

2012

An improved medium access control protocol for real-time applications in WLANs and its firmware development

Sushil Dutt
Edith Cowan University

Follow this and additional works at: <https://ro.ecu.edu.au/theses>



Part of the [Digital Communications and Networking Commons](#)

Recommended Citation

Dutt, S. (2012). *An improved medium access control protocol for real-time applications in WLANs and its firmware development*. <https://ro.ecu.edu.au/theses/475>

This Thesis is posted at Research Online.
<https://ro.ecu.edu.au/theses/475>

Edith Cowan University

Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study.

The University does not authorize you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following:

- Copyright owners are entitled to take legal action against persons who infringe their copyright.
- A reproduction of material that is protected by copyright may be a copyright infringement. Where the reproduction of such material is done without attribution of authorship, with false attribution of authorship or the authorship is treated in a derogatory manner, this may be a breach of the author's moral rights contained in Part IX of the Copyright Act 1968 (Cth).
- Courts have the power to impose a wide range of civil and criminal sanctions for infringement of copyright, infringement of moral rights and other offences under the Copyright Act 1968 (Cth). Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

**An Improved Medium Access Control Protocol
for Real-Time Applications in WLANs and Its
Firmware Development**

A Thesis Submitted in Fulfillment of the Requirements for the Degree of
Master of Engineering Science

by

Sushil Dutt

School of Engineering

Edith Cowan University

Australia

Supervisors:

Prof. Daryoush Habibi, Edith Cowan University, Australia

Dr. Iftexhar Ahmad, Edith Cowan University, Australia

December, 2011

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

DECLARATION

I certify that this thesis does not, to the best of my knowledge and belief:

- I. incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any University;
- II. contain any material previously published or written by another person except where due reference is made in the text; or
- III. contain any defamatory material.

I also grant permission to the library at Edith Cowan University to make duplicate copies of this thesis as required.

ABSTRACT

The IEEE 802.11 Wireless Local Area Network (WLAN), commonly known as Wi-Fi, has emerged as a popular internet access technology and researchers are continuously working on improvement of the quality of service (QoS) in WLAN by proposing new and efficient schemes. Voice and video over Internet Protocol (VVoIP) applications are becoming very popular in Wi-Fi enabled portable/handheld devices because of recent technological advancements and lower service costs. Different from normal voice and video streaming, these applications demand symmetric throughput for the upstream and downstream. Existing Wi-Fi standards are optimised for generic internet applications and fail to provide symmetric throughput due to traffic bottleneck at access points. Performance analysis and benchmarking is an integral part of WLAN research, and in the majority of the cases, this is done through computer simulation using popular network simulators such as Network Simulator – 2 (NS-2) or OPNET. While computer simulation is an excellent approach for saving time and money, results generated from computer simulations do not always match practical observations. This is why, for proper assessment of the merits of a proposed system in WLAN, a trial on a practical hardware platform is highly recommended and is often a requirement. In this thesis work, with a view to address the above mentioned challenges for facilitating VoIP and VVoIP services over Wi-Fi, two key contributions are made: i) formulating a suitable medium access control (MAC) protocol to address symmetric traffic scenario and ii) firmware development of this newly devised MAC protocol for real WLAN hardware. The proposed solution shows significant improvements over existing standards by supporting higher number of stations with strict QoS criteria. The proposed hardware platform is available off-the-shelf in the market and is a cost effective

way of generating and evaluating performance results on a hardware system.

ACKNOWLEDGMENTS

I would like to give my sincere thanks to my supervisors Prof. Daryoush Habibi and Dr. Iftekhar Ahmad for their guidance and encouragement to bring this work into reality. I can never forget the time-to-time motivation given by Prof. Habibi that helped me to finish this work on time. I also give special thanks to all the postgraduate students of Centre For Communications and Engineering Research, ECU for keeping the Lab full of biscuits that helped in my late night work.

I am also grateful to the IT staff, Mr. Kourosh and Zhi Huan Yu, for their unrelenting support at all times. I also wish to acknowledge the support of the administrative staff, Muriel Vaughan and Kim Gardiner. All these people are very dedicated and proactive to meet the requirements of research students. I am also very grateful to Edith Cowan University for a Masters scholarship and School of Engineering for providing financial support for this research.

I wish to express my warmest gratitude to my friends, Marty Coles, Rajesh K Ramraj, Irfan Ullah and Thair Shakir, for reviewing various parts of this work and providing their valuable recommendations.

Most of all, a million thanks to my parents, and my wife, Preeti, for their infinite support and encouragement to undertake the challenge of this research.

PUBLICATIONS

- Sushil Dutt, Iftekhar Ahmad, and Daryoush Habibi. A Novel Optimised Scheduler to Provide QoS for Video IP Telephony over Wireless Networks. Published in 13th IEEE International Conference on High Performance Computing and Communications, HPCC 2011.
- Sushil Dutt, Iftekhar Ahmad, and Daryoush Habibi. A Low Cost Atheros System-on-Chip Based Hardware Testbed for 802.11 WLAN Research. Submitted in The IEEE Embedded Systems Letters.

List of Figures

1.1	A typical video IP telephony application scenario	19
2.1	The IEEE 802.11 system architecture	29
2.2	DCF timing cycle diagram	33
3.1	TI-MAC polling cycle timing diagram	47
4.1	Netgear WNDR3700 router	56
4.2	Netgear WNDR3700 hardware	57
4.3	Top-level architectural block diagram of AR7161	58
4.4	Top-Level architectural block diagram of AR9280	59
4.5	Ethernet ports connection with CPU	61
4.6	Image of hardware inside Netgear WNDR3700	62
4.7	Nokia CA-42 USB-to-Serial cable	63
4.8	Serial end pin configuration of Nokia CA-42 cable	63
4.9	Custom build cable for the serial interface with the WNDR3700	64
4.10	TFTP script source code	65
4.11	TFTP server start screen-shot	66
4.12	Test set-up and connections	66
4.13	Hardware boot-up screen-shot	67
4.14	TFTP commands for image transfer	68

4.15	Image transfer screen-shot	68
4.16	New image run screen-shot	69
4.17	Make menuconfig screen-shot	72
4.18	An example of net_device function pointer initialisation	73
4.19	Top-level software architecture of OpenWRT	74
4.20	Packet transmission functional path in OpenWRT	75
4.21	Packet reception functional path in OpenWRT	76
5.1	TI-MAC evaluation test-bed	79
5.2	Downstream and upstream data rates with TI-MAC at 2 Mbps .	83
5.3	Downstream and upstream data rates with 802.11e EDCA at 2 Mbps	83
5.4	Downstream and upstream data rates with TI-MAC at 9 Mbps .	84
5.5	Downstream and upstream data rates with 802.11e EDCA at 9 Mbps	84
5.6	IPDV (jitter) with TI-MAC at 2 and 9Mbps	85
5.7	IPDV (jitter) with 802.11e EDCA at 2 and 9Mbps	85

List of Tables

2.1	Network Performance Parameters	40
2.2	Guidance for QoS Classes	41
2.3	Image Resolution and Bandwidth Requirement	41
2.4	Voice Data Bandwidth Requirement	41
3.1	Parameters Used to Calculate Net Throughput at 2 Mbps	51
4.1	AR7161 Technical Specifications	57
4.2	Wires Color and Label	63
5.1	Test Scenario I Parameters	81
5.2	Test Scenario II Parameters	82
5.3	Common Parameters	82
5.4	Performance of TI-MAC v 802.11e EDCA-Test Scenario I	87
5.5	Performance of TI-MAC v 802.11e EDCA-Test Scenario II	88

Nomenclature

AP	Access Point
CLI	Command Line Interface
CSMA/CA	Carrier Sensing Multiple Access / Collision Avoidance
CTS	Clear To Send
CW	Contention Window
DCF	Distributed Coordination Function
DDR	Dual Data Rate
DIFS	Distributed (Coordination Function) Interframe Space
EDCA	Enhanced Distributed Channel Access
EDCAF	Enhanced Distributed Channel Access Function
GPL	GNU Public License
HC	Hybrid Coordinator
HCCA	Hybrid Coordination Channel Access
HCF	Hybrid Coordination Function

IDE	Integrated Development Environment
IP	Internet Protocol
IPDV	IP Packet Delay Variation
IPSR	IP Packet Symmetric Ratio
IPTD	IP Packet Transfer Delay
IPTV	Internet Protocol Television
ITU	International Telecommunication Union
LAN	Local Area Network
MAC	Medium Access Control
MIMO	Multiple Input Multiple Output
MPDU	MAC Protocol Data Unit
NS	Network Simulator
OFDM	Orthogonal Frequency Division Multiplexing
OS	Operating System
PCF	Point Coordination Function
PCI	Peripheral Component Interconnect
QoS	Quality of Service
RF	Radio Frequency
RTP	Real Time Protocol
RTS	Request To Send

SI	Service Interval
SoC	System on Chip
STA	Station
TFTP	Trivial File Transfer Protocol
TI-MAC	Traffic Intellegent MAC
TS	Traffic Stream
TXOP	Transmission Opportunity
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
VOIP	Voice Over Internet Protocol
VVoIP	Voice and Video Over Internet Protocol
WLAN	Wireless Local Area Network
WM	Wireless Medium
WTN	Wireless Token Network

Contents

1	Introduction	16
1.1	Wireless Network	17
1.2	Growth of Multimedia Traffic in WLAN	18
1.3	WLAN Challenges	20
1.4	Research Objectives	21
1.5	Research Contributions	23
1.5.1	Development of an Efficient MAC Protocol for VVoIP Ap- plications	23
1.5.2	Development of an Embedded Platform for MAC Protocol Validation	24
1.6	Thesis Structure	25
2	Background and Literature Review	27
2.1	Background Study of MAC Related Work	28
2.1.1	The IEEE 802.11 WLAN Standard	28
2.1.2	Quality of service (QoS) in WLAN	30
2.1.3	Distributed Coordinated Function (DCF) in IEEE 802.11 and QoS Performance	32
2.1.4	The IEEE 802.11e Standard for QoS	32

2.1.4.1	Hybrid Coordination Function (HCF) Channel Access (HCCA)	33
2.1.4.2	Enhanced Distributed Channel Access (EDCA)	34
2.1.5	Wireless Token Network (WTN)	35
2.1.6	Other Related MAC Works	36
2.2	Background Study of Hardware Platforms for WLAN Research	37
2.2.1	Wireless Open-Access Research Platform For Networks (WARPnet) – An FPGA Platform	37
2.2.2	CalRADIO1	38
2.2.3	Sagrad STLC4560 Radio Card with STM32 Eval Board	38
2.3	Introduction to Real-Time IP Traffic Attributes	40
2.4	Summary	42
3	Traffic Intelligent MAC (TI-MAC)	43
3.1	Motivation Behind TI-MAC	44
3.2	TI-MAC - Functional Specifications and Admission Control	45
3.3	Example of Skype Video Telephony	50
3.4	Summary	52
4	Netgear Hardware and TI-MAC Firmware Development	54
4.1	OpenWRT - Wireless Freedom	55
4.2	WNDR3700 Router by Netgear	56
4.2.1	AR7161 - Atheros Wireless Network Processor	57
4.2.2	AR9280 - Atheros Single-Chip 802.11a/b/g/n WLAN Chip	58
4.3	Hardware-PC Interface Setup	60
4.3.1	Custom Build Serial Interface Cable	62
4.3.2	Procedure to Program the Hardware	64
4.3.2.1	Set-up of the TFTP Server in the Host-PC	64

4.3.2.2	Transfer of the Binary Image to RAM	65
4.4	Software Stack Description	69
4.4.1	Software Download	69
4.4.2	Build OpenWRT for the WNDR3700	71
4.5	Software Architecture	73
4.5.1	Packet Transmission	74
4.5.2	Packet Reception	75
4.6	Firmware Development	76
4.7	Summary	76
5	Test Set-up and Characterisation	78
5.1	Test Scenario	80
5.1.1	Test Scenario I	81
5.1.2	Test Scenario II	81
5.2	Result Analysis	86
5.2.1	Test Scenario I	86
5.2.2	Test Scenario II	87
5.3	Summary	88
6	Conclusion and Future Works	89
6.1	Research Contributions	89
6.2	Test Scenarios and Results	90
6.3	Suggestions for Future Work	92

Chapter 1

Introduction

Wireless networks have shown an explosive growth in the last two decades and wireless local area network (WLAN) technology, commonly known as Wi-Fi, has been the pioneer of all wireless technologies. Deployment of WLANs was initially restricted by security requirements and lack of standards, but recent advancements in wireless technologies have overcome these limitations, which has lead to a rapid penetration of Wi-Fi technology into the consumer and enterprise markets. According to the Wi-Fi Alliance [1], about 200 million households use Wi-Fi networks and there are more than 750,000 Wi-Fi hotspots worldwide. Cellular operators are considering the proliferation of WLAN as the leading interface to meet the surging mobile data usages, and standards for an integration of cellular networks and WLAN are under development. Another communication sector that has been growing very rapidly is the voice over internet protocol (VoIP) technology. The VoIP technology has revolutionised the voice communication and offers a low cost service for both domestic and international telephone calls. Considering the growing popularity and penetration of Wi-Fi and VoIP technologies, numerous market research firms forecast VoIP over Wi-Fi as one of

the biggest communication industries in the near future. This chapter includes a brief introduction to these two technologies, challenges towards the integration of these two technologies, and motivation behind this research work.

1.1 Wireless Network

A wireless network exchanges information without wires between two or more networking capable computers or other devices through radio waves. There are a variety of instruments available which use different approaches to wirelessly exchange the network data using radio waves. Mobile networks, wireless wide area network (WWAN), wireless metropolitan area network (WMAN), wireless mesh network and wireless local area network (WLAN) are the commonly implemented topologies of wireless network. Mobile networks now-a-days support 3.5G telecommunications standards and routinely carry internet data, video, and mobile TV data in addition to the voice conversations [2]. WWAN covers large areas with the help of multiple access points making point-to-point communications. These networks are used to connect remote offices of a business or a public internet system. WMAN are typically used to connect a city or large campus with the help of multiple local area networks (LANs). WiMAX is an example of WMAN which is defined in the IEEE 802.16 standard. WLAN provides data connectivity to a computer or other capable device with an access point (AP) for the access of internet services and sharing of content. The IEEE 802.11 standard defines guidelines for both AP and the clients to make these devices plug-and-play in any WLAN system. WLANs are easy to set-up and maintain and enable both technical and non-technical users to connect with the internet world. We can imagine that the time is not far away when all corners of a city will be covered by Wi-Fi connectivity. WLAN technology for internet access is rapidly spreading, and due to an increasing popularity, it is being im-

plemented in most consumer electronics instruments e.g. mobile phones, TVs, laptops, gaming devices, printers, and CCTVs . As these devices support multimedia applications, the user expects the Wi-Fi to deliver this extra load of multimedia traffic along legacy text based data. The user also expects a seamless and fast connectivity in the Wi-Fi network as in a wired network, which leads to various technical challenges. Section 1.3 discusses the key challenges in WLAN and various proposed solutions.

1.2 Growth of Multimedia Traffic in WLAN

The internet traffic pattern has been significantly changed from text based to heavy multimedia traffic [3], and the number of users are also multiplying every year. It is evident that the growth of internet traffic is heavily dominated by an increase in video data. A research conducted by Cisco, Cisco Visual Forecast 2010-2015 [4], found that one million minutes of video per second would be transmitted by 2015. This report states that - *“Global advanced video traffic, including three-dimensional (3-D) and high-definition TV (HDTV), is projected to increase 14 times between 2010 and 2015. Business IP video conferencing is projected to grow sixfold over the forecast period, growing more than two times as fast as overall business IP traffic, at a compounded annual growth rate (CAGR) of 41 percent from 2010 to 2015”*. Applications like YouTube, Skype video telephony, internet protocol TV (IPTV), video surveillance and smartphones are few to be named as the key contributors towards IP multimedia traffic.

VoIP and VVoIP communication traffic is continuously growing even on low bandwidth networks like WLAN because of technological advancements in voice and video compression technology as well as low operational costs. Moreover, the IP communication is rapidly increasing both in home gateways and business environments. Skype is becoming very popular as a tool to facilitate IP voice



Figure 1.1: A typical video IP telephony application scenario

and video calls over smartphones, computers and TV [5]. Skype had an average of 124 million connected users per month in the second quarter of 2010 and Skype subscribers made 95 billion minutes of voice and video calls in the first half of 2010, approximately 40% of which was video [6]. A survey done by IDC [7] predicts that there will be close to half a billion personal IP communication subscribers with more than \$5 billion annual spending in 2012. A typical video IP telephony scenario where multiple clients are connected with an access point is depicted in Fig. 1.1. Considering the huge IP communication market volume and the fact that more than 50% of communications is now becoming wireless [7], new strategies and tools should be developed to meet the challenges and

maintain the high QoS in a WLAN environment.

1.3 WLAN Challenges

WLAN is a widely used technology in communication networks for numerous applications such as internet access, data transfer, content sharing servers, IP telephony and video surveillance. Wireless technology is rapidly becoming a preferred mode of data transfer due to its support for high mobility, less deployment time and lower cost. Access through wireless medium, however, offers a range of research challenges. For example, it can cause a high percentage of packet collisions, and high packet delay variation [8]. The performance of WLAN systems primarily depends upon the RF characteristics and the channel access scheme. In the last decade, WLAN technologies have made tremendous progress and the IEEE 802.11 committee has made commendable achievements while enhancing the RF performance and developing the IEEE 802.11n standard capable of providing dramatically increased throughput. The IEEE 802.11n amendment recommends many enhancements like Multiple-Input-Multiple-Output (MIMO) antenna configurations, orthogonal frequency division multiplexing (OFDM) modulation technique and wider channels that improve the throughput, RF coverage and overall reliability. The channel access scheme described as medium access control (MAC) layer in the IEEE 802.11 standard is a significant attribute of performance criteria, which defines the admission control policy, scheduling and the overheads. The MAC protocol decides the time of network data delivery to the radio module of WLAN card and hence, plays an important role in QoS provisioning of the wireless network where the channel bandwidth is limited.

The IEEE 802.11 WLAN standard for MAC [9] recommends several schemes for channel access in the Wi-Fi enabled devices. The IEEE 802.11 distributed coordination function (DCF) can not provide QoS to real-time traffic like voice

and video because of a fixed contention window (CW) for all traffic categories. Whereas, the IEEE 802.11e recommends guidelines to provide QoS to the multimedia traffic by providing it a higher priority than the text based data or the background data that is given a lower priority. The IEEE 802.11e defines methodologies to achieve QoS using enhanced distributed channel access (EDCA) function and hybrid coordinated channel access (HCCA) function. Among these, EDCA is mostly implemented in WLAN systems to provide prioritised channel access by dividing data into four access categories: voice (AC_VO), video (AC_VI), background (AC_BK) and best effort (AC_BE). However, in terms of QoS for symmetric data, experiments [10] show that the EDCA scheme can not provide enough bandwidth for downstream traffic, hence a true symmetric QoS can not be achieved. The efficiency of a MAC protocol depends upon the number of clients, and traffic pattern in the upstream and downstream directions. Whereas, the traditional technologies are becoming ineffective because of recent big shift in network traffic patterns. In this work, we have developed a polling-based MAC scheduler which works on traffic symmetry pattern, and controls the upstream and downstream transmission intervals. This new MAC considers the maximum end-to-end delay limit for the multimedia packet delivery and provides a polling cycle that can meet the pledged QoS for its clients. The other related theories that focus on the improvement of the MAC protocol have been discussed in Chapter 2.

1.4 Research Objectives

As discussed above, VVoIP applications and video streaming is going to significantly contribute to network IP traffic in the next decade. The bandwidth requirements for the upstream and downstream traffic have been changed with the convergence of data, voice and video over IP. For example, video

telephony requires equal bandwidths for the upstream and downstream traffic, however, IPTV applications have more downstream traffic than the upstream, whereas, applications such as music and content sharing servers have greater upstream bandwidth requirements than the downstream. Therefore, a real-time co-ordination is required for better resource management as per the application's traffic demand and to avoid collisions among wireless stations operating in the same service area. This work focuses on the development of a new MAC protocol which addresses the channel access challenges due to VVoIP traffic in the WLANs. The new MAC achieves a better performance than the traditional MAC schemes by minimising packet collisions and the transmission overheads.

