A PROGRAMMABLE MAC BASED SYSTEM FOR REAL-TIME AND NON REAL-TIME FLOWS IN WIRELESS NETWORKS

A Thesis

by

KARTIC BHARGAV KAVERIPURAM RAMASAMY

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2017

Major Subject: Computer Engineering

ABSTRACT

Wireless networks are increasingly being used to serve both real-time and non real-time flows. The former includes applications such as VoIP and video streaming, while the latter includes applications like file transfer and web browsing. These flows have very different service requirements. On the one hand, real-time flows usually require a strict per-packet delay bound, since late packets may not be useful to the application. On the other hand, non real-time flows do not pose any stringent delay requisites and only demand high throughput.

Serving flows that have heterogeneous requirements necessitates the deployment of algorithms and rules for resource allocation that attempt to satisfy these needs. However, existing hardware does not allow such reconfigurability and is limited to providing a once-size-fits all solution. The objective of this work is to design, develop and demonstrate an architecture, specifically for software reconfigured hardware at the PHY-MAC layers that can provide such functionality at a per-flow and per-packet level, and to illustrate its superior performance to conventionally deployed solutions.

ACKNOWLEDGMENTS

I thank my advisor, Dr. Srinivas Shakkottai for his constant support and guidance throughout the course of my graduate studies. He has always been approachable, understanding and permissive. It has been a remarkable learning experience working under his supervision. I would also like to extend credit to Dr. I-Hong Hou for his resourceful insights as a collaborator of the project. Special thanks to my committee-members, Dr. Jean-Francois Chamberland and Dr.Radu Stoleru for providing valuable inputs and suggestions during my Thesis defense.

I am grateful to Simon Yau for his mentorship during the experimental phase of this project. Despite being several timezones away, he always took time out to promptly direct me at times when my progress was slow. His constructive propositions at times when the system didn't behave as expected were very helpful. I also thank Ping-Chun Hsieh for being an excellent collaborator during the development and testing phase of this project.

I appreciate all my course instructors providing a conducive learning environment with productive discussions both in class and in person. More specifically, courses on Computer Communication & Networking by Dr. A.L. Narasimha Reddy, Algorithms by Dr. Anxiao Jiang and Systems based courses by Dr. Dmitri Loguinov were instrumental at different levels of my graduate life including research, systems development and job-search. Channel Coding by Dr. Krishna Narayanan and the project-based learning course by Dr. Jean-Francois Chamberland and Dr. Gregory Huff were enjoyable and motivational in their own aspects.

Lastly I would like to thank my friends & labmates for keeping the fun alive and my family for their unconditional love and support always.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

A part of this work on Reconfigurable MAC was done in collaboration with Ping-Chun Hsieh, Simon Yau and Dr. I-Hong Hou. The results using Reconfigurable Antennas discussed in Sections 4 and 5 were obtained in collaboration with Abhay S. Anand, Dr. Gregory Huff and Dr. J-F. Chamberland. The implementation of policies discussed in Sections 4 and 5 was done in collaboration with Rajarshi Bhattacharyya. All collaborators belong to the Department of Electrical & Computer Engineering at Texas A&M University.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by the Graduate Merit Scholarship (2014-2016), offered by the Department of Electrical & Computer Engineering at Texas A&M University and a research grant funded by the EARS program of the National Science Foundation (2015-2016).

NOMENCLATURE

AP	Access Point
MAC	Medium Access Control
PHY	Physical Layer
USRP	Universal Software Radio Peripheral
SDN	Software Defined Networks
SDR	Software Defined Radio
NI	National Instruments
QoS	Quality of Service
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
FPGA	Field Programmable Gate Array
DSP	Digital Signal Processor
CSI	Channel State Indicators
MCS	Modulation and Coding Scheme
ACK	Acknowledgement
CSMA	Carrier Sense Multiple Access
VI	Virtual Instrument (in LabVIEW)
DMA	Direct Memory Access

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Wireless networks are being increasingly used to serve both real-time and non real-time traffic. The former involves applications like VoIP and multimedia streaming while the latter involves applications like File-Transfer and web browsing. These two flows pose very different service requirements. On one hand, real-time flows usually require a strict per-packet delay guarantee since late packets might not be of use to the application. On the other hand, non real-time flows do not pose any stringent delay requisites and only demand high throughput.

