis of a Batch Service Queue Arising Out of Manufacturing System Modelling," *Queueing Systems*, vol. 14, no. 3, pp. 413–426, 1993.

[93] P. Kuehn, "Approximate Analysis of General Queuing Networks by Decomposition," *Communications, IEEE Transactions on*, vol. 27, no. 1, pp. 113–126, 1979.

[94] P. Tran-Gia and T. Raith, "Performance Analysis of Finite Capacity Polling Systems with Nonexhaustive Service," *Performance Evaluation*, vol. 9, no. 1, pp. 1–16, 1988.

[95] S. Kuppa and G. Dattatreya, "Modeling and Analysis of Frame Aggregation in Unsaturated WLANs with Finite Buffer Stations," in *Communications, 2006. ICC'06. IEEE International Conference on*, vol. 3. IEEE, 2006, pp. 967–972.

[96] I. Adan and J. Resing, *Queueing Theory - Lecture Notes*, 2001. [Online]. Available: http://www.win.tue.nl/~iadan/queueing.pdf

[97] S. Ross, *Introduction to Probability Models.* Academic Press, 2009.

[98] R. Marie, "Calculating Equilibrium Probabilities for $\lambda(n)/C_k/1/N$ Queues," in *ACM Sigmetrics Performance Evaluation Review*, vol. 9, no. 2. ACM, 1980, pp. 117–125.

[99] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, and L. Stibor, "IEEE 802.11e Wireless LAN for Quality of Service," in *Proc. European Wireless*, vol. 2, 2002, pp. 32–39.

[100] S. Mangold, S. Choi, G. Hiertz, O. Klein, and B. Walke, "Analysis of IEEE 802.11e for QoS support in wireless LANs," *Wireless Communications, IEEE*, vol. 10, no. 6, pp. 40–50, 2003.

[101] A. Grilo and M. Nunes, "Performance Evaluation of IEEE 802.11e," in *Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on*, vol. 1. IEEE, 2002, pp. 511–517.

[102] S. Choi, J. Del Prado, S. Mangold, *et al.*, "IEEE 802.11e Contention-Based Channel Access (EDCF) Performance Evaluation," in *Communications, 2003. ICC'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1151–1156.

[103] Q. Ni, "Performance Analysis and Enhancements for IEEE 802.11e Wireless Networks," *Network, IEEE*, vol. 19, no. 4, pp. 21–27, 2005.

[104] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 3, pp. 535–547, 2000.

[105] E. Ziouva and T. Antonakopoulos, "CSMA/CA Performance under High Traffic Conditions: Throughput and Delay Analysis," *Computer Communications*, vol. 25, no. 3, pp. 313–321, 2002.

[106] Y. Xiao, "Performance Analysis of IEEE 802.11e EDCF Under Saturation Condition," in *Communications, 2004 IEEE International Conference on*, vol. 1. IEEE, 2004, pp. 170–174.

[107] ——, "Performance Analysis of Priority Schemes for IEEE 802.11 and IEEE 802.11e Wireless LANs," *Wireless Communications, IEEE Transactions on*, vol. 4, no. 4, pp. 1506–1515, 2005.

[108] J. Hui and M. Devetsikiotis, "A Unified Model for the Performance Analysis of IEEE 802.11e EDCA," *Communications, IEEE Transactions on*, vol. 53, no. 9, pp. 1498–1510, 2005.

[109] A. Banchs and L. Vollero, "A Delay Model for IEEE 802.11e EDCA," *Communications Letters, IEEE*, vol. 9, no. 6, pp. 508–510, 2005.

[110] Z. Kong, D. Tsang, B. Bensaou, and D. Gao, "Performance Analysis of IEEE 802.11e Contention-Based Channel Access," *Selected Areas in Communications, IEEE Journal on*, vol. 22, no. 10, pp. 2095–2106, 2004.

[111] J. Robinson and T. Randhawa, "Saturation Throughput Analysis of IEEE 802.11e Enhanced Distributed Coordination Function," *Selected Areas in Communications, IEEE Journal on*, vol. 22, no. 5, pp. 917–928, 2004.