WLAN research is primarily driven by researchers in academia and many small research centres. Due to the limited support for practical communication infrastructures, the researchers mostly use computer simulations as tools to build, evaluate and verify their proposed performance enhancement schemes. The behavior and characteristics of the new models can be predicted with simulators in a controlled environment, which is governed by numerous stochastic models [11, 12, 13]. The choice of a fixed set of stochastic models that depicts the most realistic scenario is a largely debated topic. For example, random numbers used in simulations are generated by computers, since a computer is a deterministic system, it often fall short to produce true random numbers [14, 15]. The computer follows a set of instructions that does not change, or uses constantly changing parameters, such as the time, in order to generate random numbers. Thus, according to many researchers, true random phenomenon of a system can not be modeled by a computer. For practical implementations and commercialisation, providing a proof-of-concept on a real hardware platform is essential. The process of selecting the right hardware is not trivial and researchers need to carefully consider numerous attributes such as the cost, availability of open

source code, maturity of hardware, integrated development environment (IDE) and time for development. It can be noted that most commercially available hardware does not provide access to source code of the lower layers, especially MAC and physical layer (PHY). Currently, two hardware platforms are primarily used for WLAN research: i) wireless open access research platform for networks (WARPnet) developed by Rice university [16], and ii) CalRADIO1 [17]. Details of these two platforms are discussed in Section 2.2. CalRADIO1 is a very basic prototype design and not suitable for implementing advanced MAC techniques. In comparison, WARPnet is an efficient platform for WLAN research, but it is very expensive and often not affordable for implementing a WLAN scenario that includes numerous nodes. These reasons motivated this work to conduct a research on cost effective platforms and solutions that can be used for implementing and testing of advanced MAC protocols in a WLAN.

1.5 Research Contributions

The contributions of this thesis are in two key areas: development of an efficient MAC protocol for VVoIP applications, and the development of an embedded platform for MAC protocol validation. Each of these are described below.

1.5.1 Development of an Efficient MAC Protocol for VVoIP Applications

In this work, we propose a point-coordinated round-robin MAC scheduler with a better designed service interval cycle (i.e. polling cycle) after taking into account International Telecommunication Union (ITU-T) recommendations and the symmetric nature of VVoIP traffic. This new MAC scheduler has been

named as TI-MAC (Traffic Intelligent - MAC). Chapter 3 describes the TI-MAC protocol in detail with an example. The proposed algorithm provides better synchronisation between AP and stations (STAs), deterministic IP packet transfer delay (IPTD) and very low IP packet delay variation (IPDV), commonly known as jitter. The proposed algorithm has been implemented using the 802.11 packet formats, and embedded in Atheros chipset based hardware. The IEEE 802.11e EDCA is the usual QoS scheme in commercial routers. Thus, we have undertaken a performance comparison against the IEEE 802.11e EDCA on the same hardware platform.

1.5.2 Development of an Embedded Platform for MAC Protocol Validation

This work presents an Atheros SoC based hardware platform embedded with firmware built from an open source Linux software stack, and the related testbed environment. The hardware is easily available in the retail market, and provides a convenient way to build a link with a Host-PC for programming and debugging purposes. The complete source code for various networking layers is available from OpenWRT website [18]. We have used the Atheros’s “ath9k” device driver module [19], which is available as Open Source in Linux distribution, to implement the TI-MAC algorithm and to demonstrate the capabilities of the proposed hardware platform. The “ath9k” device driver comes with the IEEE 802.11e EDCA compliant QoS implementation. Therefore, this platform is also suitable for comparing the performance of newly developed MAC protocols against the 802.11e EDCA. This hardware platform and the related embedded software development is described in Chapter 4.

1.6 Thesis Structure

Chapter 1 Introduction Describes the growth of multimedia traffic in WLANs, motivation behind this research, the contributions to this research, and the structure of this thesis.

Chapter 2 Background and Literature Review Presents an overview of literature review. This chapter discusses the IEEE 802.11 WLAN standard and the various channel access schemes e.g. DCF, PCF, EDCA and HCCA. This chapter also discusses about the hardware platforms available for the implementation and evaluation of new MAC protocols. This chapter finishes with an introduction to real-time IP traffic attributes.

Chapter 3 Traffic Intelligent MAC (TI-MAC) Describes the proposed MAC scheduler for real-time applications. This chapter discusses the motivation behind TI-MAC and its functional specifications. This chapter also presents the admission control policy of TI-MAC which is followed by an example of Skype video telephony.

Chapter 4 Netgear Hardware and TI-MAC Firmware Development Presents an off-the-shelf available WLAN router manufactured by Netgear and a Linux distribution, called OpenWRT. This chapter describes the internal hardware details of this router followed by various steps to set-up an OpenWRT based embedded platform for the firmware development and debugging. This chapter also describes network architecture implemented in OpenWRT stack and the functional flow in the path of packet transmission and reception. The source code of TI-MAC scheduler is presented in appendix A.

Chapter 5 Test Set-up and Characterisation Presents the testbed set-up for TI-MAC performance evaluation. This chapter describes two test scenarios for video IP telephony at image resolutions: i) 128×96 , and ii) 352×288 . This chapter also presents various graphs, which show the results achieved with TI-MAC and the IEEE 802.11 EDCA. This is followed by a result analysis for both test scenarios.

Chapter 6 Conclusion and Future Works Presents a summary of the contributions of this work, related literature, test scenarios and results, and suggests ideas for future work.

Chapter 2

Background and Literature Review

As discussed in Chapter 1, WLANs are gaining popularity both in home and business environments. The IEEE 802 standard committee defines the recommendations for the WLAN systems to make various devices compatible with each other and released the first WLAN standard in 1997 as the IEEE 802.11. This standard incorporates the guidelines for WLAN device manufacturers for the implementation of the MAC and PHY protocols. The committee in the subsequent years released various amendments such as the IEEE 802.11/a/b/g to support higher data rates and OFDM for better reception. The initial standards did not focus on the QoS for multimedia applications which are sensitive to real-time attributes such as delay and jitter. In 2005, the IEEE 802.11e amendments were approved by the committee to provide an enhanced QoS for the real-time multimedia traffic. This chapter discusses various MAC schemes in the IEEE 802.11 standards and other related works that have been done to improve the QoS experience for multimedia traffic. This chapter also discusses

available hardware options for the WLAN research and their limitations.

2.1 Background Study of MAC Related Work

The IEEE 802.11 is commonly adopted MAC and PHY standard in Wi-Fi commercial systems. DCF, PCF, EDCA and HCCA are the various MAC protocols defined in this standard. This section presents an overview on functional specifications of the these protocols and their QoS performance. A point-coordinated channel access scheme named wireless token network (WTN) proposed by Centre For Communications and Engineering Research (CCER), Edith Cowan University (ECU) has also been discussed here [20]. This is followed by a discussion on other related work published by various researchers.

2.1.1 The IEEE 802.11 WLAN Standard

The IEEE 802.11 standard [9] defines MAC and PHY specifications for wireless connectivity for fixed, portable and moving stations (STA) in a service area. Specifically, this standard:

- describes the functions and services required by the IEEE 802.11 compliant devices to operate in ad-hoc and infrastructure networks.
- defines the MAC protocol to support the asynchronous MAC service data units (MSDU) delivery services.
- defines several PHY signaling techniques and interface functions controlled by the IEEE 802.11 MAC.

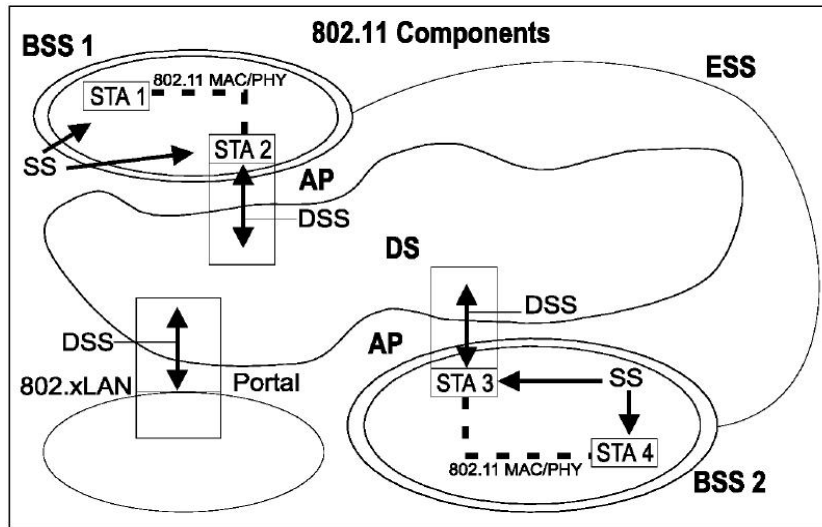


Figure 2.1: The IEEE 802.11 system architecture

- defines the requirements and procedures for data confidentiality in wireless medium.

The IEEE 802.11 standard committee defines the MAC functions that are based upon CSMA/CA (Carrier Sensing Multiple Access / Collision Avoidance). Each data-frame transmitted by a STA can be acknowledged by an acknowledgment (ACK) packet from the recipient. 'No ACK' is considered as packet undelivered and must be retransmitted by the source STA. The collision avoidance is primarily achieved by physical channel sensing, inter-frame spacing and a random back-off contention window (CW). There is also a provision of request to send (RTS) and clear to send (CTS) frames which can be used to avoid collisions from hidden STAs. The IEEE 802.11 did not define any guidelines to achieve QoS until the emergence of the IEEE 802.11e draft. A typical IEEE 802.11 system elements are shown in Fig. 2.1 [9].

The key elements in an 802.11 network are as follows:

Station (STA): Any device that contains the IEEE 802.11 compliant MAC and PHY interface to the wireless medium (WM).

Access Point (AP): Any entity that has STA functionality and provides access to the distribution services, via the wireless medium (WM) for associated STAs.

Basic Service Set (BSS): A set of STAs that have successfully synchronised with an access point. Two or more BSS are connected via a wired network through access points.

Distribution System Service (DSS): The set of services provided by the distribution system (DS) that enable the MAC to transport MSDUs between the STAs that are not in direct communication with each other over a single instance of the WM. These services include: i) transport of MSDUs between the access points (APs) of basic service sets (BSSs) within an extended service set (ESS), ii) transport of MSDUs between portals and BSSs within an ESS, and iii) transport of MSDUs between STAs in the same BSS in cases where the MSDU has a multicast or broadcast destination address or where the destination is an individual address and the STA is associated with an AP.

Extended Service Set (ESS): A set of one or more interconnected basic service sets (BSSs) that appears as a single BSS to the logical link control (LLC) layer at any STA associated with one of those BSSs.

2.1.2 Quality of service (QoS) in WLAN

The guaranteed QoS specifications [21] were drafted by Internet Engineering Task Force (IETF) in September, 1997. This specification describes the network element behaviour required to deliver a guaranteed service e.g. guaranteed delay and bandwidth in the internet. Guaranteed service provides firm and

mathematically provable bounds on end-to-end datagram queuing delays. This service makes it possible to provide a service that guarantees both delay and bandwidth. There are primarily two types of delay in packet transmission: i) a fixed delay (e.g. transmission delays), and ii) a queuing delay. The fixed delay is a property of the chosen path, which is determined not by guaranteed service, but by the routing mechanism. Only queuing delay, which generally results in jitter, is determined by guaranteed service. Guaranteed service guarantees that datagrams arrive within the guaranteed delivery time and are not discarded due to queue overflows, provided that the traffic stays within its pledged traffic parameters. This service is intended for applications which need a firm guarantee that a datagram arrives no later than a certain time after it was transmitted by its source. For example, the audio and video "play-back" applications are intolerant of any datagram arriving after their play-back time. Applications that have hard real-time requirements also require guaranteed service.

QoS provisioning is a very challenging task in WLAN systems. Packet collisions, delay and jitter are the key areas to be addressed in a WLAN system [22]. Initial development of the IEEE 802.11 does not consider much about guaranteed QoS for throughput and delay sensitive multimedia applications. The IEEE 802.11 defines two MAC functions: i) DCF (Distributed Coordinated Function) and ii) PCF (Point Coordinated Function). The PCF is seldom implemented because of high complexity and lack of robustness [23]. In addition, it may result in poor performance as demonstrated in [24, 25]. Presently, the IEEE 802.11e working group is working on delivering guaranteed QoS for real-time applications.

2.1.3 Distributed Coordinated Function (DCF) in IEEE 802.11 and QoS Performance

In DCF, all STAs wait for a common DCF Inter-frame Space (DIFS) period once the channel is sensed idle by the network hardware. STAs further wait for a random back-off period slot in the interval $[0, CW]$. The random back-off slot is used to avoid a potential possibility of conflicts among the STAs sensing the channel and ready to send data after DIFS at the same time. The range of random function which calculates the random back-off time is same for all STAs and all traffic categories. In this model, the probability of having the same random number and hence collision, increases with an increase in number of STAs. Moreover, all types of traffic are hosted equally and queued in a single queue. The real-time data has to wait for the clearance of the front queue data before being sent over-the-air, and at some occasions may have to wait for a longer time than the accepted upper bound of delivery time. Thus, the DCF is highly unsuitable for video IP telephony applications where the clients and the access point continuously send data due to higher data rate requirements. Fig. 2.2 [9] shows a typical timing cycle in 802.11 DCF medium access mechanism.

2.1.4 The IEEE 802.11e Standard for QoS

The IEEE 802.11e [9] standard defines a mechanism for QoS provisioning in WLAN systems which was not addressed in the IEEE 802.11. It provides two access control functions to support QoS: i) Hybrid Coordination Function (HCF) Channel Access (HCCA) for controlled access, and ii) Enhanced Distributed Channel Access (EDCA) for distributed access.

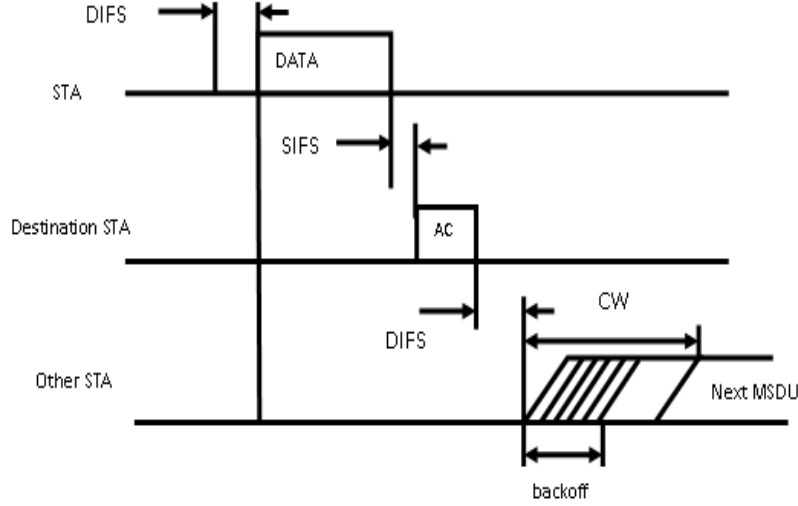


Figure 2.2: DCF timing cycle diagram

2.1.4.1 Hybrid Coordination Function (HCF) Channel Access (HCCA)

HCCA [26] in the IEEE 802.11e is a medium access method that is designed to provide hard and parametrised QoS guarantee for critical applications. HCCA works on the concept of bandwidth reservation for the stations and is controlled by a hybrid coordinator (HC). The IEEE 802.11e HCCA introduces a concept of Traffic Stream (TS), where TS is a set of MAC protocol data units (MPDUs) to be delivered subject to QoS parameters in a traffic specifications (TSPEC). A non-access point (non-AP) STA based on its TSPEC requirements requests the HC for TXOPs, both for its own transmissions as well as for transmissions from the AP to itself. The HC, which is collocated at the AP, either accepts or rejects the request based on an admission control policy. If the request is accepted, the HC schedules TXOPs for both the AP and the non-AP STA. A STA can request for maximum eight TSs with AP. All of these TSs are serviced by TXOPs at a fixed scheduled service interval (SI). The IEEE 802.11e HCCA only provides recommendations rather than a compliance standard, and

hence, due to non-availability of any industry standard HCCA scheduler, HCCA is not widely implemented in wireless LAN systems.

2.1.4.2 Enhanced Distributed Channel Access (EDCA)

EDCA of the IEEE 802.11e defines a mechanism for channel access based on traffic category. As per this mechanism, the traffic is divided into four categories with different priorities: i) Background (AC_BK, Lowest priority), ii) Best Effort (AC_BE), iii) Video (AC_VI), and iv) Voice (AC_VO, Highest Priority). The differentiation of services to each traffic category is achieved by the following parameters:

- amount of time a STA senses the channel to be idle before backoff or transmission.
- the length of the contention window to be used for the backoff.
- the duration a STA may transmit after it acquires the channel.

EDCA resolves the problem of contentions among different traffic inside the station queues. However, if all the STAs have traffic of equal priority in their transmit queues and want channel access, wireless channel contention becomes similar to DCF. This leads to obvious problems of DCF, e.g. packet collisions, random packet transfer delay and jitter. Moreover, EDCA is not a viable solution for symmetric traffic. Authors in [10] show that the upstream traffic is served well in EDCA, but the downstream throughput is not sufficient for good quality video. This work also observed the same behavior in the hardware test set-up based evaluation. Therefore, a true QoS can not be delivered by EDCA.

2.1.5 Wireless Token Network (WTN)

Wireless Token Network [10], [20] is a point coordinated Medium Access Control protocol proposed by Wyatt et al at CCER, ECU. WTN is a time division multiplexed (TDM) token passing network with an admission control mechanism. Due to the TDM nature of the upstream and downstream traffic at the AP, WTN provides a dedicated window of network access time to the AP's traffic (i.e. downstream). The network is centralised with all management functions residing and running in the AP. A client can not send the MSDU unless it holds a valid token, thus removing the possibility of collisions.

However, WTN has the following limitations:

- The timing cycle considered for evaluation has a duration of 130 ms, which leaves very little room for end-to-end delay [27].
- There is no mechanism for token recovery. In the case of a token being lost, the whole TXOP will be missed for the respective client.
- The downstream time reserved in the polling cycle is roughly twice the upstream time, whereas, the total data for the upstream and downstream is nearly the same. It shows a lack of synchronisation between the AP and STAs.

This work has considered WTN a reference model and provided the improvements over WTN after careful considerations of the above mentioned limitations.

2.1.6 Other Related MAC Works

Symmetric real-time data applications over WLAN have not been adequately studied and evaluated. Some techniques have been proposed to improve the QoS for VoIP applications in WLANs. Authors in [28], [29], [30], [31], [32], [33] proposed various schemes to resolve VoIP issues for voice telephony, but none of them addressed the issues arising from the symmetric nature of VoIP traffic. In video IP telephony, the problem is different because it demands more bandwidth and has symmetric traffic, which makes the scenario more complex. In [34], authors evaluated a video conferencing scenario over the IEEE 802.11g using DCF and EDCA, but did not propose any enhancement. Authors in [35, 36], [37], [38], [39], [40] proposed various schemes to support voice and video over WLAN but for one-way streaming only. In [41], Bejerano *et al.* solve the problem of downstream traffic bottlenecks in AP by decreasing CW_{\min} , but they assumed a video streaming scenario and ignored the upstream traffic. In [42], authors proposed an adaptive polling scheme to allocate transmission opportunity (TXOP) periods to upstream traffic, but for downstream traffic, EDCA is used, which does not provide an acceptable QoS experience for receiving stations.

In [43], authors clearly defined a symmetric voice and video scenario, and they proposed a scheme where video quality suffered while serving voice. They temporarily assigned a higher priority to the AP to resolve the downstream traffic bottlenecks in access category-3 (AC3). By assigning a higher priority to downstream traffic, the upstream traffic is virtually treated as best-effort traffic. This approach leads to traffic congestion and more packet collisions during the upstream transmission. IPDV is another bottleneck with this approach which is inherent in EDCA.