Next-generation applications in areas such as Augmented Reality (AR), vehicular networks and Unmanned Aerial Systems (UAS) go a step further in imposing tightness of the delay constraint. The current WiFi standards have been primarily optimized to support best-effort traffic and can certainly not meet the few millisecond level latency requirement that these control applications expect. This is due to the fact that WiFi is built around random access, and treats all packets identically irrespective of the nature of the flow. Thus, wireless networks need to evolve not only to satisfy throughput requirements but to also meet other stipulations on low-latency.

While quite a few Medium Access Control (MAC) protocols have been proposed to target the above mentioned metrics, not many have been experimentally evaluated to establish realistic performance guarantees. MAC protocols have very strict timing requirements, which leads to tight coupling between the protocols and the underlying hardware. Our goal is to fill this void in building a platform for such experimentation, and conducting realistic performance analysis experiments on it.

Data networks require decision making rules at both an individual packet level, as well as the level of aggregates of packets such as flows. The timescale at which these decision rules operate are different. We entitle the low-level decisions made on a per-packet basis such as modulation and scheduling decisions at the sub-millisecond timescale as “mechanisms”, while high-level schemes on a larger-timescale basis over aggregates of packets that choose between different scheduling schemes are called “policies”. For example, two available mechanisms at the MAC layer might be Max-weight scheduling and Deficit-based queuing (to be defined later), whereas the choice of which of them to employ on a flow is a policy decision.

Our first objective is to implement mechanisms (operating at timescales less than 1ms) that account for different service requirements of non real-time (elastic) and real-time (inelastic) flows on a per-packet level. Choosing a set of mechanisms results in a set of performance statistics perceived by flows, such as throughput, latency, jitter and loss rate, which together quantify the Quality of Service (QoS) of that particular flow. We also desire to study the impact on QoS on flows attained by choosing different sets of mechanisms.

Following this, we develop control schemes using which policy decisions (operating at larger timescales of the order of seconds) on which mechanisms to deploy can be selected in a centralized fashion. Thus, our second goal is to construct a framework, whereby these decisions can be communicated to a wireless Access Point (AP), and its impact on the QoS of a particular flow can be measured.

1.1 System Architecture

We now proceed to illustrate a framework designed to achieve our centralized control objective. Figure 1.1 shows a high-level overview of the complete control system at the wireless Access Point (AP). The diagram illustrates per-packet mechanisms at three layers that we would ideally desire to implement. At the physical layer, the antenna radiation pattern determines the received signal strength at the destination. Also at the physical layer, the modulation scheme employed chooses how best to utilize the received signal strength by choosing an appropriate modulation scheme. Finally, at the data link layer, MAC chooses which packet to schedule from different competing flows at each time.