[112] P. Engelstad and O. Østerbø, "Non-Saturation and Saturation Analysis of IEEE 802.11e EDCA with Starvation Prediction," in *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems.* ACM, 2005, pp. 224–233.

[113] J. Hu, G. Min, M. Woodward, and W. Jia, "A Comprehensive Analytical Model for IEEE 802.11e QoS Differentiation Schemes under Unsaturated Traffic Loads," in *Communications, 2008. ICC'08. IEEE International Conference on.* IEEE, 2008, pp. 241–245.

[114] X. Chen, H. Zhai, X. Tian, and Y. Fang, "Supporting QoS in IEEE 802.11e wireless LANs," *Wireless Communications, IEEE Transactions on*, vol. 5, no. 8, pp. 2217–2227, 2006.

[115] The MathWorks, "MATLAB R2010a," Natick, Massachusetts, 2010.

[116] S. Jeong, H. Owen, J. Copeland, and J. Sokol, "QoS Support for UDP/TCP Based Networks," *Computer communications*, vol. 24, no. 1, pp. 64–77, 2001.

[117] S. Lam and G. Xie, "Burst Scheduling: Architecture and Algorithm for Switching Packet Video," in *INFOCOM'95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People. Proceedings. IEEE.* IEEE, 1995, pp. 940–950.

[118] M. Becchi, "From Poisson Processes to Self-Similarity: a Survey of Network Traffic Models," Washington University in St. Louis, Tech. Rep., 2008.

[119] M. Marcon, M. Dischinger, K. Gummadi, and A. Vahdat, “The Local and Global Effects of Traffic Shaping in the Internet,” in *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on.* IEEE, 2011, pp. 1–10.

[120] R. Bruno, M. Conti, and E. Gregori, “Throughput Analysis and Measurements in IEEE 802.11 WLANs with TCP and UDP Traffic Flows,” *Mobile Computing, IEEE Transactions on*, vol. 7, no. 2, pp. 171–186, 2008.

[121] S. Floyd, T. Henderson, and A. Gurtov, “RFC 2582: The NewReno Modification to TCPs Fast Recovery Algorithm,” Network Working Group, Tech. Rep., 1999.

[122] IEEE 802.11T Task Group, “Recommended Practice for the Evaluation of 802.11 Wireless Performance,” *IEEE Unapproved Draft Std P802.11.2 / D1.01*, 2008.

[123] F. Li, M. Li, R. Lu, H. Wu, M. Claypool, and R. Kinicki, “Measuring Queue Capacities of IEEE 802.11 Wireless Access Points,” in *Broadband Communications, Networks and Systems, 2007. BROADNETS 2007. Fourth International Conference on.* IEEE, 2007, pp. 846–853.

[124] H. Park, J. Lee, and B. Kim, “TCP Performance Issues in LTE Networks,” in *ICT Convergence (ICTC), 2011 International Conference on.* IEEE, 2011, pp. 493–496.

[125] M. Claypool, R. Kinicki, M. Li, J. Nichols, and H. Wu, “Inferring Queue Sizes in Access Networks by Active Measurement,” *Passive and Active Network Measurement*, pp. 227–236, 2004.

[126] Description of Windows 2000 and Windows Server 2003 TCP Features. [Online]. Available: http://support.microsoft.com/kb/224829

[127] T. Lakshman and U. Madhow, “The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss,” *Networking, IEEE/ACM Transactions on*, vol. 5, no. 3, pp. 336–350, 1997.

[128] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, and P. Sinha, “Understanding TCP Fairness over Wireless LAN,” in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2. IEEE, 2003, pp. 863–872.

[129] J. Postel, “RFC 793: Transmission Control Protocol,” Defense Advanced Research Projects Agency, Tech. Rep., 1981.

[130] W. Stevens, *TCP/IP Illustrated: the Protocols.* Addison-Wesley Professional, 1994, vol. 1.

[131] J. Postel, “RFC 791: Internet Protocol,” Defense Advanced Research Projects Agency, Tech. Rep., 1981.

[132] V. Cerf and R. Kahn, “A Protocol for Packet Network Intercommunication,” *Communications, IEEE Transactions on*, vol. 22, no. 5, pp. 637–648, 1974.