Further, the spectralink voice priority (SVP) protocol [44], developed by Spectralink, gives exceptionally stable connections [10] in the upstream direc-

tion which would make it suitable for use in unidirectional real-time broadcast services. However, it cannot support symmetrical bidirectional traffic with high load.

2.2 Background Study of Hardware Platforms for WLAN Research

The researchers prefer to implement and test their new algorithms in a hardware to get real-time performance. There are only a few hardware available in the field of WLAN research. This section presents an overview of such hardware that are designed for the purpose of research.

2.2.1 Wireless Open-Access Research Platform For Networks (WARPnet) – An FPGA Platform

WARPnet, developed by the Centre For Multimedia Communication, Rice University, is an FPGA based platform for real-time validation of proposed schemes in 802.11 WLANs. WARPnet has flexibility to customise all the networking layers and comes with custom software for PHY and MAC layers. This platform is also a good solution for PHY level research where it has excellent flexibility to change the PHY level attributes. In WARPnet, the 802.11 MAC is implemented in C-code as a simple Carrier Sense Medium Access MAC (CSMA-MAC) [16] without any traffic prioritised queues. Researchers commonly use advanced MAC schemes such as the 802.11e EDCA and HCCA as references to benchmark their own proposed solutions. Both of these schemes are based on the priority of application traffic, and multiple transmission queues are used for packet transmission. Therefore, researchers have to implement a priority queue based soft-MAC from scratch, which requires a reasonable amount of time and

effort. Moreover, the cost of one FPGA kit is around US\$6,500. To implement a WLAN with single AP and 20 clients, for example, requires 21 FPGA kits, which costs around US\$136,500 in total. This makes WLAN implementation using WARPnet an expensive proposition, which many academic researchers may not afford.

2.2.2 CalRADIO1

CalRADIO1 [17] is a low-cost platform used for the 802.11b MAC layer research purposes. It runs on an ARM7 processor which executes the networking layer and functions above. The MAC layer is implemented in C-code and is driven by a TMS320VC5471 Texas Instruments digital signal processor (DSP). There is a memory-mapped buffer in the DSP which is referenced by both the ARM7 and the DSP for exchange of data packets and control information. CalRADIO1 comes with a minimal software development kit which includes a basic implementation of the MAC protocol. There are no prioritised queues for different categories of traffic as stated in the IEEE 802.11e to implement QoS requirements. Therefore, the user has to implement a soft-MAC with prioritised queues and a software control for priority based transmissions. In summary, CalRADIO1 is a good low-cost platform for MAC layer development, but in terms of software availability it comes only with a single queue MAC implementation.

2.2.3 Sagrad STLC4560 Radio Card with STM32 Eval Board

This is an integrated solution built upon a STM32 [45] microprocessor evaluation board and a wireless radio card manufactured by Sagrad [46], [47]. The wireless radio card is designed from an STLC4560 [48], a single chip 802.11b/g

WLAN radio chip. The wireless radio card has a serial peripheral interface (SPI) and secure digital I/O (SDIO) interfaces to communicate with the host STM32 board. This custom solution has cost advantages but doesn't implement a full soft-MAC. Some of the MAC functions are implemented in the hardware which gives no control to the programmer. This hardware comes with a lower MAC (LMAC) software binary, and a header file which contains only the application programming interface (API) prototype declarations. There is no access to the source code of the APIs. Hence, this hardware is not suitable for the new MAC firmware implementation where control over all parameters is desired.

There are other similar kinds of hardware which are developed on SoC from Broadcom and Texas Instruments but most often these come with a binary library of the LMAC rather than full open source code. Open source is an utmost requirement for a new MAC scheme to be implemented in software. This limitation leads us towards evaluating Linux based hardware. These are, in general, available with open source code under the GNU public license (GPL). After careful consideration of different hardware and its suitability for new MAC scheme development, this work found an AP manufactured by Netgear. This AP is readily available off-the-shelf at very low price. This hardware can also be trusted for its reliability and performance characteristics because it is a final customer product and would have gone through rigorous testing and compliance checks. Whereas, the research platforms discussed above are in the prototype stage and may not have gone through the extensive testing of a commercial product. Chapter 4 describes more in detail about all the features of this hardware.

2.3 Introduction to Real-Time IP Traffic Attributes

International Telecommunication Union (ITU-T) recommendations [8] Y.1540 and Y.1541 define five IP packet transfer performance parameters. Among these, IPTD, IPDV and IP packet loss ratio (IPLR) are three major performance evaluation criteria for real-time data applications.

A low IPLR is desirable for multimedia packet delivery. For example, H.263 or H.264 compressed video consists of I (Intra Frame), P (Predictive Inter Frame) and B (Bi-Predictive Inter Frame) frames. Decoding of P frames depends on earlier I or P frames and decoding of B frames depends on earlier and future frames. A loss of any B or P frames will have a negative and cascaded impact on video quality. Further, real-time packets should be delivered to the final recipient within an acceptable end-to-end transfer delay (i.e. $IPTD \leq T_{max}$, where T_{max} is application dependent). An ideal system should have a fixed end-to-end transfer delay. A variation in IPTD is called IP packet delay variation (IPDV), and leads to jitter in event synchronisation at the destination. For example, lip-synch is a common problem while decoding voice and video because received packets of clock recovery are not referenced with voice and video events.

Tables 2.1 and 2.2 [8] show the ITU-T's recommended upper bound values for QoS network performance parameters and guidance for IP QoS classes, respectively.

Table 2.1: Network Performance Parameters				
Network Performance Parameter	Class 0	Class 1	Class 2	
IPTD (ms)	100	400	100	
IPDV (ms)	50	50	U*	
IPLR	1×10^{-3}	1×10^{-3}	1×10^{-3}	

* - unspecified

Table 2.2: Guidance for QoS Classes

QoS Class	Application (Example)	Node Mechanism
0	Real-time, Jitter Sensitive, High interaction (VoIP, Video Conferencing)	Separate Queue with priority servicing
1	Real-time, Jitter Sensitive, Interaction (VoIP, Video Conferencing)	Separate Queues
2	Transaction Data, Highly Interactive	Separate Queue, Drop priority

For video IP telephony, throughput requirements mainly depend on the pixel resolution and compression techniques used at the application layers. If we consider H.263 image with YC_bC_r 4:2:0 profile and H.264 with YC_bC_r 4:2:2 profile, with 30 frames per second, Table 2.3 shows approximate bandwidth requirement for standard ITU image resolutions used in video telephony [49], [50]. The compressed rates are scaled to the nearest 64 kbps multiple value to meet ITU-T H.320 [49] requirement. Table 2.4 [42] indicates bandwidth requirement for voice codecs.

Table 2.3: Image Resolution and Bandwidth Requirement

Resolution	H.263 Compressed Rate (bps)	H.264 Compressed Rate (bps)
128×96	128000	128000
176×144	320000	256000
352×288	1216000	832000

Table 2.4: Voice Data Bandwidth Requirement

Codec	Bit Rate (kbps)	Payload (Bytes)	Packet Duration (ms)
G.711	64	160/240	20/30
G.723.1	6.4	30	38.5
G.726	32	120	30

A Wi-Fi multimedia station with H.264 and G.711 codecs will require a minimum 192 kbps and 896 kbps in each direction to make a video IP call at 128×96 and 352×288 image resolutions, respectively.

2.4 Summary

In this chapter, a background of various MAC schemes in the IEEE 802.11 standard (e.g. DCF, EDCA, HCCA) have been discussed along their shortcomings to address the symmetric traffic nature for VVoIP applications. The other related works of various researchers have also been summarised here. This chapter also presents various hardware available for WLAN research and their limitations. At the end, this chapter highlights an AP which is manufactured by Netgear and embedded with a firmware image built from a Linux open source software stack. Finally, a background of various real-time IP traffic attributes and terminologies has been discussed here, which would be frequently used in the rest of this thesis.

Chapter 3

Traffic Intelligent MAC (TI-MAC)

The TI-MAC is a smart point-coordinated scheduler that provides channel access opportunities to the clients as per their agreed parameters. It has a strict admission control policy where new clients are not entertained so that a pledged QoS can be provided to the existing clients. At the same time, it provides a flexibility to the clients to utilise their bandwidth as per the ongoing IP traffic symmetry ratio. A wireless client has two-directional traffic, called the upstream and the downstream traffic. The multimedia applications running in an associated client may have different ratios of upstream and downstream traffic. A TI-MAC enabled WLAN client has freedom to use its assigned bandwidth only for the upstream data, only for the downstream data, or in a shared pattern as per the amount of upstream and downstream traffic. Thus, TI-MAC provides an opportunity to the clients to decide the QoS fairness by themselves rather than via the AP.

3.1 Motivation Behind TI-MAC

The IEEE 802.11 committee took an initiative to offer QoS in WLAN by drafting the IEEE 802.11e standard, a much needed standard to meet QoS requirements for bandwidth and time-delay critical applications. The standard proposed MAC enhancements on the basis of prioritised and parametrised traffics. Prioritised channel access is provided through the EDCA scheme where a high channel access probability is given to a higher priority traffic or having higher QoS constraints than a lower priority traffic. Because of the contention-based nature, EDCA cannot provide a strict QoS, though it may only provide a relative QoS. Due to the distributed and non-deterministic behaviour of EDCA, it is not suitable for real time and performance critical applications (e.g., video telephony or live streaming).

The HCCA in the IEEE 802.11e standard is another polling based MAC scheduler developed for providing guaranteed and parametrised QoS to performance critical applications. The HCCA is controlled by a hybrid coordinator (HC) which is collocated with the access point. HCCA introduces a concept of traffic stream (TS), and each TS is associated with a set of traffic specifications (TSPEC). The TSPEC for a TS is negotiated by a QoS STA (QSTA) with the HC, before a TS can be served through HCCA. After the successful negotiation of the TS, transmission opportunities (TXOP) are granted to the TS at specific service intervals (SI). A QSTA can negotiate up to eight TSs with the HC. The length of TXOP is decided by the scheduler on the basis of traffic information that the HC has for the TS. HCCA also implements an admission control policy such that if a TS cannot be served with the TSPEC under negotiations, the access will be denied. Thus, HCCA provides a mechanism for resource reservation and service intervals to conform to the pledged QoS guarantees. However, HCCA has its own disadvantages, which limit its popularity

in practical systems. Firstly, HCCA demands a complex software architecture and is not a viable option for resource constrained hardware systems. Secondly, the negotiation for a TS with the HC requires extra data exchange. This adds significant overhead. The HCCA is not suitable for short-time traffic streams because the overhead itself may become equivalent to actual stream data. When a stream is either under or over utilising resources than the reserved amount, a new negotiation is required for readjusting resource reservation. The QSTA does not have the authority itself to readjust the bandwidth within the various TSs. It is evident that, because of similar kinds of HCCA constraints, there is hardly a vendor independent HCCA compliant access point available in the retail market.

The above mentioned limitations of EDCA and HCCA encouraged this work to develop an efficient MAC scheme, which can offer guaranteed QoS like HCCA but with minimum overheads, and can control the reserved resources as per the traffic symmetry requirements of the QSTA.

3.2 TI-MAC - Functional Specifications and Admission Control

The TI-MAC is a polling cycle based scheduler which is co-located in the access point. Similar to the HCCA, the TI-MAC provides collision-free channel access where a STA is allowed to send the packets only when polled. Each STA has been considered to have different traffic symmetry in the upstream and downstream direction. The nature of traffic in each direction may have different bandwidth requirements. These typically depend upon the type of application a STA is serving. For example, Wi-Fi enabled video surveillance

cameras want to upstream more while downstream only requires a few hundred bytes of data. However, a Wi-Fi enabled home video gateway requires more downstream bandwidth to send live or on-demand video streams but only few hundred bytes of user request data are needed in the upstream. As discussed earlier, video telephony application ideally needs equal bandwidth in each direction. The TI-MAC uses a mechanism of resource reservation to satisfy QoS guarantees, where the STAs have full control over utilising the reserved resources as per its “current” traffic requirements.

ITU-T recommendations Y.1540 and Y.1541 define five IP packet transfer performance parameters, but do not define any parameter for traffic symmetry classification. This work introduces a new concept of IP traffic symmetry ratio (IPSR) and uses the IPSR to control the upstream and downstream TXOP interval. The IPSR of a STA is defined as the ratio of upstream bandwidth to the total bi-directional bandwidth requirement. A station can declare its IPSR during a capability exchange period with the access point after analysing the average upstream and downstream bandwidth requirement. The IPSR can easily be updated by sending a new IPSR request to the access point. The access point uses this IPSR to intelligently control the upstream and downstream TXOPs from the pledged data bandwidth.

An AP controlled polling cycle starts with upstream period, and the AP unicasts a poll packet for the first station. The station retrieves its upstream TXOP from the poll packet and starts transmitting its packet queues as per Equation (3.4). At the end of TXOP, the station will send a “Txop_end” packet to the access point. Txop_end packet synchronises the AP and STAs, and helps to avoid any contention situation between AP and STAs on the boundaries of TXOPs. A missing poll packet results in the loss of the whole TXOP and a loss of Txop_end may cause the start of a new poll packet at the previous TXOP

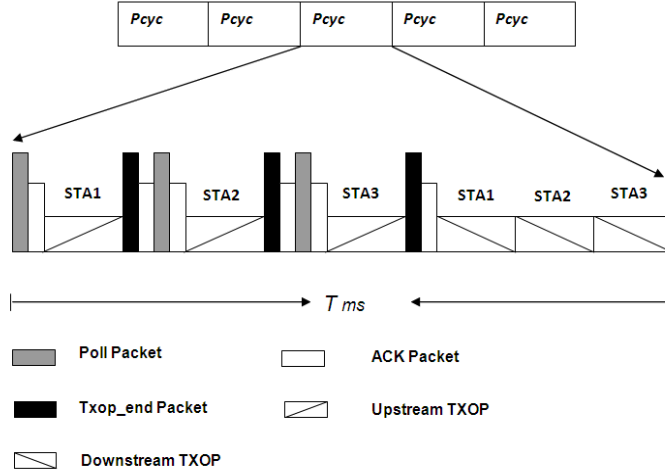


Figure 3.1: TI-MAC polling cycle timing diagram

time boundary. Thus, the poll packets and Txop_end packets are considered critical packets, and hence are always followed by an acknowledgment (ACK). Moreover, before transmitting a packet, the AP or STAs will confirm whether the remaining TXOP is sufficient to transmit the next MAC protocol data unit (MPDU). Once all the STAs are polled, the AP starts downstreaming for each station within their respective downstream TXOP, as per Equation (3.5). An example polling cycle timing diagram is shown in Fig. 3.1.

The TI-MAC is simple enough to implement in the software and works in a round-robin fashion. It defines an optimised polling cycle p_{cyc} of duration T_{ms} depending on IPTD, IPSR and system latencies, where all the associated stations are served within this duration as per their QoS agreement. The period of polling cycle should leave enough room for actual end-to-end IP packet transfer delay and expected jitter. The TI-MAC also suggests a simple admission control policy. If the access point does not have enough resource available, no new station will be associated or a low resolution service may be offered to such

stations to conform with the contracted QoS objectives.

Now, we will present the formula to calculate the TXOP per station (ξ_i) and outline the condition for the maximum number of stations (N) which can be associated in a polling cycle. The maximum number of stations which can be associated within p_{cyc} of duration T ms depends on the bi-directional mean data rate requirement per station (b_i), physical layer (PHY) transmission link rate (C), and system overheads (η) due to real-time scheduling in an embedded system.

The polling cycle duration is defined as:

$$T = IPTD - \beta \quad (3.1)$$

where, $IPTD$ is the upper bound end-to-end IP packet transfer delay, and β is a scaling factor which depends on various propagation delays from the point of origin to access point and codec latencies.

Now, the bi-directional mean data rate requirement per station (b_i *kbps*) is equally distributed between the total number of polling cycles in a second. Total number of polling cycles ($N_{p_{cyc}}$) can be defined as:

$$(N_{p_{cyc}}) = \frac{1000}{(T + \eta)} \quad (3.2)$$

TXOP per station will be allocated as per the following equation:

$$\xi_i = \frac{(1000 \times b_i)}{(C \times N_{p_{cyc}})} \times \delta \text{ ms} \quad (3.3)$$

where i is the station index and δ is the overhead index due to SIFS, Slot-time, Real-Time Protocol (RTP), User Datagram Protocol (UDP), IP, 802.11,

Physical Layer Convergence Protocol (PLCP) headers and PLCP preamble.

The total TXOP allocated to a station is used for the upstream and downstream as per their IPSR (μ_i), where the upstream TXOP ($\xi_i^{upstream}$) and the downstream TXOP ($\xi_i^{downstream}$) can be defined as:

$$\xi_i^{upstream} = \xi_i \times \mu_i \quad (3.4)$$

$$\xi_i^{downstream} = \xi_i \times (1 - \mu_i) \quad (3.5)$$

where μ_i is ratio of the mean upstream data rate ($b_i^{upstream}$) to the bi-directional mean data rate(b_i):

$$\mu_i = \frac{b_i^{upstream}}{b_i} \quad (3.6)$$

The total number of stations (N) which can be associated with an AP should meet the following condition:

$$\sum_{i=1}^N (\xi_i) \leq T \quad (3.7)$$

By using Equations (3.2) to (3.7), we can calculate the required upstream and downstream TXOP for each station. This approach can be followed for any symmetric and asymmetric traffic application with strict QoS requirements. In Section 3.3, we have demonstrated an example and used the above derived equations to calculate the upstream and downstream TXOPs and maximum number of STAs that can be associated with an access point.

3.3 Example of Skype Video Telephony

Let us consider a real-time traffic scenario where the maximum accepted value of IPTD is 150 to 200 *mSec*. Optical fibers are used instead of copper wires because of lower attenuation, lower delivery time, lower interference, and higher data capacity. The signal pulses travel at a speed of light in optical fibers. The speed of light is 300,000 kilometers per second in a vacuum, but in a medium, it drops down by an index of refraction. The larger the index of refraction, the slower the speed of light in that medium. The index of refraction for optical fibre is typically 1.52. Therefore, in optical fibres, a signal travels approximately 12,000 kilometers in 60 *mSec*, which is the distance between Sydney and New York [51]. Thus, the expected IPTD, or the packet delivery time, from PHY-to-PHY would be 60 *mSec*. The TI-MAC is a polling-based scheme with a guaranteed service interval. Hence, the jitter, or queuing delay, is close to 0. We may consider a further 60 *mSec* for the system latencies, encoding and decoding latencies, routing and switching delays etc. So, $\beta = 120$ *mSec*.

As per equation (3.1), $T = 200 - 120 = 80$ *mSec*

Next, in Netgear WNDR3700 [52] hardware test platform, we observed a system latency (η) of $\simeq 2$ *mSec*.

So, as per equation (3.2),

$$(N_{p_{cyc}}) = \frac{1000}{(80 + 2)} \simeq 12$$

Now, let us calculate the overhead index δ . Table 3.1 shows the various parameters and overheads for a typical real-time IP packet transmitted thorough WLAN interface. To calculate the value of δ , we have to first calculate the value of net throughput ψ that can be achieved at a PHY data rate (C) of 2 Mbps. However, the equations (3.8) to (3.12) can be used for all available PHY rates

Table 3.1: Parameters Used to Calculate Net Throughput at 2 Mbps

Parameter	Value
Payload	500 Bytes
RTP,UDP,IP,802.11	12,8,20,30 Bytes
SIFS	10 μSec
Slot Time	20 μSec
PLCP Preamble + Header	192 μSec
PHY Rate	2 Mbps
CW_{min}, CW_{max}	0, 0

defined in the IEEE 802.11 standard [9].

Assuming parameters in Table 3.1, each packet will consists of 570 bytes, which includes overheads due to RTP, UDP, IP, and 802.11 headers.