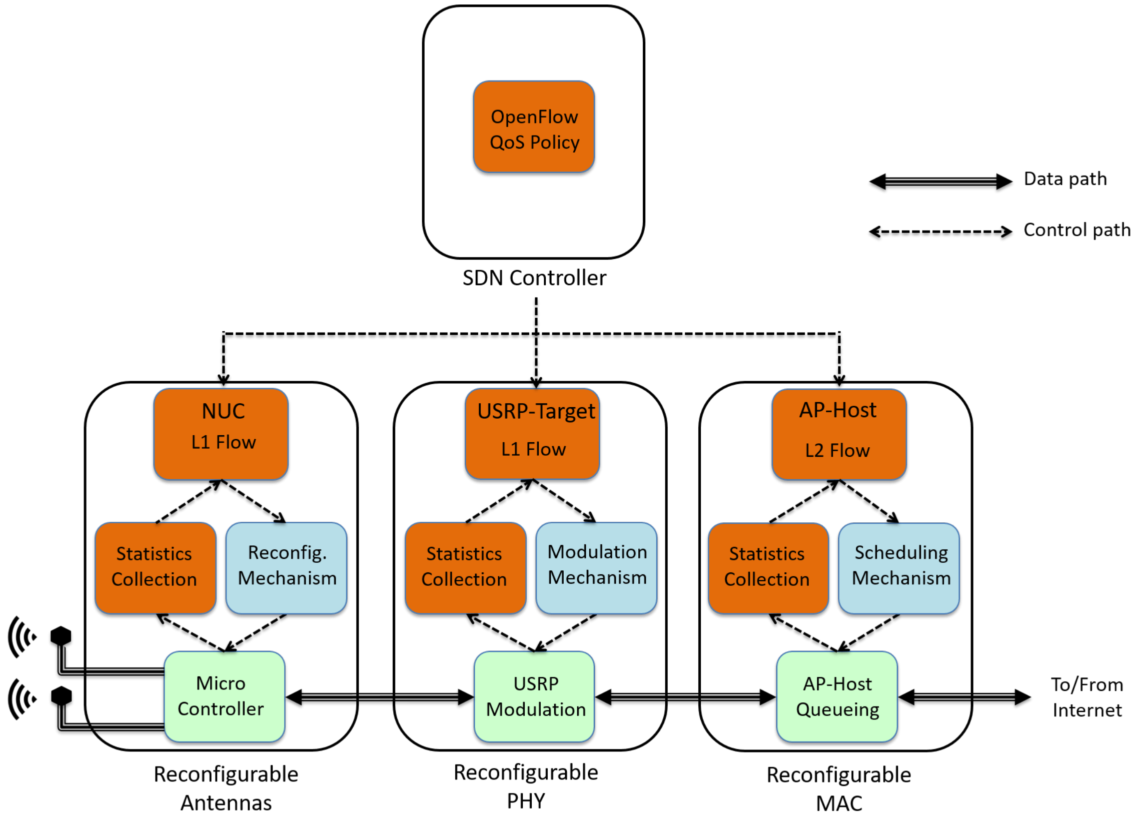

Figure 1.1: High-Level Overview of the Wireless Control

These three elements are typically implemented on a single chipset with tight integration in commercially available hardware, with little room for choice between possible mechanisms. The principle reason for such integration is the need to support large data rates, which implies per-packet processing and transmission must be accomplished at sub-millisecond timescales. However, central to our ideas is the ability to reconfigure hardware dynamically. Thus, we require a platform that can support fast reconfigurability, while yet supporting reasonable data rates.

Our architecture is built around a programmable MAC platform called WiMAC [1], which is a general-purpose wireless testbed for the rapid prototyping of user-definable MAC protocols. The hardware of WiMAC consists of an NI USRP that supports the modulation and coding schemes, as well as simple scheduling decisions, coupled with a host computer (typically a laptop) that supports more complex scheduling decisions, as well as networking and application functionalities. This platform is paired with software reconfigurable antennas that are controlled by an independent setup using an Arduino microcontroller providing decisions serially communicated to it via a host computer (typically an Intel NUC).

As mentioned above, the functionality of WiMAC in taking scheduling decisions is split between the Host (software) and the USRP Target (hardware). Both are programmed using the LabView Communications Design Suite with an appropriate use-point (host or USRP) specified for running each block. At the current stage of evolution, WiMAC allows for relatively simple scheduling decisions to be made on the USRP, necessitating the need to push most aspects of scheduling decisions to the host computer. However, per-packet processing is still feasible at the sub-millisecond range, and hence throughput of the order of $10 - 15$ Mbps is still attainable. Thus, in our architecture, the mechanism is implemented on the host computer. Finally, we note that it appears that this limitation will likely be overcome using the next version of LabView Communications that allows for greater functionality (and hence higher per-packet processing speeds) on the USRP.

An additional limitation at the current time is our inability to configure the antenna radiation pattern at a per-packet level. The reconfigurable antennas are driven through a system independent of the USRP that actually performs the modulation and coding functionalities. Thus, the radiation pattern has to be relatively static as far as each packet is concerned, and can only be changed over packet aggregates. This limitation too appears to be temporary, and we are already working on driving antenna reconfiguration directly via USRP to attain per-packet selection of the radiation pattern.

Once we have a platform for mechanism design, we need to be able to actually select which combination of mechanisms to use on a packet aggregate basis. As mentioned above, we entitle this choice as a policy decision, and the policy is what results in a certain vector of QoS statistics to be achieved for a particular flow. Furthermore, the impact of a policy decision on QoS needs to be communicated back, so as to obtain feedback on the performance of a particular flow.

Our architecture uses centralized determination of policies, and our choice of protocol for communicating policy decisions as well as feedback on their impact is OpenFlow [2]. OpenFlow provides a completely defined communication protocol that enables a centralized controller to interact with the connected switches in order to perform further processing on routing and packet forwarding. As is clear from the description, OpenFlow is intended to operate on the network layer, whereas we desire functionality on the PHY-MAC layers. Indeed, OpenFlow is an implementation of the concept of Software Defined Networking (SDN) that calls for dynamic reconfiguration at the network layer. However, we find that this limitation to the network layer is relatively easy to overcome, and the advantage provided by OpenFlow in terms its increasing hardware support outweighs any benefits of designing a custom protocol.