[133] L. Brakmo and L. Peterson, “TCP Vegas: End to End Congestion Avoidance on a Global Internet,” *Selected Areas in Communications, IEEE Journal on*, vol. 13, no. 8, pp. 1465–1480, 1995.

[134] V. Paxson and M. Allman, “RFC 2988: Computing TCPs Retransmission Timer,” Network Working Group, Tech. Rep., 2000.

[135] M. Allman, S. Floyd, and C. Partridge, “RFC 3390: Increasing TCPs Initial Window,” Network Working Group, Tech. Rep., 2002.

[136] M. Allman, V. Paxson, and W. Stevens, “RFC 2581: TCP Congestion Control,” Network Working Group, Tech. Rep., 1999.

[137] W. Stevens, “RFC 2001: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms,” Network Working Group, Tech. Rep., 1997.

[138] L. Brakmo, S. O’malley, and L. Peterson, “TCP Vegas: New Techniques for Congestion Detection and Avoidance,” in *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4. ACM, 1994, pp. 24–35.

[139] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4. IEEE, 2004, pp. 2490–2501.

[140] J. Hoe, "Start-Up Dynamics of TCP's Congestion Control and Avoidance Schemes," Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1995.

[141] ——, "Improving the Start-Up Behavior of a Congestion Control Scheme for TCP," in *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 4. ACM, 1996, pp. 270–280.

[142] C. Caini and R. Firrincieli, "TCP Hybla: a TCP Enhancement for Heterogeneous Networks," *International Journal of Satellite Communications and Networking*, vol. 22, no. 5, pp. 547–566, 2004.

[143] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[144] K. Song, Q. Zhang, and M. Sridharan, "Compound TCP: A Scalable and TCP-Friendly Congestion Control for High-Speed Networks," *Proceedings of PFLDnet 2006*, 2006.

[145] M. Corson, J. Macker, and G. Cirincione, "Internet-Based Mobile Ad Hoc Networking," *Internet Computing, IEEE*, vol. 3, no. 4, pp. 63–70, 1999.

[146] R. Schmitz, M. Torrent-Moreno, H. Hartenstein, and W. Effelsberg, "The Impact of Wireless Radio Fluctuations on Ad Hoc Network Performance," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on.* IEEE, 2004, pp. 594–601.

[147] M. Allman, V. Paxson, and E. Blanton, "RFC 5681: TCP Congestion Control," Internet Engineering Task Force, Tech. Rep., 2009.

[148] T. Henderson, D. Kotz, and I. Abyzov, "The Changing Usage of a Mature Campus-wide Wireless Network," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking.* ACM, 2004, pp. 187–201.

[149] D. Lee, B. Carpenter, and N. Brownlee, "Observations of UDP to TCP Ratio and Port Numbers," in *Internet Monitoring and Protection (ICIMP), 2010 Fifth International Conference on.* IEEE, 2010, pp. 99–104.

[150] J. Rajahalme, A. Conta, B. Carpenter, and S. Deering, "RFC 3697: IPv6 Flow Label Specification," Network Working Group, Tech. Rep., 2004.

[151] M. Kim, Y. Won, and J. Hong, "Characteristic Analysis of Internet Traffic from the Perspective of Flows," *Computer Communications*, vol. 29, no. 10, pp. 1639–1652, 2006.

[152] D. Skordoulis, "Simulation Analysis of the IEEE 802.11n Next-Generation Wireless LAN," Master's thesis, Brunel University, ECE, 2005.

[153] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices," in *Proceedings of the 2008 ACM CoNEXT Conference.* ACM, 2008, p. 11.

[154] C. Lee, D. Lee, and S. Moon, "Unmasking the Growing UDP Traffic in a Campus Network," in *Passive and Active Measurement.* Springer, 2012, pp. 1–10.

[155] T. Karagiannis, A. Broido, M. Faloutsos, *et al.*, "Transport Layer Identification of P2P Traffic," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement.* ACM, 2004, pp. 121–134.

[156] A. Moore and K. Papagiannaki, "Toward the Accurate Identification of Network Applications," *Passive and Active Network Measurement*, pp. 41–54, 2005.