Time t_{ota} required to transmit this packet over-the-air:

$$t_{ota} = \frac{Bits}{C} = \frac{570 \times 8}{2 \times 10^6} = 2280 \mu Sec \quad (3.8)$$

Total time t_{Total} required to transmit this packet:

$$t_{Total} = t_{ota} + SIFS + SlotTime + PLCP = 2280 + 10 + 20 + 192 = 2502 \mu Sec \quad (3.9)$$

Total number of packets n transmitted over a second:

$$n = \frac{1}{2502 \times 10^{-6}} \approx 400 \quad (3.10)$$

Net data rate ψ obtained:

$$\psi = n \times Payload = 400 \times 500 \times 8 = 1.6 Mbps \quad (3.11)$$

$$\delta = \frac{C}{\psi} = \frac{2}{1.6} = 1.25 \quad (3.12)$$

For example, a Skype video call requires equal data rate for its upstream and downstream traffic with an IPSR of 0.5. As per Table 2.3 and 2.4, it requires an approximate bandwidth of 192 kbps in each direction to make a video call which comprises of H.264 compressed video with a pixel resolution of 128x96 and G.711 compressed audio. So, $b_i = 384kbps$.

As per equation (3.3), the reserved TXOP can be given as:

$$\xi_i = \frac{1000 \times 384}{2 \times 10^6 \times 12} \times 1.25 = 20mSec$$

As per equation (3.4) & (3.5), the upstream and downstream TXOP equals:

$$\xi_i^{upstream} = 20 \times 0.5 = 10mSec$$

$$\xi_i^{downstream} = 20 \times (1 - 0.5) = 10mSec$$

Further, as per equation (3.7), a maximum of four ($N = 4$) Skype video clients can be associated with an access point at a PHY data rate of 2 Mbps. Further video clients may be added, but at the cost of a poor quality of video IP call. Therefore, it is important to have bandwidth reservation for the clients to meet the agreed QoS requirements.

3.4 Summary

This chapter explains the functional specifications of the TI-MAC scheduler and the motivation behind the development of TI-MAC. The chapter starts with highlighting the inherent disadvantages of EDCA and HCCA. These are

the two MAC schemes drafted by the IEEE 802.11e committee to address the QoS requirements. The limitations of EDCA and HCCA triggered the motivation towards development of the proposed TI-MAC. This chapter also describes the TI-MAC polling mechanism, and the multiplexing of polling requests and TXOPs within a polling cycle. The TI-MAC requires a simpler software architecture compared to the complex architecture for HCCA. This chapter also shows the derivation of various equations for the calculation of upstream and downstream TXOP, as well as the condition for the maximum number of clients. At the end, an example of Skype video IP telephony has been presented with minutiae details, and all the derived equations have been used to find the TXOP values and the maximum number of clients.

Chapter 4

Netgear Hardware and TI-MAC Firmware Development

The validation of a new MAC algorithm on an embedded platform provides a true picture of the efficiency of the algorithm. This chapter discusses a Linux OS based embedded platform that can be used to implement a soft-MAC and validate the algorithm in a real-time environment. Linux is a preferred real-time operating system (RTOS) for the development of firmware or embedded software. Linux comes with the freedom to use and distribute, usually with no cost. Most of the embedded platforms have the same peripheral interfaces around the CPU e.g. USB, SATA, DDR and UART, and Linux allows the use of a common firmware for each of these peripherals in different embedded platforms. The firmware is available as Open Source and the developers have the freedom to customise the firmware as per their requirements and releasing

it back to the Linux community. Thus, software development for a platform can be achieved in a short time period by reusing existing source code in the Linux distribution. There are various customised Linux distributions available for various applications e.g. Linux distributions for desktop and servers as well as for mobile platforms. There is a similar type of customised distribution for WLAN platforms, called OpenWRT [53, 54]. OpenWRT has firmware support for various hardware platforms and SoC e.g. Netgear, D-Link, Linksys and Atheros. Netgear manufactures a wireless dual band gigabit router designed from an advanced SoC from Atheros, named WNDR3700, and the commercial firmware is compiled from OpenWRT Linux distribution. Therefore, OpenWRT and Netgear WNDR3700 make a nice platform to evaluate a new MAC protocol in practical hardware at a very low cost.

4.1 OpenWRT - Wireless Freedom

OpenWRT [55] is targeted for networking embedded platforms and offers tremendous support for off-the-shelf wireless routers such as Netgear, D-Link, Linksys, Asus, Atmel and Thomson. OpenWRT provides a complete customisable framework where the developers can select the packages to suit their applications. This means the user can remove unwanted packages to make room for the add-on packages. Because of the increasing popularity of OpenWRT and zero cost for Linux drivers, the manufacturers are also moving towards OpenWRT based software development. OpenWRT is currently being used in VoIP systems, sensor networks, industrial control systems, both wired and wireless networks and many more.



Figure 4.1: Netgear WNDR3700 router

4.2 WNDR3700 Router by Netgear

The WNDR3700 [52] is an IEEE 802.11 b/g/n 2.4 GHz and IEEE 802.11a/n 5.0 GHz dual band wireless router manufactured by Netgear. OpenWRT provides guidelines to establish a computer interface and compile a customised binary image for this router. Moreover, the hardware of this router is designed with the Atheros's advanced and high-performance wireless network processor SoC, the AR7161. The AR7161 provides a peripheral component interconnect (PCI) interface for WLAN modules. The WLAN module is designed with the Atheros's AR9280. The AR9280 is a single-chip dual band WLAN solution for PCI-express. This router is available off-the-shelf in the retail market at a cost of US\$200 only. Images of the WNDR3700 are shown in Figs. 4.1 and 4.2. The next two subsections elaborate further about the chipsets used for this hardware.

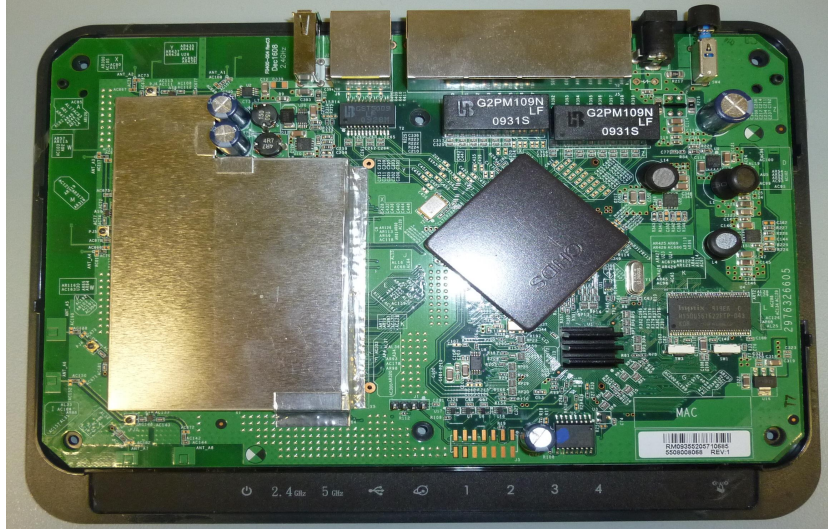


Figure 4.2: Netgear WNDR3700 hardware

Table 4.1: AR7161 Technical Specifications

Parameter	Specification
Processor Core	MIPS 24K
Processor Speed	680 MHz
Memory Interface	DDR1/SDRAM
Flash	SPI* Serial Flash
USB	USB 2.0 Host
Ethernet	MII/RMII/GMII/RGMII

SPI - Serial Peripheral Interface

4.2.1 AR7161 - Atheros Wireless Network Processor

The AR7161 [56] is Atheros's high-performance wireless network processor which enables an efficient design for the solutions that target multiple services like video, audio and data, simultaneously. It also provides a PCI 2.3 interface to connect the WLAN modules, 2xgigabit ethernet ports for LAN and WAN connectivity, 2xUSB interface and a DDR interface to support high-speed memory. A top-level architectural diagram is shown in Fig. 4.3 and technical specifications are summarised in Table 4.1.

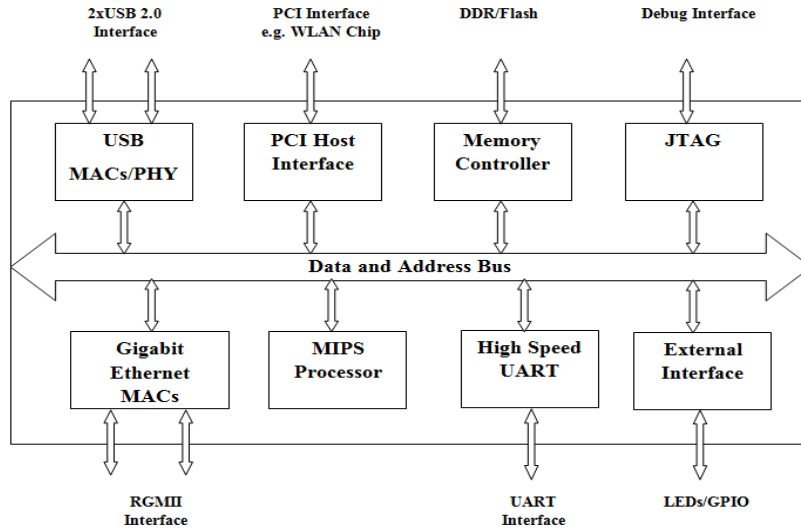


Figure 4.3: Top-level architectural block diagram of AR7161

4.2.2 AR9280 - Atheros Single-Chip 802.11a/b/g/n WLAN Chip

The AR9280 [57] is a highly integrated single-chip solution for 2.5 GHz and 5.0 GHz WLAN. It also enables 2xMIMO (Multiple-Input-Multiple-Output) configuration for applications demanding high throughput and robust link quality. The AR9280 is IEEE 802.11n compliant and has an integrated soft-MAC, baseband processor and a PCIe (PCI Express) interface. The top-level architectural diagram is depicted in Fig. 4.4 [57].

The AR9280 supports two concurrent traffic streams using up to two integrated transmit chains and receive chains for high throughput and performance. The transmit chain combines baseband in-phase (I) and quadrature (Q) signals, converts to desired radio frequency, and passes the RF signal to multiple antennas. The receive chain produces I and Q signals after down-converting the received RF signal to the baseband frequency. Further, the IEEE 802.11 MAC

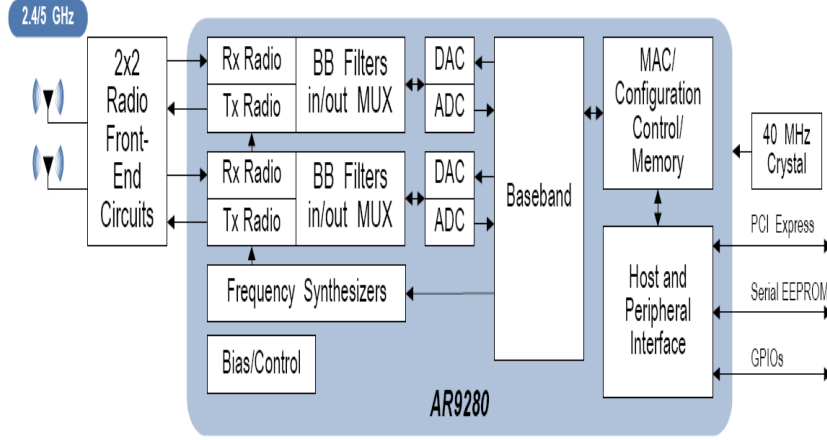


Figure 4.4: Top-Level architectural block diagram of AR9280

data services are provided by the MAC inside the AR9280. The AR9280 comprises of a full software MAC which means that a host processor (e.g. the AR7161) can control all MAC functions. None of the MAC data service control functions run inside the AR9280 as firmware as in a hard-MAC. This feature provides full liberty to the external host to control Tx and Rx queues processing. The host controls the AR9280 by programming configuration registers through the PCIe interface. The MAC consists of ten queue control units (QCU) and ten distributed coordination function (DCF) control units (DCU) for the transmission of MPDUs. Each QCU is associated with one DCU. The top 2 DCUs are used for beacon frame handling purposes and rest of the eight DCUs are assigned priorities as per the enhanced distributed channel access function (ED-CAF) priority levels. The DCU with the highest priority is given the channel access by the arbiter. A Tx descriptor holds the memory pointers to the transmit buffer, RTS and CTS control, ACK control and other controls for the physical block. The QCU parses these descriptors and prepares a new data or control

frame for the DCU. Once it has received channel access, the DCU transfers the data or control frame to the physical block to transmit over-the-air. Similarly, a received frame at the physical block is immediately transferred to the Rx queue and an interrupt is signaled to the host to read the data frame. All the parameters of the Tx and Rx queues are host programmable which makes the AR9280 an excellent device for MAC protocol research.

4.3 Hardware-PC Interface Setup

A personal computer (PC) or laptop with a Linux OS is required to develop an interface with the Netgear WNDR3700 hardware. For Linux installation, a virtual machine environment with Virtual Machine (VM) player was created in the PC, also, the Ubuntu 10.04 (desktop edition) Linux OS was installed from VM player. VM player makes an excellent arrangement for Linux OS within a Windows machine. It is very helpful for users who are new to Linux OS and do not want Linux as the main OS of their computer. Ubuntu 10.04 (desktop edition) comes with a nice graphical desktop manager called GNOME. GNOME can be used for file-system browsing, files and directory management, application launching, and other similar jobs without the need of a command line interface (CLI). This distribution is also embedded with a graphical package manager called Synaptic Package Manager (SPM) and can be used for new package installation, upgrade and package removal.

On the hardware platform, there are four LAN ports which may be used to transfer files in memory. The files in this research work refer to the newly compiled binary image of the developed firmware. A trivial file transfer protocol (TFTP) connection was established between the LAN port of the hardware and

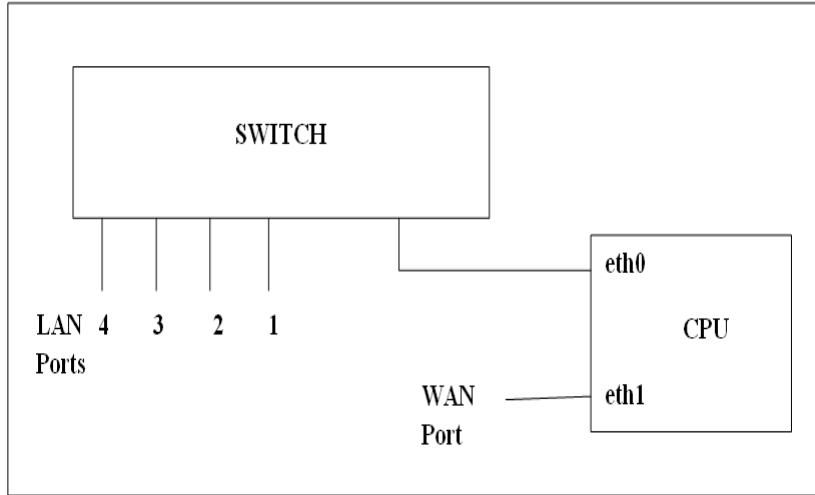


Figure 4.5: Ethernet ports connection with CPU

the Host-PC, to program the newly compiled firmware in the memory of the hardware. The hardware boots with the standard bootloader (uboot) which enables the ethernet0 (eth0) port. Eth0 is connected with the LAN ports through a switch. A connection block diagram for the ethernet ports is shown in Fig. 4.5. There are two other interfaces available on-board to debug the software: i) Universal Asynchronous Transmitter Receiver (UART), and ii) Joint Action Test Group (JTAG). This work used the UART interface, also commonly known as serial interface, to monitor the software execution in a hyperterminal window of the Host-PC and to send bash commands from a command line interface. The UART link can be built by a modified Nokia CA-42 USB-to-Serial cable. This cable is available off-the-shelf in the electronics stores and costs around US\$10. Once the hardware is flashed with the newly compiled firmware image, the interface can be disconnected, allowing the hardware to work as a standalone wireless device.

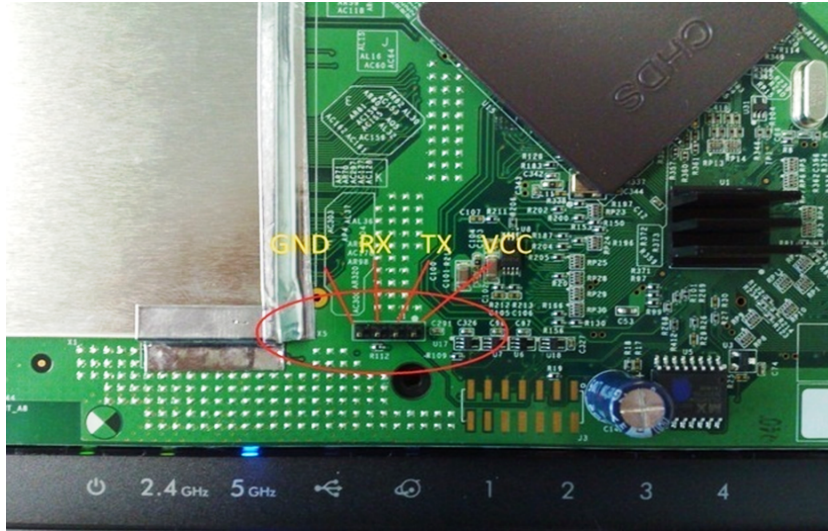


Figure 4.6: Image of hardware inside Netgear WNDR3700

4.3.1 Custom Build Serial Interface Cable

As discussed above, a simple USB-to-Serial interface can be built with a Nokia CA-42 data cable. The red eclipse in Fig. 4.6 shows a serial connector “J1” on the aforementioned hardware to plug into the serial connector. The custom built cable was connected between the USB interface of the Host-PC and the connector “J1” of the hardware. A Nokia CA-42 USB-to-Serial cable which was used for this purpose is shown in Fig. 4.7.

To modify the cable, the serial connector of this cable was chopped-off revealing three wires with colors - blue, green, and white. The associativity was later established between these wires and pins on the serial connector with the help of a multimeter. The wire labels, with their respective color and pin number, are shown in Table 4.2. These three wires were later soldered on a single row, three pin berg-stick, as per Table 4.2 and Fig 4.8. The final shape of custom build cable is shown in Fig. 4.9.



Figure 4.7: Nokia CA-42 USB-to-Serial cable

Table 4.2: Wires Color and Label

Wire Color	Pin	Label
Blue	6	3.0V Rx Data
Green	7	3.0V Tx Data
White	8	Gnd

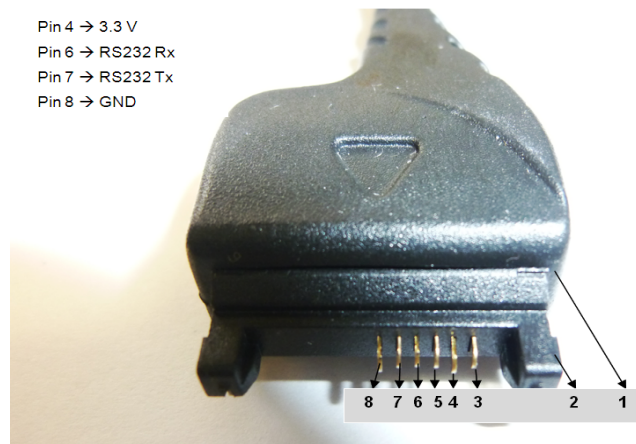


Figure 4.8: Serial end pin configuration of Nokia CA-42 cable



Figure 4.9: Custom build cable for the serial interface with the WNDR3700

4.3.2 Procedure to Program the Hardware

To program the targeted hardware, a trivial file transfer protocol (TFTP) server-client model was used. A TFTP server was set-up in the Host-PC, while a TFTP client was started in the hardware resulting from bootloader code. A hyper-terminal session (working as a front-end controller for the WNDR3700 hardware) was also opened and used to send commands to the hardware. Linux Host-PC has a “minicom” utility to start a hyper-terminal session. The TFTP client running in the hardware requests the TFTP server to transfer the binary image by sending various commands from the minicom terminal.

4.3.2.1 Set-up of the TFTP Server in the Host-PC

The steps for setting-up a TFTP server are as follows:

```

service tftp
{
    protocol      = udp
    port          = 69
    socket_type   = dgram
    wait          = yes
    user          = nobody
    server        = /usr/sbin/in.tftpd
    server_args   = -s /srv/tftpboot
    disable       = no
}

```

Figure 4.10: TFTP script source code

1. Write a tftp script named “tftp” and save in */etc/xinetd.d*. Fig. 4.10 shows the source code of the script.
2. Start the tftp server from a command line interface window by executing the command: */etc/int.d/xinetd restart*. Fig. 4.11 shows the screen-shot after the command execution.

4.3.2.2 Transfer of the Binary Image to RAM

Fig. 4.16 shows the test set-up and connections used to transfer the new compiled firmware image into the hardware.

The following steps discuss the image transfer procedure and their respective screen-shots:

1. Connect an ethernet cable between the Host-PC and one of the LAN ports (any one of the four LAN ports but not the WAN port) of the targeted

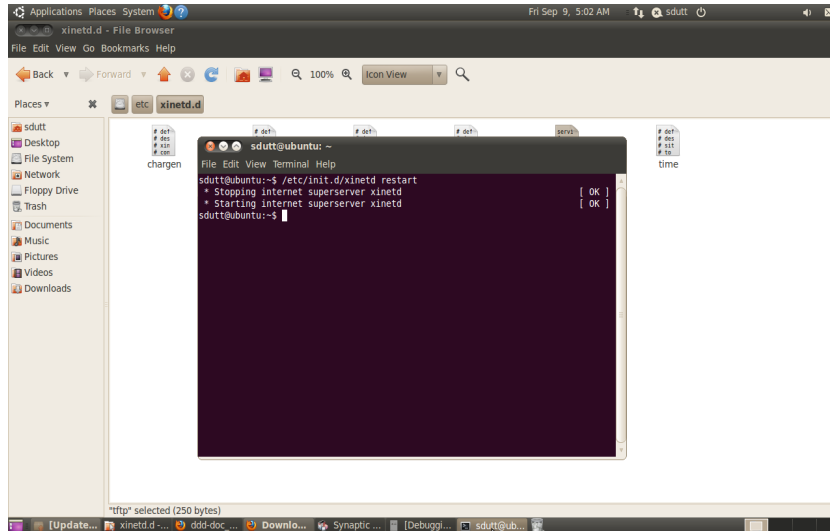


Figure 4.11: TFTP server start screen-shot

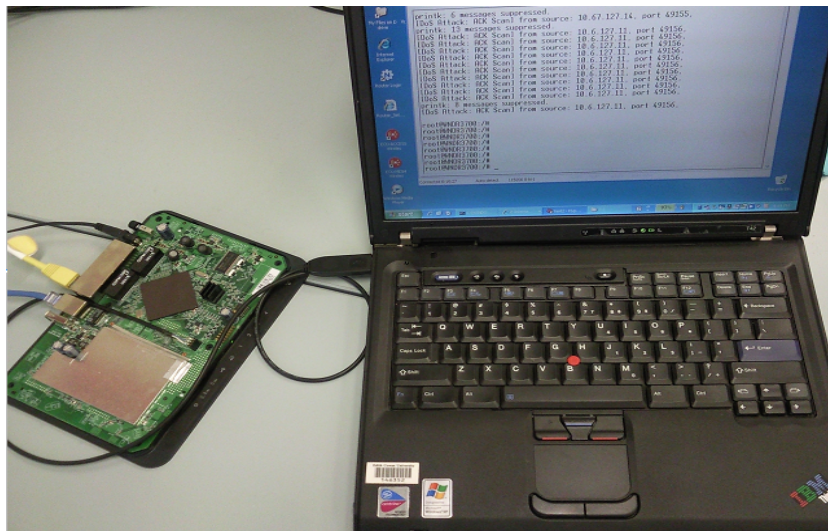
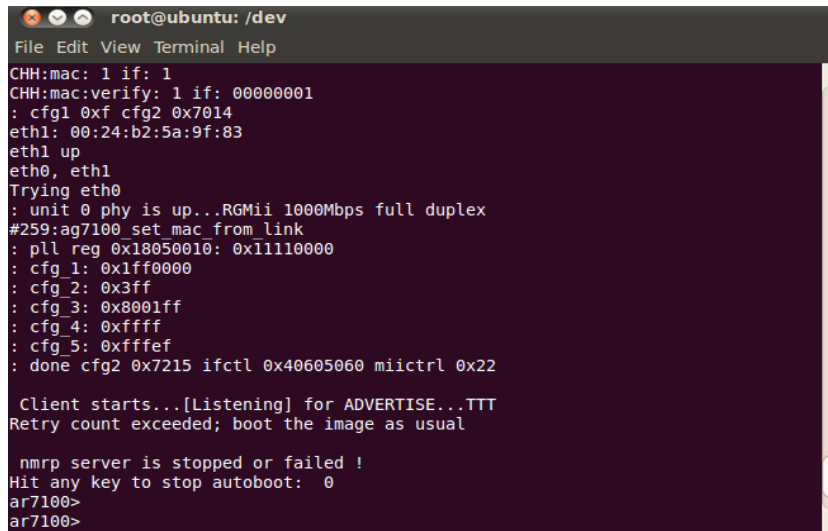


Figure 4.12: Test set-up and connections



```
root@ubuntu: /dev
File Edit View Terminal Help
CHH:mac: 1 if: 1
CHH:mac:verify: 1 if: 00000001
: cfg1 0xf cfg2 0x7014
eth1: 00:24:b2:5a:9f:83
eth1 up
eth0, eth1
Trying eth0
: unit 0 phy is up...RGMii 1000Mbps full duplex
#259:ag7100 set_mac from link
: pll reg 0x18050010: 0x11110000
: cfg_1: 0x1ff0000
: cfg_2: 0x3ff
: cfg_3: 0x8001ff
: cfg_4: 0xfffff
: cfg_5: 0xffffef
: done cfg2 0x7215 ifctl 0x40605060 miictrl 0x22

Client starts...[Listening] for ADVERTISE...TTT
Retry count exceeded; boot the image as usual

nmrp server is stopped or failed !
Hit any key to stop autoboot: 0
ar7100>
ar7100>
```

Figure 4.13: Hardware boot-up screen-shot

WNDR3700 hardware.

2. Connect the custom build serial cable to the header “J1” on the WNDR3700 and set the local hyperterminal session (e.g. minicom) for 115200 bps 8N1, no software flow control and no hardware flow control.
3. Set the Host-PC ethernet port to use a static IP address of 192.168.1.2 and netmask 255.255.255.0.
4. At the power on reset (POR), the hardware boots up with the uboot bootloader. A further boot-up of the kernel and the application from flash memory can be aborted by hitting the 'EnterKey' once the bootloader finishes. Fig. 4.13 shows a screen-shot of the hyper-terminal window once the hardware boots-up and the main application execution is aborted.
5. Fig. 4.13 shows that the eth0 is up and running in full duplex mode after

branch and the compilation process. The dependent packages can be downloaded by running following commands from a CLI or with the help of Synaptic Package Manager (SPM) in Ubuntu 10.04.

```
sudo apt-get install subversion g++ ncurses-term zlib1g-dev gawk flex
```

```
sudo apt-get install patch openssh-server minicom tftp tftpd gettext
```

```
sudo apt-get install libncurses5 libncurses5-dev
```

The source code of the backfire branch can be downloaded as follows:

```
cd ~
```

```
mkdir openwrt
```

```
svn co svn : //svn.openwrt.org/openwrt/branches/backfire
```

Once the backfire branch has been downloaded, there can be found four directories in the base:

```
tools
```

```
toolchain
```

```
package
```

target

The directories */tools* and */toolchain* refer to common tools which are used to build the firmware image, the compiler, and the C library. The result of compilation process is three new directories: i) */builddir/host*, which is a temporary directory for building the target independent tools, ii) */builddir/toolchain – arch*, which is used for building the toolchain for a specific architecture, and iii) */stagingdir/toolchain – arch*, where the resulting toolchain is installed [55]. There is no need to do anything with the toolchain directory unless we intend to add a new version of the components.

The */package* directory contains all the dependent packages. In OpenWRT distribution, almost everything is a package. A software package can be added to the firmware to provide new features, or removed to save space. The */target* directory refers to the embedded platform and contains items which are specific to an embedded platform. The */target/linux* directory, which is broken down by platform, contains patches to the kernel and the profile config. There is also a */target/imagebuilder* directory, which describes how to package a firmware for a specific platform.

4.4.2 Build OpenWRT for the WNDR3700

OpenWRT takes a non-traditional approach to build a firmware, and download and apply patches. It compiles everything from scratch, including the cross compiler. OpenWRT does not contain any executable or the source code. It is an automated system for downloading the source code, patching them to work with the given platform, and compiling them correctly for that platform [55].

OpenWRT distribution can be downloaded as discussed in Section 4.4.1.

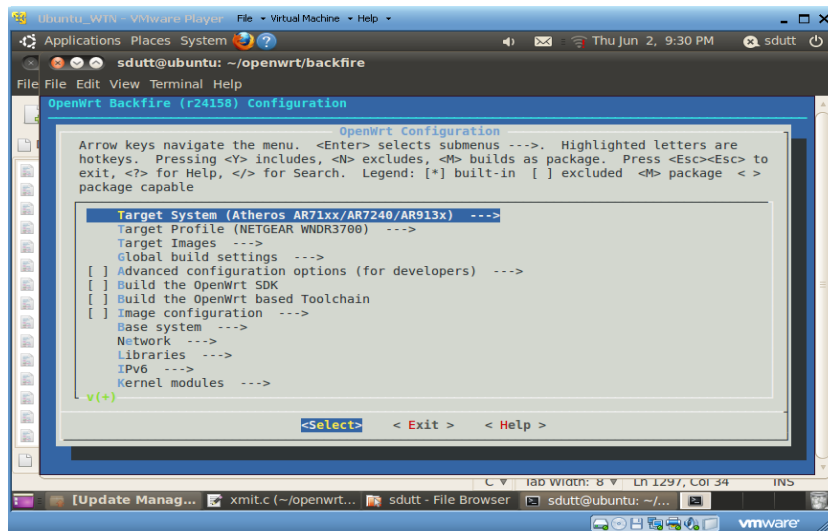


Figure 4.17: Make menuconfig screen-shot

Once it is done, then run the buildroot of OpenWRT i.e. '*make menuconfig*'. It will validate the Host-PC for the required dependencies and tools. A screen-shot of '*make menuconfig*' is shown in Fig. 4.17. While '*make menuconfig*' is running, a configuration window as shown in Fig. 4.17 will prompt the user to select the relevant configurations. For the WNDR3700 hardware, the target system is "Atheros AR71xx/AR7240/AR913x", and the target profile is "NETGEAR WNDR3700". The configuration should be saved before exiting from the selection window.

The '*make menuconfig*' will set the appropriate environment and configuration to build the intended target i.e. the WNDR3700. To build the firmware, type '*make*' and '*return*'. It takes approximately half an hour to build the image from scratch. The separate binaries are built for RAM and flash memory. The binary images are stored into the */openwrt/backfire/bin/ar71xx* directory. The new binary image can be programmed to the RAM by following steps presented in Section 4.3.2.2.

```
static const struct net_device_ops ieee80211_dataif_ops =
{
.ndo_open                = ieee80211_open ,
.ndo_stop                = ieee80211_stop ,
.ndo_uninit              = ieee80211_teardown_sdata ,
.ndo_start_xmit          = ieee80211_subif_start_xmit ,
.ndo_set_multicast_list  = ieee80211_set_multicast_list ,
.ndo_change_mtu          = ieee80211_change_mtu ,
.ndo_set_mac_address     = ieee80211_change_mac ,
.ndo_select_queue        = ieee80211_netdev_select_queue
};
```

Figure 4.18: An example of `net_device` function pointer initialisation

4.5 Software Architecture

Linux kernel is an open source monolithic operating system [58], and has modular architecture. The hardware drivers are compiled as modules and loaded when a hardware is detected. The architecture of Linux kernel provides a hardware independent and uniform interface. This uniform interface is provided through the concept of network devices at TCP/IP layer i.e. 'net_device'. The TCP/IP layer implements a function-pointer based mechanism, which provides a generic interface with the hardware abstraction layer (HAL). An unique 'net_device' object is created for each available physical interface e.g. eth0, eth1, wlan0, wlan1. When a 'net_device' is requested, the function pointers are initialised to the relevant functions of the soft-MAC. In the case of WLAN, the soft-MAC framework is known as 'mac80211'. For example, for wlan0, the application opens a 'net_device' with 'ieee80211' functions, which are defined in 'mac80211'. Fig. 4.18 shows the initialisation of function pointers when a wlan0 interface is opened.

A top level software architecture representing various layers in OpenWRT is shown in Fig. 4.19. WLAN device drivers are divided into two modules: i)

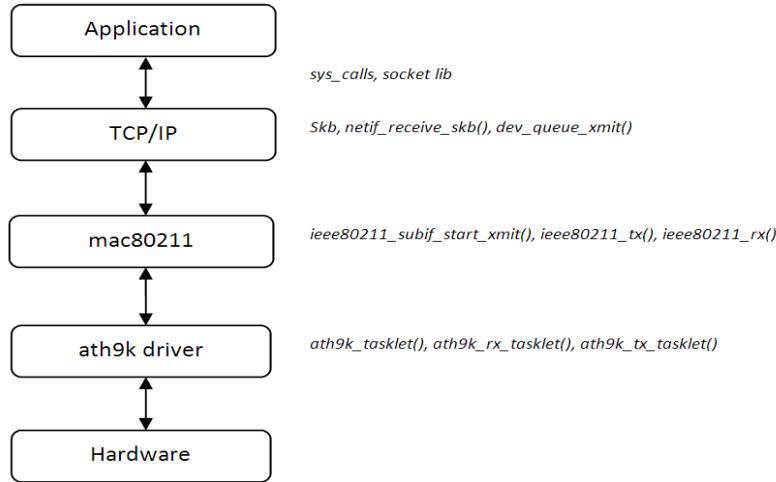


Figure 4.19: Top-level software architecture of OpenWRT

hardware dependent module, and ii) protocol module (soft-MAC). The hardware dependent modules are different for each vendor and based on capabilities. The soft-MAC provides an interface for most of the MAC functionality defined in the IEEE 802.11 protocol [58]. The WNDR3700 router uses a hardware dependent driver by Atheros, the 'ath9k'. The 'mac80211' uses the low-level functions of the 'ath9k' device driver to transmit and receive the data packets. The functional flow between various kernel layers during the data packet transmission and reception is explained in the following two subsections.

4.5.1 Packet Transmission

The packet arrival at the TCP/IP layer is handled by the function `dev_queue_xmit()`, which in turn calls the function `dev_hard_start_xmit()`. Both of these functions are defined in "net" driver of the kernel. This function in turn calls a function pointer 'ndo_start_xmit', which is mapped to function `ieee80211_subif_start_xmit()` defined in `mac80211`. The `ieee80211_subif_start_xmit()` calls the `ieee80211_tx()`

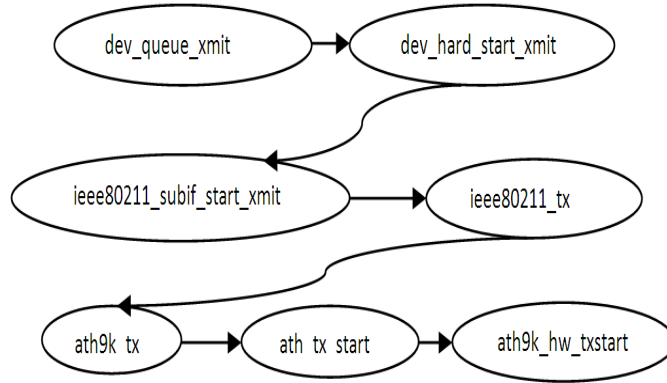


Figure 4.20: Packet transmission functional path in OpenWRT

function, which finally calls `ath9k_tx()` from the 'ath9k' device driver. The sequence of various function calls during packet transmission is shown in Fig. 4.20.

4.5.2 Packet Reception

A new packet arrival at the PHY is notified by an interrupt. This interrupt is handled by an interrupt service routine (ISR), `ath_isr()`, defined in the 'ath9k' driver. The received packet is handled by `ath_rx_tasklet()`, which retrieves the packet bytes from the Rx queue buffer in hardware. The function `ath_rx_tasklet()` also maps the packet in the socket buffer structure, 'skb', and passes this 'skb' to the function `ieee80211_rx()`. The `ieee80211_rx()` function is defined in 'mac80211'. The function `ieee80211_rx()` passes this 'skb' to the upper network layer by calling the function `netif_receive_skb()`. The network layer passes the received 'skb' to the upper layer for further delivery to the application layer. It may also send the 'skb' to `dev_queue_xmit()` for further routing

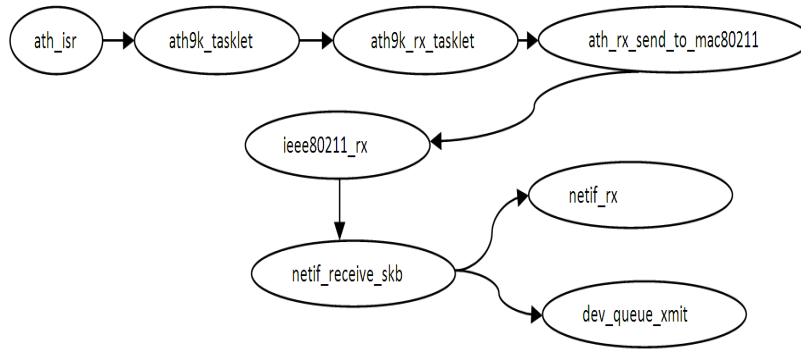


Figure 4.21: Packet reception functional path in OpenWRT

of the received packets to another interface e.g. WAN port. The sequence of various function calls during packet reception is shown in Fig. 4.21.

4.6 Firmware Development

To implement the TI-MAC polling cycle presented in Chapter 3, two state machines are developed in “C” programming language. First state machine runs in the AP that generates the polling cycles for four associated stations, and the second state machine runs in each remote station. The state machine running in the remote station waits for the polling cycles, and starts the packet transmission once received. The source code of both state machines is presented in appendix A.

4.7 Summary

This chapter discusses internal hardware details of Netgear WNDR3700

router, and software architecture of OpenWRT. The WNDR3700 router is built from the Atheros's advanced and high performance network processor, the AR7161, and the AR9280 802.11a/b/g/n WLAN chip. This hardware is embedded with a firmware image built from OpenWRT software stack. OpenWRT and WNDR3700 provide an excellent solution for the IEEE 802.11 based MAC research. This chapter also presents a framework to establish a Hardware-PC interface by using an USB-to-Serial cable. This interface is used to load the compiled binary image into the hardware, and for debugging purposes. A procedure to build a custom USB-to-Serial cable has been also explained in Section 4.3.1. This is followed by the steps for transferring a new compiled binary image into the hardware. At the end, Linux kernel architecture and the low-level functional flow in the path of packet transmission and reception is also presented here.

Chapter 5

Test Set-up and Characterisation

For performance analysis and characterisation, a testbed, as shown in Fig. 5.1, was established. The TI-MAC scheduler was implemented in the Atheros kernel driver using programming language “C” and Linux real-time operating system application programming interfaces (APIs). The complete Linux stack was recompiled, and the image was reprogrammed in the hardware as per the steps discussed in Chapter 4.