[157] Ellacoya Networks. Ellacoya IP Service Control System. [Online]. Available: http://www.ellacoya.com

[158] Packeteer. Packetshaper. [Online]. Available: http://www.packeteer.com

[159] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark," in *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4. ACM, 2005, pp. 229–240.

[160] S. Sen, O. Spatscheck, and D. Wang, "Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 512–521.

[161] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "ACAS: Automated Construction of Application Signatures," in *Proceedings of the 2005 ACM SIGCOMM Workshop on Mining Network Data*. ACM, 2005, pp. 197–202.

[162] T. Choi, C. Kim, S. Yoon, J. Park, B. Lee, H. Kim, H. Chung, and T. Jeong, "Content-Aware Internet Application Traffic Measurement and Analysis," in *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, vol. 1. IEEE, 2004, pp. 511–524.

[163] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, "Network Monitoring Using Traffic Dispersion Graphs," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2007, pp. 315–320.

[164] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow Clustering Using Machine Learning Techniques," *Passive and Active Network Measurement*, pp. 205–214, 2004.

[165] A. Moore and D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1. ACM, 2005, pp. 50–60.

[166] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-Service Mapping for QoS: a Statistical Signature-Based Approach to IP Traffic Classification," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2004, pp. 135–148.

[167] J. Erman, M. Arlitt, and A. Mahanti, “Traffic Classification Using Clustering Algorithms,” in *Proceedings of the 2006 SIGCOMM workshop on Mining Network Data.* ACM, 2006, pp. 281–286.

[168] T. Auld, A. Moore, and S. Gull, “Bayesian Neural Networks for Internet Traffic Classification,” *Neural Networks, IEEE Transactions on*, vol. 18, no. 1, pp. 223–239, 2007.

[169] N. Williams, S. Zander, and G. Armitage, “A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.

[170] Canonical Ltd. Ubuntu 8.10 (Intrepid Ibex). [Online]. Available: http://old-releases.ubuntu.com/releases/8.10/

[171] Netgear. WG511T–108 Mbps Wireless PC Card. [Online]. Available: http://support.netgear.com/product/WG511T

[172] USRobotics. USR5410–802.11g Wireless Turbo PC Card. [Online]. Available: http://www.usr.com/support/product-template.asp?prod=5410

[173] S. Avallone, S. Guadagno, D. Emma, A. Pescapè, and G. Ventre, “D-ITG Distributed Internet Traffic Generator,” in *Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings. First International Conference on the.* IEEE, 2004, pp. 316–317.

[174] A. Botta, A. Dainotti, and A. Pescapè, “Multi-protocol and multi-platform traffic generation and measurement,” *INFOCOM 2007 DEMO Session*, 2007.

[175] S. Avallone, A. Botta, D. Emma, S. Guadagno, and A. Pescapè, “D-ITG V. 2.4 Manual,” *University of Napoli Federio II, Tech. Rep*, 2004.

[176] F. Fuentes and D. Kar, “Ethereal vs. Tcpdump: a Comparative Study on Packet Sniffing Tools for Educational Purpose,” *Journal of Computing Sciences in Colleges*, vol. 20, no. 4, pp. 169–176, 2005.

[177] carl9170 Driver. [Online]. Available: http://wireless.kernel.org/en/users/Drivers/carl9170

[178] ar9170 Driver. [Online]. Available: http://wireless.kernel.org/en/users/Drivers/ar9170

[179] Broadcom BRCMSMAC(PCIe) and BRCMFMAC(SDIO) Drivers. [Online]. Available: http://wireless.kernel.org/en/users/Drivers/brcm80211

[180] K. Bennett and C. Campbell, "Support Vector Machines: Hype or Hallelujah?" *ACM SIGKDD Explorations Newsletter*, vol. 2, no. 2, pp. 1–13, 2000.

[181] TGnSync, "TGnSync Proposal Technical Specification," 2004.

[182] S. Jeon and J. Lee, "Adaptive Frame Aggregation Scheme for Energy Efficiency in WLAN," in *Consumer Electronics (ICCE), 2011 IEEE International Conference on.* IEEE, 2011, pp. 463–464.