This research work was targeted for real-time traffic applications such as symmetric video telephony, which have $\text{IPSR}(\mu_i)$ equal to 0.5. As per the example presented in Section 3.3, a polling cycle of 80 *mSec* was assumed because it gives sufficient room for propagation delays and codec latencies. This work assumed that video and voice data were multiplexed together as a single stream above the transport layer. There were four stations in our testbed with two way traffic and the stations were associated with one access point. This work also assumed that all the four stations had the same multimedia application for

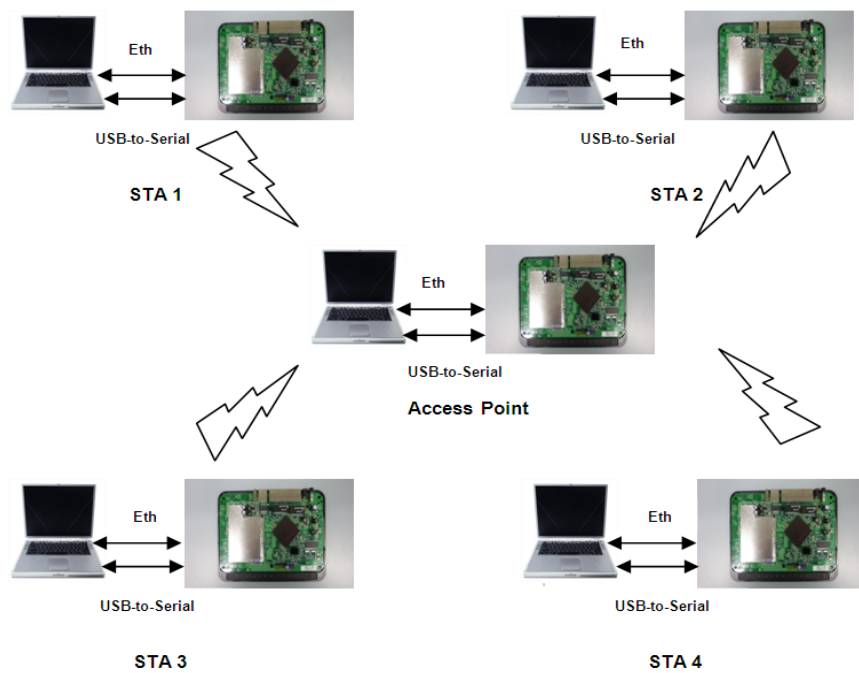


Figure 5.1: TI-MAC evaluation test-bed

video telephony and hence, they had the same bandwidth requirement. Two scenarios were tested for image resolutions of 128×96 and 352×288 at PHY data rate of 2 Mbps and 9 Mbps, respectively. The ITU-T H.322 standard for visual telephony system [59] defines various video resolutions for QoS, and the selected two resolutions are suitable for multiple wireless clients in a wideband or broadband network. A typical video telephony scenario contains two real-time streams in each direction i.e. one voice and one video. The parameters used in test scenarios I & II are summarised in Tables 5.1 & 5.2, respectively. The common parameters are summarised in Table 5.3.

Moreover, this work assumed an optimised packet length control from application layer rather than fragmentation at the MAC layer. An optimised packet length may be decided by noting that a packet length, if too short, causes congestion and adds more overheads due to headers. At the same time, a packet length which is too large, increases the packet loss ratio [60]. Keeping these facts in mind, we used packet sizes of 790 and 1195 bytes.

For the experiments, a constant bit rate (CBR) model had been considered for video IP telephony traffic [61]. The variable bit rate (VBR), which is a known problem with real-time data, is not an issue for video IP telephony. Here, the user is static or has limited motion, and due to this the video data rates for successive group-of-pictures (GOPs) remain nearly constant. Moreover, the performance of TI-MAC was evaluated at full bandwidth limit. In case of less real-time traffic than the allocated bandwidth, the remaining bandwidth can be used for best-effort traffic, e.g., sharing data or music files. To meet the strict QoS requirement, the allocated bandwidth can not be shared among other users.

5.1 Test Scenario

This section presents two test scenarios and their respective parameters con-

sidered, to evaluate the performance of TI-MAC and the IEEE 802.11e EDCA.

5.1.1 Test Scenario I

The image resolution of 128×96 requires a bandwidth of 210 kbps in each direction with H.264 video and G.711 voice, and it includes overheads from RTP, UDP, IP, IEEE 802.11 headers. A payload size of 720 bytes was generated in every 30 ms, which contained 240 bytes of voice and 480 bytes of video. Moreover, 70 bytes were appended to compensate for various header overheads, resulting in a packet size of 790 bytes.

Table 5.1: Test Scenario I Parameters

Parameter	Value
Voice Packet	240 Bytes
Video Packet	480 Bytes
RTP,UDP,IP,802.11	12,8,20,30 Bytes
Voice Rate	64 kbps
Video Rate	128 kbps
Overheads	18 kbps
PHY Rate	2 Mbps

5.1.2 Test Scenario II

The image resolution of 352×288 requires a bandwidth of 956 kbps in each direction with H.264 video and G.711 voice. A payload size of 1125 bytes was generated in every 10 ms and 240 bytes of voice were multiplexed in one of the three consecutive packets.

Table 5.2: Test Scenario II Parameters

Parameter	Value
Voice Packet	240 Bytes
Video Packet	1125 Bytes
RTP,UDP,IP,802.11	12,8,20,30 Bytes
Voice Rate	64 kbps
Video Rate	836 kbps
Overheads	56 kbps
PHY Rate	9 Mbps

Table 5.3: Common Parameters

Parameter	Value
T	80 <i>ms</i>
β	20 <i>ms</i>
η	≈ 3 <i>ms</i>
μ_i	0.5
Packet Type	RTP/UDP
PLCP Preamble + Header	192 μSec
CW_{min}, CW_{max} - TI-MAC	0, 0
CW_{min}, CW_{max} - 802.11e EDCA	15, 31
SIFS	10 μSec
Slot Time	20 μSec

In the experiments, this work observed throughput for upstream and downstream traffic and packet delay variation. This work also intended to see the fairness of TXOP allocation of the scheduler to all the stations in a real-time embedded hardware where multiple tasks were running.

Figs. 5.2, 5.3, 5.4 and 5.5 show the upstream and downstream data rates between the access point and the four stations. These charts also display the MAC addresses (Dest MAC - Src MAC) of the access point and stations used in our experiments. For example in Fig. 5.2, 00:24:B2:57:CA:5E is destination MAC address and 00:24:B2:5A:9F:82 is source MAC address. Figs. 5.6 and 5.7 show the measured IPDV (jitter) with TI-MAC and 802.11e EDCA, respectively.

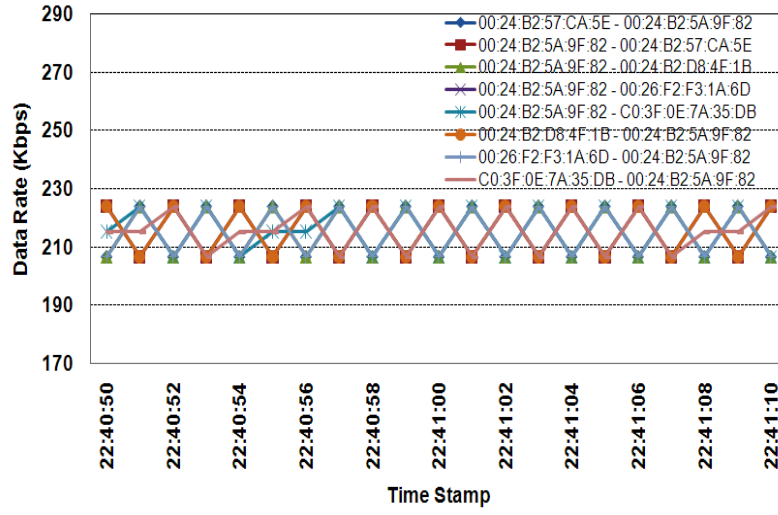


Figure 5.2: Downstream and upstream data rates with TI-MAC at 2 Mbps

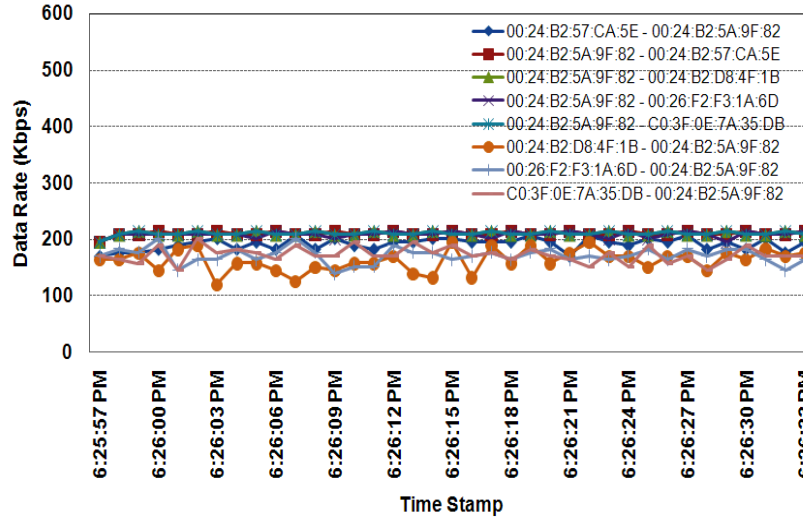


Figure 5.3: Downstream and upstream data rates with 802.11e EDCA at 2 Mbps

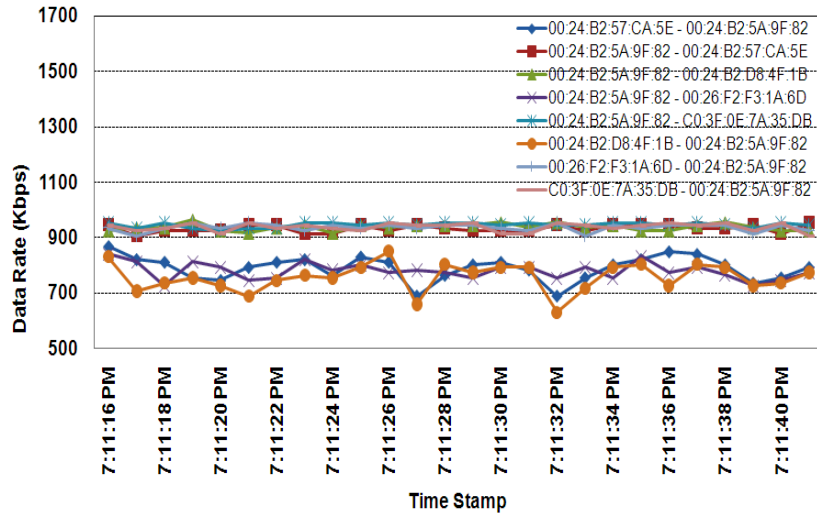


Figure 5.4: Downstream and upstream data rates with TI-MAC at 9 Mbps

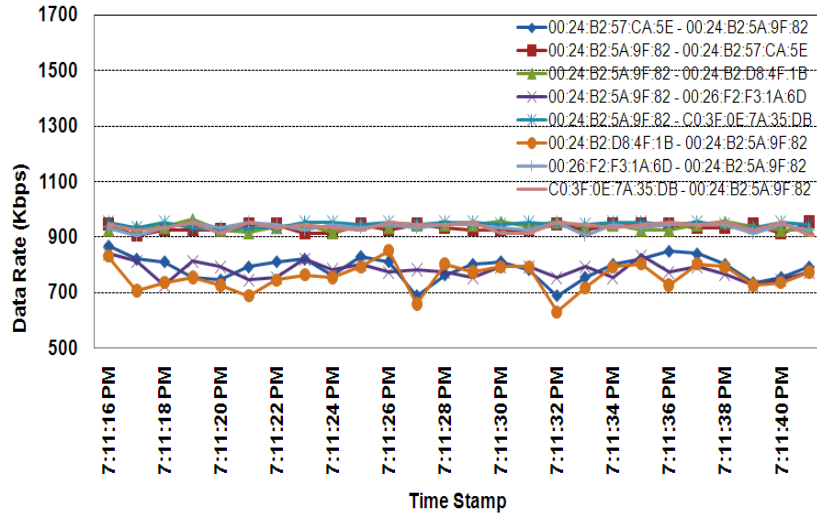


Figure 5.5: Downstream and upstream data rates with 802.11e EDCA at 9 Mbps

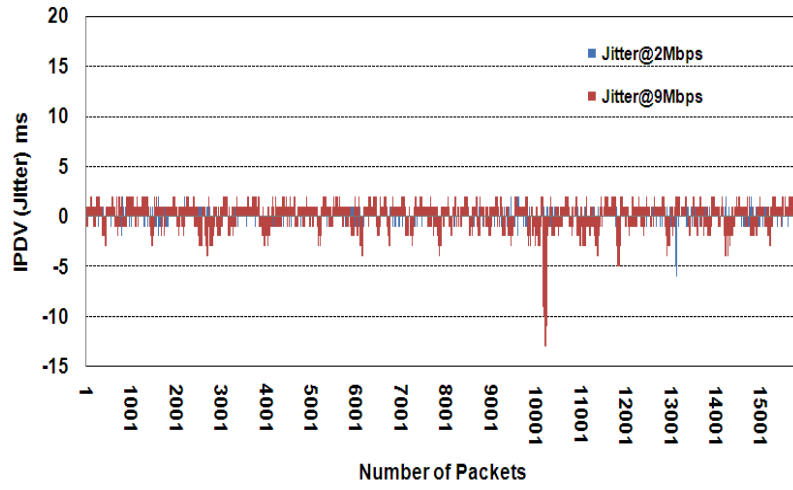


Figure 5.6: IPDV (jitter) with TI-MAC at 2 and 9Mbps

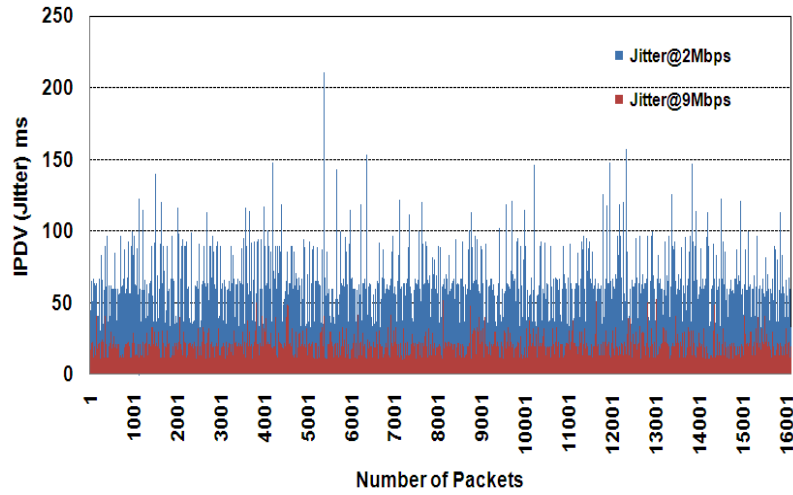


Figure 5.7: IPDV (jitter) with 802.11e EDCA at 2 and 9Mbps

5.2 Result Analysis

This section draws an analysis over the throughput and jitter results achieved in the aforementioned test scenarios. The analysis shows that the TI-MAC enabled STAs deliver better performance than the IEEE 802.11e EDCA.

5.2.1 Test Scenario I

At 2 Mbps PHY bit rate with TI-MAC, the upstream and downstream for all the STAs have an average data rate of 214 kbps and 215 kbps (Fig. 5.2), which gives an IPSR of approximately 0.5. The standard deviation is only 8 kbps, which is 3.7% of the throughput, indicating that bi-directional streams are very stable. This bandwidth is enough to make a video call with an image resolution of 128×96 which requires a bandwidth of 210 kbps. Moreover, jitter is also less than 6 ms (Fig. 5.6).

At the same time, in the 802.11e EDCA, the upstream is served well, but the downstream traffic does not receive enough bandwidth. The upstream and downstream have an average data rate of 207 kbps and 160 kbps (Fig. 5.3), respectively, which gives an IPSR of approximately 0.6. This bandwidth is not enough to make a video display with an image resolution of 128×96 which requires a bandwidth of 210 kbps in each direction. The standard deviation for downstream traffic is 15.8 kbps which is approximately 10% of the throughput. Moreover, 24% of the packets in the downstream direction are dropped at the access point. The big drop in the downstream direction is due to data clogging in access point as discussed in Section 2.1.4.2. From the jitter graph it is evident that packet arrival is very non-deterministic and jitter frequently exceeding the threshold limit of 50 ms (Fig. 5.7).

Table 5.4: Performance of TI-MAC v 802.11e EDCA-Test Scenario I						
MAC	US*	DS*	IPSR	Packet Loss (US, DS)	Max. Jitter	QoS Criteria Satisfied
TI-MAC	214	215	0.5	0%, 0%	6	Yes
802.11e EDCA	207	160	0.6	1.4%, 24%	211	No

* - US - UpStream, DS - DownStream

5.2.2 Test Scenario II

At 9 Mbps PHY rate with TI-MAC, the upstream and downstream traffic have an average data rates of 942 kbps and 945 kbps (Fig. 5.4), respectively, for each station, which gives an IPSR of approximately 0.5. The standard deviation is 25 kbps, which is only 2.6% of the throughput, indicating that bi-directional data rates are very stable. This bandwidth is very close to what is required to make a video call with an image resolution of 352×288 , which requires a bandwidth of 956 kbps. Only 1.5% of the packets are dropped in each direction. Moreover, jitter is less than 13 ms (Fig. 5.6).

Again with the 802.11e MAC, the upstream traffic is served well, but the downstream traffic does not receive enough bandwidth. The upstream and downstream have an average data rate of 900 kbps and 770 kbps (Fig. 5.5), respectively, which gives an IPSR of approximately 0.6. The standard deviation for the downstream traffic is 50 kbps, which is approximately 6.5% of the throughput. This downstream bandwidth is not enough to make a video call with an image resolution of 352×288 . Moreover, 5.9% of packets are dropped in upstream and 19.5% of the packets are dropped in downstream. Packet arrival is also non-deterministic and jitter most of the time remains close to 50 ms (Fig. 5.7).

Table 5.5: Performance of TI-MAC v 802.11e EDCA-Test Scenario II						
MAC	US*Rate (kbps)	DS* Rate (kbps)	IPSR	Packet Loss (US DS)	Max. Jitter	QoS Criteria Satisfied
TI-MAC	942	945	0.5	1.5%,1.1%	13	Yes
802.11e EDCA	900	770	0.6	5.9%, 19.5%	53	No

* - US - UpStream, DS - DownStream

Thus, from the above performance summary depicted in Tables 5.4 and 5.5, it is clearly evident that the IEEE 802.11e EDCA failed to meet QoS criteria. The IEEE 802.11e EDCA could not provide enough bandwidth to any station to make video call at the required resolution. TI-MAC on the other hand provides a centralised resource management and delivers substantially improved performance. Moreover, the hand-shake mechanism in TI-MAC nullifies the possibility of packet collisions.

5.3 Summary

This chapter presents the testbed set-up used for performance evaluation of TI-MAC, and its comparison against the IEEE 802.11e EDCA. We have targeted two video IP telephony scenarios at different image resolutions. Section 5.1 discusses these two test scenarios and their respective MAC and PHY parameters. This chapter also depicts the graphs of throughput and jitter measurements at 2 Mbps and 9 Mbps for TI-MAC and the IEEE802.11e EDCA. At the end, this chapter presents a result analysis of the targeted test scenarios, and shows that TI-MAC outperformed EDCA by meeting a strict QoS criteria.

Chapter 6

Conclusion and Future Works

This chapter focuses on the contributions of this work: i) development of an efficient MAC protocol for VVoIP applications, and ii) development of an embedded platform for MAC protocol validation. This is followed by a discussion of the test scenarios considered for performance evaluation, results, and suggestions for future work.

6.1 Research Contributions

In this thesis, a round-robin scheduler optimised for symmetric real-time traffic from applications such as voice and/or video IP telephony in WLANs has been presented. This work has identified the need for a new wireless MAC protocol to support real-time multimedia applications and introduced a new attribute for IP traffic symmetry pattern, named IP traffic symmetry ratio (IPSR). The proposed traffic intelligent MAC (TI-MAC) scheduler allocates each station a TXOP as per the QoS agreement and TXOP utilisation depends on IPSR of the STA. The polling cycle is optimised after taking real-time attributes into con-

siderations, e.g. IPTD, IPSR, IPDV and latencies of the system. The TI-MAC makes good synchronisation between the AP and STAs through an efficient hand-shake mechanism. This scheduler has been tested for symmetric traffic, and the performance has been benchmarked against the IEEE 802.11e EDCA. Detailed functional specification and admission control policy of TI-MAC have been discussed in Chapter 3.

This thesis also presents a hardware testbed for the IEEE 802.11 WLAN researches. It is widely accepted that, for implementation and commercialisation of new ideas in WLAN, a proof of concept conducted in a hardware platform is crucial, if not mandatory. There are two well defined hardware platforms available for WLAN research: i) WARPnet, and ii) CALRADIO1. These hardware provide a limited framework for a new MAC protocol implementation, and related cost of development is also very high. This work has presented a cost effective hardware platform that can be used for implementing and testing of the MAC protocols in a practical system. This work has used an off-the-shelf Netgear router as the practical system and has implemented TI-MAC as a test case. Chapter 4 has presented various steps to build an evaluation platform with Netgear router. The hardware platform has generated results that are consistent with other similar studies (e.g. using NS2) and as such, the effectiveness of the hardware platform has been demonstrated in this work.

6.2 Test Scenarios and Results

In video IP telephony, video stream occupies most of the bandwidth. Thus, we have considered video and voice quality as one of the QoS criterias and defined two test scenarios at different image resolutions: i) 128×96 , and ii) 352×288 . The two image resolutions can provide a good quality video in a handheld device

and laptop. The test scenarios have also carefully considered various overheads due to RTP, UDP, IP and MAC headers. Chapter 5 has tabulated various MAC and PHY parameters used for the packet transmission. We have considered jitter as another parameter for QoS criteria. This work has collected the results of throughput and jitter at 2 Mbps and 9 Mbps. Chapter 5 has also presented the graphs of throughput and jitter for both TI-MAC and the IEEE 802.11e EDCA.

Tables 5.4 and 5.5 present a performance comparison between TI-MAC and the IEEE 802.11e EDCA. The TI-MAC outperforms the IEEE 802.11e EDCA by supporting more number of stations and meets strict QoS criteria. Whereas, the IEEE 802.11e EDCA does not provide sufficient bandwidth to even a single station. The TI-MAC treats both upstream and downstream traffic fairly with desirable IPSR of 0.5, and IPDV of less than 13 ms. The result summary is presented below.

At 2 Mbps, TI-MAC delivers throughput of 214 kbps and 215 kbps in the upstream and downstream, respectively. The maximum jitter is less than 6 ms and no packet loss. Whereas, EDCA delivers throughput of 207 kbps and 160 kbps in the upstream and downstream, respectively. The maximum jitter goes upto 211 ms, and packet loss of 1.4 % and 24 % occurs in the upstream and downstream, respectively.

At 9 Mbps, TI-MAC delivers throughput of 942 kbps and 945 kbps in the upstream and downstream, respectively. The maximum jitter is less than 13 ms, and packet loss of 1.5 % and 1.1 % occurs in the upstream and downstream, respectively. Whereas, EDCA delivers throughput of 900 kbps and 770 kbps in the upstream and downstream, respectively. The maximum jitter goes upto 53 ms, and packet loss of 5.9 % and 19.5 % occurs in upstream and downstream

respectively.

6.3 Suggestions for Future Work

In this work, TI-MAC has been validated for VVoIP traffic that has symmetric data in both the directions. As a future work, the TI-MAC may be evaluated for IPTV traffic which has only downstream streaming and occasionally upstream data. Similarly, WLAN surveillance cameras which continuously upstream the video can also be a targeted application for TI-MAC as a future work. TI-MAC has been evaluated with a test traffic generated by a software application. Future work could use live Skype video call to generate the video and voice traffic.

TI-MAC provides an excellent mechanism to control the IPSR in run-time. The traffic symmetry requirement of a client may change time-to-time. A STA can exchange the new IPSR information with the AP to tune the values for the upstream and downstream TXOPs. In the current work, we have assumed a fixed IPSR of 0.5, which is targeted for video IP telephony application. The future work could define a packet format for IPSR exchange, and use this control packet to modify the IPSR as per change in the traffic symmetry. A test application capable of generating traffic at various IPSR could be developed to validate the run-time shifting of IPSR.

This work has considered 4 STAs and 1 AP in the testbed for performance evaluation of TI-MAC. The STAs and AP are the Netgear WNDR3700 routers. In a BSS, there can be any number of STAs running with TI-MAC without a limitation. Thus, to prove the scalability of Netgear hardware and TI-MAC, a testbed with more numbers of STAs may be demonstrated as a future work. Moreover, we have evaluated TI-MAC in an outdoor environment which had a

line of sight link between STAs and AP, and no obstructions. As a future work, TI-MAC could also be evaluated in an open office and closed office environment, which do not provide a line of sight link and have multiple obstructions.

APPENDIX A

A.1 Base STA State Machine Code

```
void ath_tmac_baseSTA_work(struct work_struct *work)
{
    struct ath_softc *sc = container_of(work, struct ath_softc, wtn_work);
    struct ieee80211_hw *hw = sc->hw;
    volatile unsigned long ds_delay;
    struct ath_tx_control txctl;
    int state = UP_STREAM;
    struct ath_hw *ah = sc->sc_ah;
    struct ath_common *common = ath9k_hw_common(ah);
    struct sk_buff *skb_1, *skb_2, *skb_3, *skb_4;
    struct sk_buff *skb_data_1, *skb_data_2, *skb_data_3, *skb_data_4;
    unsigned char *data;
    struct ieee80211_hdr qos_hdr;
    unsigned char dest_mac_1[] = {0x00, 0x24, 0xb2, 0xd8, 0x4f, 0x1b};
    unsigned char dest_mac_2[] = {0xC0, 0x3F, 0x0E, 0x7A, 0x35, 0xDB};
    unsigned char dest_mac_3[] = {0x00, 0x24, 0xb2, 0x57, 0xCA, 0x5E};
    unsigned char dest_mac_4[] = {0x00, 0x26, 0xF2, 0xF3, 0x1A, 0x6D};
    unsigned char src_mac[] = {0x00, 0x24, 0xb2, 0x5a, 0x9f, 0x82};
    int band = hw->conf.channel->band;
    struct ieee80211_supported_band *sband = &sc->sbands[band];
    __le16 fc;
    __le32 *qos_control;
    struct ieee80211_tx_info *tx_info_1;
    struct ieee80211_tx_info *tx_info_2;
    struct ieee80211_tx_info *tx_info_3;
    struct ieee80211_tx_info *tx_info_4;
    struct ieee80211_tx_info *tx_info_data_1;
    struct ieee80211_tx_info *tx_info_data_2;
    struct ieee80211_tx_info *tx_info_data_3;
    struct ieee80211_tx_info *tx_info_data_4;
    struct ieee80211_sta sta;
    volatile unsigned long current_jiffies, txop=0;
    int wtn_err=0, i=0, temp, seq_no=0;
    int txop_req=0, txop_remain=0;
    skb_1 = alloc_skb(100, GFP_KERNEL);
    if (!skb_1)
    {
        printk("\nSKB ALLOCATION ERROR IN WTN TASK\n");
        return;
    }
}
```

```

skb_2 = alloc_skb(100, GFP_KERNEL);
if (!skb_2)
{
    printk("\nSKB ALLOCATION ERROR IN WIN TASK\n");
    return;
}

skb_3 = alloc_skb(100, GFP_KERNEL);
if (!skb_3)
{
    printk("\nSKB ALLOCATION ERROR IN WIN TASK\n");
    return;
}

skb_4 = alloc_skb(100, GFP_KERNEL);
if (!skb_4)
{
    printk("\nSKB ALLOCATION ERROR IN WIN TASK\n");
    return;
}

skb_data_1 = alloc_skb(1800, GFP_KERNEL);
if (!skb_data_1)
{
    printk("\nSKB ALLOCATION ERROR IN WIN TASK\n");
    return;
}

skb_data_2 = alloc_skb(1800, GFP_KERNEL);
if (!skb_data_2)
{
    printk("\nSKB ALLOCATION ERROR IN WIN TASK\n");
    return;
}

skb_data_3 = alloc_skb(1800, GFP_KERNEL);
if (!skb_data_3)
{
    printk("\nSKB ALLOCATION ERROR IN WIN TASK\n");
    return;
}

skb_data_4 = alloc_skb(1800, GFP_KERNEL);
if (!skb_data_4)
{
    printk("\nSKB ALLOCATION ERROR IN WIN TASK\n");

```

```

    return;
}

tx_info_1 = IEEE80211_SKB_CB(skb_1);
tx_info_2 = IEEE80211_SKB_CB(skb_2);
tx_info_3 = IEEE80211_SKB_CB(skb_3);
tx_info_4 = IEEE80211_SKB_CB(skb_4);

tx_info_data_1 = IEEE80211_SKB_CB(skb_data_1);
tx_info_data_2 = IEEE80211_SKB_CB(skb_data_2);
tx_info_data_3 = IEEE80211_SKB_CB(skb_data_3);
tx_info_data_4 = IEEE80211_SKB_CB(skb_data_4);

memset(tx_info_1, 0, sizeof(*tx_info_1));
memset(tx_info_2, 0, sizeof(*tx_info_2));
memset(tx_info_3, 0, sizeof(*tx_info_3));
memset(tx_info_4, 0, sizeof(*tx_info_4));
memset(tx_info_data_1, 0, sizeof(*tx_info_data_1));
memset(tx_info_data_2, 0, sizeof(*tx_info_data_2));
memset(tx_info_data_3, 0, sizeof(*tx_info_data_3));
memset(tx_info_data_4, 0, sizeof(*tx_info_data_4));
memset(&qos_hdr, 0, sizeof(qos_hdr));
memset(&txctl, 0, sizeof(txctl));
printk("\nattached node in wtn=%x", sc->an);

tx_info_1->flags=0x14;
tx_info_1->band=band;
tx_info_1->control.rates[0].idx = 1;
tx_info_1->control.rates[0].count = 4;
tx_info_2->flags=0x14;
tx_info_2->band=band;
tx_info_2->control.rates[0].idx = 1;
tx_info_2->control.rates[0].count = 4;
tx_info_3->flags=0x14;
tx_info_3->band=band;
tx_info_3->control.rates[0].idx = 1;
tx_info_3->control.rates[0].count = 4;
tx_info_4->flags=0x14;
tx_info_4->band=band;
tx_info_4->control.rates[0].idx = 1;
tx_info_4->control.rates[0].count = 4;
tx_info_data_1->flags=0x14;
tx_info_data_1->band=band;
tx_info_data_1->control.rates[0].idx = 5;
tx_info_data_1->control.rates[0].count = RETRY_1;
tx_info_data_2->flags=0x14;

```

```

tx_info_data_2->band=band;
tx_info_data_2->control.rates[0].idx = 5;
tx_info_data_2->control.rates[0].count = RETRY_1;
tx_info_data_3->flags=0x14;
tx_info_data_3->band=band;
tx_info_data_3->control.rates[0].idx = 5;
tx_info_data_3->control.rates[0].count = RETRY_1;
tx_info_data_4->flags=0x14;
tx_info_data_4->band=band;
tx_info_data_4->control.rates[0].idx = 5;
tx_info_data_4->control.rates[0].count = RETRY_1;
skb_reserve(skb_1, 40);
memset(skb_put(skb_1, 2), 0x99, 2);
skb_reserve(skb_2, 40);
memset(skb_put(skb_2, 2), 0x99, 2);
skb_reserve(skb_3, 40);
memset(skb_put(skb_3, 2), 0x99, 2);
skb_reserve(skb_4, 40);
memset(skb_put(skb_4, 2), 0x99, 2);
skb_reserve(skb_data_1, 100);
memset(skb_put(skb_data_1, MPDU_SIZE), 0xab, MPDU_SIZE);
skb_reserve(skb_data_2, 100);
memset(skb_put(skb_data_2, MPDU_SIZE), 0xcd, MPDU_SIZE);
skb_reserve(skb_data_3, 100);
memset(skb_put(skb_data_3, MPDU_SIZE), 0xef, MPDU_SIZE);
skb_reserve(skb_data_4, 100);
memset(skb_put(skb_data_4, MPDU_SIZE), 0x55, MPDU_SIZE);
fc = cpu_to_le16(IEEE80211_FTYPE_DATA | IEEE80211_STYPE_QOS_DATA);

qos_hdr.frame_control = fc;
qos_hdr.duration_id = cpu_to_le16(0x000);
qos_hdr.seq_ctrl=0; seq_no = 0;
qos_hdr.seq_ctrl |= cpu_to_le16(seq_no);
/* DA BSSID SA */
memcpy(qos_hdr.addr1, dest_mac_1, ETH_ALEN);
memcpy(qos_hdr.addr2, src_mac, ETH_ALEN);
memcpy(qos_hdr.addr3, src_mac, ETH_ALEN);
qos_control = (__le32*) skb_push(skb_1, 4);
memcpy(skb_push(skb_1, 24), &qos_hdr, 24);
*qos_control = 0x01000100;
qos_control = (__le32*) skb_push(skb_data_1, 4);
memcpy(skb_push(skb_data_1, 24), &qos_hdr, 24);
*qos_control = 0x21002100; /* with ACK */
/* DA BSSID SA */
memcpy(qos_hdr.addr1, dest_mac_2, ETH_ALEN);
memcpy(qos_hdr.addr2, src_mac, ETH_ALEN);

```

```

memcpy(qos_hdr.addr3, src_mac, ETH_ALEN);
qos_control = (__le32*) skb_push(skb_2, 4);
memcpy(skb_push(skb_2, 24), &qos_hdr, 24);
*qos_control = 0x01000100;
qos_control = (__le32*) skb_push(skb_data_2, 4);
memcpy(skb_push(skb_data_2, 24), &qos_hdr, 24);
*qos_control = 0x21002100;
/* DA BSSID SA */
memcpy(qos_hdr.addr1, dest_mac_3, ETH_ALEN);
memcpy(qos_hdr.addr2, src_mac, ETH_ALEN);
memcpy(qos_hdr.addr3, src_mac, ETH_ALEN);
qos_control = (__le32*) skb_push(skb_3, 4);
memcpy(skb_push(skb_3, 24), &qos_hdr, 24);
*qos_control = 0x01000100;
qos_control = (__le32*) skb_push(skb_data_3, 4);
memcpy(skb_push(skb_data_3, 24), &qos_hdr, 24);
*qos_control = 0x21002100;
/* DA BSSID SA */
memcpy(qos_hdr.addr1, dest_mac_4, ETH_ALEN);
memcpy(qos_hdr.addr2, src_mac, ETH_ALEN);
memcpy(qos_hdr.addr3, src_mac, ETH_ALEN);
qos_control = (__le32*) skb_push(skb_4, 4);
memcpy(skb_push(skb_4, 24), &qos_hdr, 24);
*qos_control = 0x01000100;
qos_control = (__le32*) skb_push(skb_data_4, 4);
memcpy(skb_push(skb_data_4, 24), &qos_hdr, 24);
*qos_control = 0x21002100;

skb_1->priority = 1;
skb_set_queue_mapping(skb_1, 3);
skb_data_1->priority = 1;
skb_set_queue_mapping(skb_data_1, 3);
skb_2->priority = 1;
skb_set_queue_mapping(skb_2, 3);
skb_data_2->priority = 1;
skb_set_queue_mapping(skb_data_2, 3);
skb_3->priority = 1;
skb_set_queue_mapping(skb_3, 3);
skb_data_3->priority = 1;
skb_set_queue_mapping(skb_data_3, 3);
skb_4->priority = 1;
skb_set_queue_mapping(skb_4, 3);
skb_data_4->priority = 1;
skb_set_queue_mapping(skb_data_4, 3);
txctl.txq = sc->tx.txq_map[skb_get_queue_mapping(skb_1)];
ds_delay = msecs_to_jiffies(5000);

```

```

set_current_state(TASK_UNINTERRUPTIBLE);
schedule_timeout(ds_delay);
ath9k_disable_ps(sc);
sc->wtn_ap_tx_start = false;
/* stop further AP Tx */
sc->wtn_ap_task = current;
txop = msecs_to_jiffies(10);
set_user_nice(current, -20);
while(1)
{
    if(skb_1->data[22] == 0xf0)
        skb_1->data[22] = 0x00; /* Reset Seq Control */

    pkt_txed = false;
    if (ath_wtn_tx_start(hw, skb_1, &txctl) != 0)
    {
        printk("\n TI-MAC POLL ERROR \n");
    }
    while(!pkt_txed);
    skb_1->data[22] += 0x10;
    /* Update seq num */
    sc->wtn_sta_txop_end = false;
    ds_delay = msecs_to_jiffies(10);
    set_current_state(TASK_UNINTERRUPTIBLE);
    schedule_timeout(ds_delay);
    current_jiffies = jiffies;
    while(!sc->wtn_sta_txop_end)
    {
        if(jiffies > current_jiffies)
            sc->wtn_sta_txop_end = true;
    }

    if(skb_2->data[22] == 0xf0)
        skb_2->data[22] = 0x00; /* Reset Seq Control */

    pkt_txed = false;
    if (ath_wtn_tx_start(hw, skb_2, &txctl) != 0)
    {
        printk("\n TI-MAC POLL ERROR \n");
    }
    while(!pkt_txed);
    skb_2->data[22] += 0x10;
    /* Update seq num */
    sc->wtn_sta_txop_end = false;
    ds_delay = msecs_to_jiffies(10);
    set_current_state(TASK_UNINTERRUPTIBLE);

```

```

schedule_timeout(ds_delay);
current_jiffies = jiffies;
while(!sc->wtn_sta_txop_end)
{
    if(jiffies > current_jiffies)
        sc->wtn_sta_txop_end = true;
}

if(skb_3->data[22] == 0xf0)
    skb_3->data[22] = 0x00; /* Reset Seq Control */

pkt_txed = false;
if (ath_wtn_tx_start(hw, skb_3, &txctl) != 0)
{
    printk("\n TI-MAC POLL ERROR \n");
}
while(!pkt_txed);
    skb_3->data[22] += 0x10;
/* Update seq num */
sc->wtn_sta_txop_end = false;
ds_delay = msecs_to_jiffies(10);
set_current_state(TASK_UNINTERRUPTIBLE);
schedule_timeout(ds_delay);
current_jiffies = jiffies;
while(!sc->wtn_sta_txop_end)
{
    if(jiffies > current_jiffies)
        sc->wtn_sta_txop_end = true;
}

if(skb_4->data[22] == 0xf0)
    skb_4->data[22] = 0x00; /* Reset Seq Control */

pkt_txed = false;
if (ath_wtn_tx_start(hw, skb_4, &txctl) != 0)
{
    printk("\n TI-MAC POLL ERROR \n");
}
while(!pkt_txed);
    skb_4->data[22] += 0x10;
/* Update seq num */
sc->wtn_sta_txop_end = false;
ds_delay = msecs_to_jiffies(10);
set_current_state(TASK_UNINTERRUPTIBLE);
schedule_timeout(ds_delay);
current_jiffies = jiffies;

```

```

while (!sc->wtn_sta_txop_end)
{
    if (jiffies > current_jiffies)
        sc->wtn_sta_txop_end = true;
}

current_jiffies = jiffies;
txop_remain = 10*1000; /* microsec */

txop_req = (((skb_data_1->len + 4)*8)/36) + 158 + 20; /*in uSec */
while (txop_remain >= txop_req )
{
    /* send station traffic now */
    if (skb_data_1->data[22] == 0xf0)
        skb_data_1->data[22] = 0x00;
    pkt_txed = false;
    /* (data_len + FCS)*8 bits/2mbps + 158(SIFS+PREMBLE)*/
    if (txop_req <= txop_remain)
    {
        if (ath_wtn_tx_start(hw, skb_data_1, &txctl) != 0)
        {
            printk("\n TI-MAC Data ERROR \n");
        }
        skb_data_1->data[22] += 0x10;
        while (!pkt_txed);
        txop_remain -= txop_req;
    }
}
udelay(20);

current_jiffies = jiffies;
txop_remain = 10*1000; /* microsec */

txop_req = (((skb_data_2->len + 4)*8)/36) + 158 + 20; /*in uSec */
while (txop_remain >= txop_req )
{
    /* send station traffic now */
    if (skb_data_2->data[22] == 0xf0)
        skb_data_2->data[22] = 0x00;
    pkt_txed = false;
    /* (data_len + FCS)*8 bits/2mbps + 158(SIFS+PREMBLE)*/
    if (txop_req <= txop_remain)
    {
        if (ath_wtn_tx_start(hw, skb_data_2, &txctl) != 0)
        {
            printk("\n TI-MAC Data ERROR \n");
        }
    }
}

```

```

        skb_data_2->data[22] += 0x10;
        while(!pkt_txed);
        txop_remain -= txop_req;
    }
    udelay(20);
}

current_jiffies = jiffies;
txop_remain = 10*1000; /* microsec */

txop_req = (((skb_data_3->len + 4)*8)/36) + 158 + 20; /*in uSec */
while(txop_remain >= txop_req )
{
    /* send station traffic now */
    if(skb_data_3->data[22] == 0xf0)
        skb_data_3->data[22] = 0x00;
    pkt_txed = false;
    /* (data_len + FCS)*8 bits/2mbps + 158(SIFS+PREMBLE)*/
    if(txop_req <= txop_remain)
    {
        if (ath_wtn_tx_start(hw, skb_data_3, &txctl) != 0)
        {
            printk("\n TI-MAC Data ERROR \n");
        }
        skb_data_3->data[22] += 0x10;
        while(!pkt_txed);
        txop_remain -= txop_req;
    }
    udelay(20);
}

current_jiffies = jiffies;
txop_remain = 10*1000; /* microsec */

txop_req = (((skb_data_4->len + 4)*8)/36) + 158 + 20; /*in uSec */
while(txop_remain >= txop_req )
{
    /* send station traffic now */
    if(skb_data_4->data[22] == 0xf0)
        skb_data_4->data[22] = 0x00;
    pkt_txed = false;
    /* (data_len + FCS)*8 bits/2mbps + 158(SIFS+PREMBLE)*/
    if(txop_req <= txop_remain)
    {
        if (ath_wtn_tx_start(hw, skb_data_4, &txctl) != 0)
        {

```

```

        printk("\n TI-MAC Data ERROR \n");
    }
    skb_data_4->data[22] += 0x10;
    while(!pkt_txed);
    txop_remain -= txop_req;
}
udelay(20);
}

if(wtn_task_started==false)
    break;
}
printk("\nComing out of state machine");
kfree_skb(skb_1);
kfree_skb(skb_2);
kfree_skb(skb_3);
kfree_skb(skb_4);
kfree_skb(skb_data_1);
kfree_skb(skb_data_2);
kfree_skb(skb_data_3);
kfree_skb(skb_data_4);
}

int ath_wtn_tx_start \
(struct ieee80211_hw *hw, struct sk_buff *skb_poll, struct ath_tx_control *txctl)
{
    struct sk_buff *skb;
    skb = skb_copy(skb_poll, GFP_KERNEL);
    return ath_tx_start(hw, skb, txctl);
}

```

A.2 Remote STA State Machine Code

```

void ath_timac_sta_tx_work(struct work_struct *work)
{
    struct ath_softc *sc = container_of(work, struct ath_softc, wtn_sta_tx_work);
    struct ieee80211_hw *hw = sc->hw;
    struct ath_tx_control txctl;
    struct ath_hw *ah = sc->sc_ah;
    struct ath_common *common = ath9k_hw_common(ah);
    struct sk_buff *skb, *skb_txop_end;
    struct ieee80211_hdr qos_hdr;
    /*unsigned char src_mac[]={0x00,0x24,0xb2,0xd8,0x4f,0x1b};*/ /* STA1 */
    /*unsigned char src_mac[]={0xC0,0x3F,0x0E,0x7A,0x35,0xDB};*/ /* STA 2 */
    unsigned char src_mac[]={0x00,0x24,0xb2,0x57,0xCA,0x5E}; /* STA 3 */
    /*unsigned char src_mac[]={0x00,0x26,0xF2,0xF3,0x1A,0x6D};*/ /* STA 4 */
}

```

```

unsigned char dest_mac[]={0x00,0x24,0xb2,0x5a,0x9f,0x82};
int band = hw->conf.channel->band;
struct ieee80211_supported_band *sband = &sc->sbands[band];
unsigned long current_jiffies, data_txed=0;
int txop_req = 0, txop_remain = 0;
volatile unsigned long ds_delay;
__le16 fc;
__le32 *qos_control;
struct ieee80211_tx_info *tx_info, *tx_info_txop_end;
struct ieee80211_sta sta;
int pkt_cnt = 0, error = 0;
skb = alloc_skb(1600, GFP_KERNEL);
if (!skb)
{
    printk("\nSKB ALLOCATION ERROR IN TI-MAC TASK\n");
    return;
}

skb_txop_end = alloc_skb(100, GFP_KERNEL);
if (!skb_txop_end)
{
    printk("\nskb_txop_end ALLOCATION ERROR IN TI-MAC TASK\n");
    return;
}

tx_info = IEEE80211_SKB_CB(skb);
memset(tx_info, 0, sizeof(*tx_info));
tx_info_txop_end = IEEE80211_SKB_CB(skb_txop_end);
memset(tx_info_txop_end, 0, sizeof(*tx_info_txop_end));
memset(&qos_hdr, 0, sizeof(qos_hdr));
memset(&txctl, 0, sizeof(txctl));
/*memset(&sta, 0, sizeof(sta));*/
tx_info->control.sta = &sta;*/
tx_info->flags=0x14;
tx_info->band=band;
tx_info->control.rates[0].idx = 9;
tx_info->control.rates[0].count = 1;
tx_info_txop_end->flags=0x14;
tx_info_txop_end->band=band;
tx_info_txop_end->control.rates[0].idx = 1;
tx_info_txop_end->control.rates[0].count = 4;
skb_reserve(skb, 100);
memset(skb_put(skb, 1050), 0x33, 1050);
fc = cpu_to_le16(IEEE80211_FTYPE_DATA | IEEE80211_STYPE_QOS_DATA);
skb_reserve(skb_txop_end, 40);
memset(skb_put(skb_txop_end, 2), 0xdd, 2);

```

```

qos_hdr.frame_control = fc;
qos_hdr.duration_id = 0x00;
qos_hdr.seq_ctrl=0;
/* BSSID SA DA */
memcpy(qos_hdr.addr1, dest_mac, ETH_ALEN);
memcpy(qos_hdr.addr2, src_mac, ETH_ALEN);
memcpy(qos_hdr.addr3, dest_mac, ETH_ALEN);
qos_control = (__le32*) skb_push(skb, 4);
memcpy(skb_push(skb, 24), &qos_hdr, 24);
*qos_control = 0x21002100;
qos_control = (__le32*) skb_push(skb_txop_end, 4);
memcpy(skb_push(skb_txop_end, 24), &qos_hdr, 24);
*qos_control = 0x01000100;
skb->priority = 1;
skb_set_queue_mapping(skb, 3);
skb_txop_end->priority = 1;
skb_set_queue_mapping(skb_txop_end, 3);
txctl.txq = sc->tx.txq_map[skb_get_queue_mapping(skb)];
ds_delay = msecs_to_jiffies(5000);
set_current_state(TASK_UNINTERRUPTIBLE);
schedule_timeout(ds_delay);
ath9k_disable_ps(sc);
sc->wtn_sta_task = current;
printk("\nCurrent Task = %d ", sc->wtn_sta_task);
printk("\nTXOP_REQUIRED = %d", (((skb->len + 4)*8)/36) + 158+20));
set_user_nice(current, -20);
while(1)
{
    txop_remain = (sc->wtn_txop_jiffies)*10*1000; /* microsec */
    txop_req = (((skb->len + 4/*FCS=4*/)*8/*bits*/)/36/*rate*/ + 158+20;/*in uS
    while(txop_remain >= txop_req)
    {
        if(skb->data[22] == 0xf0)
            skb->data[22] = 0x00; /* Reset the Seq Control */
        pkt_txed = false;
        if(txop_req <= txop_remain)
        {
            if(ath_wtn_tx_start(hw, skb, &txctl) !=0)
            {
                printk("\n STA PKT ERROR \n");
            }

            /* Wait for TX INT */
            skb->data[22] += 0x10; /* Update seq num */

            while(!pkt_txed);

```

```

        txop_remain -= txop_req;
    }
    /* this delay was added for 54mbps, for 12,9,6,2 results were
taken without this */
    udelay(20);
}

if (skb_txop_end->data[22] == 0xf0)
    skb_txop_end->data[22] = 0x00; /* Reset the Seq Control */
pkt_txed = false;
if (ath_wtn_tx_start(hw, skb_txop_end, &txctl) != 0)
{
    printk("\n STA PKT ERROR \n");
}
/* Wait for TX INT */
skb_txop_end->data[22] += 0x10; /* Update seq num */
while (!pkt_txed);
    if (wtn_sta_task_started==false)
        break;

    set_current_state(TASK_UNINTERRUPTIBLE);
    schedule();
}

kfree_skb(skb);

return;
}

```

Bibliography

- [1] CISCO : The Future of Hotspots: Making Wi-Fi as Secure and Easy to Use as Cellular. last googled on 12th dec 2011.
- [2] Hequan Wu. Some thoughts on the transformation of information and communication technologies. In *Technologies Beyond 2020 (TTM), 2011 IEEE Technology Time Machine Symposium on*, page 1, june 2011.
- [3] N. Papaoulakis. Congestion avoidance mechanism in wlans. In *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*, pages 1 –5, sept. 2007.
- [4] CISCO : [http://newsroom.cisco.com/press-release content?type=webcontent&articleId=324003](http://newsroom.cisco.com/press-release/content?type=webcontent&articleId=324003). last cited on 13th nov 2011.
- [5] M. Cardenete-Suriol, J. Mangues-Bafalluy, A. Maso, and M. Gorricho. Characterization and comparison of skype behavior in wired and wireless network scenarios. In *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 2167 –2172, nov. 2007.
- [6] Skype. <http://about.skype.com>. 2011-last cited on 29th April 2011.
- [7] Ebay. <http://ebayinkblog.com/wp-content/uploads/2009/01/skype-fast-facts-q4-08.pdf>. 2011-last cited on 29th April 2011.
- [8] N. Seitz. ITU-T QoS standards for IP-based networks. *Communications Magazine, IEEE*, 2003, 41(6):82–89, 2003.
- [9] IEEE802.11Standard. Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pages C1–1184, 2007.
- [10] J. Wyatt, D. Habibi, I. Ahmad, and H. Zen. Providing qos for symmetrical voice/video traffic in wireless networks. In *15th IEEE International Conference on Networks, ICON*, pages 312–317, 2007.

- [11] K.D. Huang, K.R. Duffy, D. Malone, and D.J. Leith. Investigating the validity of ieee 802.11 mac modeling hypotheses. In *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, pages 1 –6, sept. 2008.
- [12] P. Latkoski, T. Janevski, and B. Popovski. Modelling and simulation of ieee 802.11a wireless local area networks using sdl. In *Electrotechnical Conference, 2006. MELECON 2006. IEEE Mediterranean*, pages 680 –683, may 2006.
- [13] T. Sadiki and P. Paimblanc. Modelling new indoor propagation models for wlan based on empirical results. In *Computer Modelling and Simulation, 2009. UKSIM '09. 11th International Conference on*, pages 585 –588, march 2009.
- [14] C.J.A. Bastos-Filho, J.D. Andrade, M.R.S. Pita, and A.D. Ramos. Impact of the quality of random numbers generators on the performance of particle swarm optimization. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 4988 –4993, oct. 2009.
- [15] Minyong Qi and Jinxin Dong. Research and application of entropy in the sequence randomness test. In *Computational Intelligence and Industrial Applications, 2009. PACIA 2009. Asia-Pacific Conference on*, volume 2, pages 224 –227, nov. 2009.
- [16] WARPnet: <http://warp.rice.edu/trac/wiki/CSMAMAC>. last cited on 10th october 2011.
- [17] Riccardo Manfrin, Andrea Zanella, and Michele Zorzi. Functional and performance analysis of calradio 1 platform. *2009 Eighth IEEE International Symposium on Network Computing and Applications*, 2009.
- [18] OpenWRT : <http://downloads.openwrt.org/backfire/>. last cited on 10th october 2011.
- [19] ath9k device driver: <http://linuxwireless.org/en/users/Drivers/ath9k>. last cited on 10th october 2011.
- [20] J. Wyatt and D. Habibi. Wtn - providing qos for voip in 802.11 networks. In *IEEE Region 10 Conference TENCN 2006. 2006*, pages 1–4.
- [21] S. Shenker et al. Specification of guaranteed quality of service, 1997. 1997.
- [22] J.K. Choi, J.S. Park, J.H. Lee, and K.S. Ryu. Review on qos issues in ieee 802.11 w-lan. In *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, volume 3, pages 5 pp. –2113, feb. 2006.

- [23] Hongqiang Zhai, Xiang Chen, and Yuguang Fang. A call admission and rate control scheme for multimedia support over ieee 802.11 wireless lans. In *Quality of Service in Heterogeneous Wired/Wireless Networks, 2004. QSHINE 2004. First International Conference on*, pages 76 – 83, oct. 2004.
- [24] A. Lindgren, A. Almquist, and O. Schelen. Evaluation of quality of service schemes for ieee 802.11 wireless lans. In *Local Computer Networks, 2001. Proceedings. LCN 2001. 26th Annual IEEE Conference on*, pages 348–351, 2001.
- [25] Jing-Yuan Yeh and Chienhua Chen. Support of multimedia services with the ieee 802-11 mac protocol. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 1, pages 600–604, 2002.
- [26] Shenglin Shi, Guangxi Zhu, and Gang Su. An optimized qos traffic-scheduling algorithm based on hcca. In *Intelligent Computation Technology and Automation (ICICTA), 2008 International Conference on*, volume 2, pages 229–233, oct. 2008.
- [27] C. Casetti, C.-F. Chiasserini, L. Merello, and G. Olmo. Supporting multimedia traffic in 802.11e w lans. In *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, volume 4, pages 2340 – 2344 Vol. 4, may-1 june 2005.
- [28] Wang Wei, Liew Soung Chang, and V. O. K. Li. Solutions to performance problems in voip over a 802.11 wireless lan, 2005.
- [29] Koo Ki-Jong and Kim Do-young. A scalable speech/audio coder control algorithm to improve the qos of voip calls over w lans. In *14th European Wireless Conference, EW*, pages 1–6, 2008.
- [30] Quan Wan and Hui Du Ming. Improving the performance of wlan to support voip application. In *2nd International Conference on Mobile Technology, Applications and Systems*, pages 5 pp.–5, 2005.
- [31] Yuan Wenpeng, Zhu Guangxi, and Liu Gan. Cross-layer schemes for optimization of voip over 802.11e wlan. In *IEEE Global Telecommunications Conference, GLOBECOM*, pages 4883–4887, 2007.
- [32] Yang Lu, Chao Zhang, Jianhua Lu, and Xiaokang Lin. A mac queue aggregation scheme for voip transmission in wlan. In *IEEE Wireless Communications and Networking Conference, WCNC*, pages 2121–2125, 2007.
- [33] Guo Fanglu and Chiueh Tzi-cker. Software tdma for voip applications over ieee802.11 wireless lan. In *IEEE 26th IEEE International Conference on Computer Communications. INFOCOM*, pages 2366–2370.
- [34] M. Shimakawa, D. P. Hole, and F. A. Tobagi. Video-conferencing and data traffic over an ieee 802.11g wlan using dcf and edca. In *IEEE International Conference on Communications, ICC*, volume 2, pages 1324–1330 Vol. 2, 2005.

- [35] Hsu Ming-Chuan and Chen Yaw-Chung. Enhanced pcf protocols for real-time multimedia services over 802.11 wireless networks. In *26th IEEE International Conference on Distributed Computing Systems Workshops, ICDCS Workshops*, pages 56–56, 2006.
- [36] Luo Hongli and Shyu Mei-Ling. An optimized scheduling scheme to provide quality of service in 802.11e wireless lan. In *11th IEEE International Symposium on Multimedia, ISM*, pages 651–656, 2009.
- [37] A. R. Rebai, S. Hanafi, and H. Alnuweiri. A new inter-node priority access enhancement scheme for ieee 802.11 wlangs. In *9th International Conference on Intelligent Transport Systems Telecommunications, ITST*, pages 520–525, 2009.
- [38] H. Zen, D. Habibi, A. Rassau, I. Ahmad, and J. Wyatt. A segregation based mac protocol for real-time multimedia traffic in wlangs. In *IFIP International Conference on Wireless and Optical Communications Networks, WOCN*, pages 1–5, 2007.
- [39] Z.A.B.M. Noh, T. Suzuki, and S. Tasaka. A packet scheduling scheme for audio-video transmission with ieee 802.11e hcca and its application-level qos assessment. In *Communications, 2006. APCC '06. Asia-Pacific Conference on*, pages 1 –5, 31 2006-sept. 1 2006.
- [40] Zhengyong Feng, Guangjun Wen, Zixuan Zou, and Fanqing Gao. Red-txop scheme for video transmission in ieee802.11e edca wlan. In *Communications Technology and Applications, 2009. ICCTA '09. IEEE International Conference on*, pages 371 –375, oct. 2009.
- [41] Yigal Bejerano, Hyoung-Gyu Choi, Seung-Jae Han, and Thyaga Nandagopal. Performance tuning of infrastructure-mode wireless lans. In *Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, pages 60–69, 2010.
- [42] Chen Yaw-Chung and Yeh Han-Ru. An adaptive polling scheme supporting audio/video streaming in wireless lans. In *12th IEEE International Workshop on Future Trends of Distributed Computing Systems, FTDCS*, pages 16–22, 2008.
- [43] Guo Tiantian, Cai Jianfei, Foh Chuan Heng, and Zhang Yu. Improving videophone transmission over multi-rate ieee 802.11e networks. In *IEEE International Conference on Communications, ICC*, pages 3258–3262, 2008.
- [44] SpectraLink : SpectraLink Voice Priority Quality of service for voice traffic on wireless LANs.
- [45] www.st.com. Um0488 user manual, stm3210e-eval evaluation board. 2010.

- [46] www.sagrad.com. Sagrad sg901-1028 miniature wi-fi radio, doc: Sg914-0029 rev. 0.5. 2010.
- [47] www.sagrad.com. Sg901-1087 kraken ii intelligent wi-fi radio, doc: Sg914-0036 rev. 1.0. 2010.
- [48] www.st.com. Stlc4560, single chip 802.11b/g wlan radio data sheet, rev 1. 2008.
- [49] ITU-T. Narrow-band visual telephone systems and terminal equipment. H.320, 2004.
- [50] ITU-T. Visual telephone systems and terminal equipment for local area networks which provide a guranteed quality of service. H.322, 1996.
- [51] <http://en.wikipedia.org/wiki/Opticalfiber>. Optical fibre, last cited 03/09/2011. 2011.
- [52] OpenWRT : <http://wiki.openwrt.org/toh/netgear/wndr3700>. last cited on 10th october 2011.
- [53] A.P. Ortega, X.E. Marcos, L.D. Chiang, and C.L. Abad. Preventing arp cache poisoning attacks: A proof of concept using openwrt. In *Network Operations and Management Symposium, 2009. LANOMS 2009. Latin American*, pages 1 –9, oct. 2009.
- [54] C.E. Palazzi, M. Brunati, and M. Roccetti. An openwrt solution for future wireless homes. In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pages 1701 –1706, july 2010.
- [55] OpenWRT : <https://openwrt.org/>. last cited on 10th october 2011.
- [56] Atheros. Product overview : Ar7161 high performance, wireless network processor. 2010.
- [57] Atheros. Datasheet : Ar9280 single-chip 2xmimo mac/bb/radio with pci express interface for 802.11n 2.4 and 5 ghz wlans. Oct, 2009.
- [58] M. Vipin and S. Srikanth. Analysis of open source drivers for ieee 802.11 wlans. In *Wireless Communication and Sensor Computing, 2010. ICWCSC 2010. International Conference on*, pages 1 –5, jan. 2010.
- [59] ITU-T. Visual telephone systems and terminal equipment for local area networks which provide a guranteed quality of service. H.322, 1996.
- [60] T. Frantti. Fuzzy packet size optimization for delay sensitive traffic in ad hoc networks. In *IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 2633–2637, 2009.
- [61] Jordi Perez-Romero, Oriol Sallent, Ramon Agusti, and Miguel Angel Diaz-Guerra. Video-telephony traffic model. In *Radio resource management strategies in UMTS*, page 298, 2007.