To summarize, the overall objective of this work is to create a set of mechanisms that can be implemented on the WiMAC, with a corresponding set of policies selected using a centralized SDN controller communicating OpenFlow. Built on a mechanism-policy separation framework, the proposed system intends to achieve reconfigurability at the PHY, MAC and antenna layers.

1.2 Related Work - Mechanisms

1.2.1 Max-Weight Scheduling

The class of Max-Weight scheduling policies has been extensively studied in networking theory since its first conception in [3]. The algorithm operates by finding a maximum weight schedule of the conflict graph at each instant of decision. The weight is usually characterized by the state parameters of each client, such as the queue length of the packet buffer and the transmission rate of the wireless channel.

Max-Weight scheduling has been applied to various setups of wireless networks to achieve optimal throughput performance without having prior-knowledge of the nature of arrival processes [4, 5, 6, 7]. Moreover, Max-Weight scheduling has also been proved to achieve good delay performance under some conditions [8, 9].

As a centralized control algorithm, Max-Weight is especially suitable for wireless infrastructure-based networks where the Access Point (AP) can periodically collect channel and queue state information of its clients. This would be followed by solving the maximum weight independent set problem which, in our case reduces to simply finding a single link with the largest weight.

1.1.1.1 Q-CSMA: A Distributed Algorithm for Ad-hoc P2P Networks

The absence of a centralized infrastructure in ad-hoc P2P networks necessitates the nodes to perform their own activity decisions based on local information.

Q-CSMA (Queue-length based CSMA/CA) is a distributed approximation of the Max-Weight algorithm for ad-hoc P2P wireless networks which has been shown to be throughput optimal [10].

In [11, 12], it was suggested that, by using weights as functions of queue lengths, one can achieve throughput optimality. Proceeding this, a network utility maximization interpretation was provided for the CSMA algorithm [10, 13].

In Q-CSMA, a link makes transmission decisions based on the states of its neighbouring links in the preceding time slot. The weights correspond to some function of queue lengths and are assumed to change very slowly compared to the time needed for the CSMA Markov chain to attain steady state. Here, flowss are characterized according to their intended destinations (essentially Virtual Output Queueing).

1.1.1.2 Experimental Evaluation of Q-CSMA

Despite the marked progress in theoretical aspects of Max-Weight based scheduling policies, realistic implementations have been hard to come by. Majority of the current implementations have been used only to study parameter setting or as a verification over elementary scenarios [14, 15].

A notable exception amongst these class of implementations is the experimental evaluation of distributed optimal CSMA protocols by Nardelli et al. in [16].

The implementation itself revolves around a Common Code Architecture (CCA) presented in [15]. This comprises of both a simulation over Glomosim and a protocol implementation over standard 802.11 hardware which incorporates a driver modification to adapt the Contention Window (CW) of transmitters as the algorithm requires.

The authors have then proceeded to investigate the behavior of Q-CSMA in various scenarios differing in network topology as well as channel qualities of contending links. The throughput performance of Q-CSMA when exposed to congestion (with multiple competing TCP flows) has also been examined.

It is worthwhile to note that such an experimental study has not only deepened the general understanding of the protocol in question and the philosophy behind its conception, it could also potentially motivate the design of future optimal enhancements as required by the consumer.

1.2.2 Deficit based Queues for Delay-Sensitive Traffic

Providing services for flows with delay constraints over wireless links has gained extensive research interest over the past decade. The early work done in this regard [17, 18, 19] didn't theoretically address three very important problems for providing services: scheduling algorithms, admission control, and utility maximization.

At the same time, other well-known scheduling mechanisms proposed to enable QoS support on unreliable wireless channels [4, 5, 6, 7] fail to provide provable performance guarantee on per-packet delays.

To target these open-ended issues, work by Hou et al. [20] provides a tractable formulation for real-time wireless networks. The model incorporates both delay bounds and delivery ratio requirements for real-time flows in wireless networks.

The system in picture is an infrastructure based wireless network with an AP and a group of clients. Each client generates one real-time flow.

Time is slotted where the length of a time slot is chosen to be large enough to accommodate the time needed to transmit a packet with its overheads. These slots are further combined to form intervals, where packet arrivals are assumed to happen at the beginning of each interval. Packets from real-time flows are constrained with a delay bound equal to the length of the interval. In other words, packets arriving at the beginning of an interval are valid only if they are transmitted by the end of that interval. Packets unable to meet the delay bound are considered to have expired and are discarded from the system. Thus, dropping expired packets would guarantee that the delay of every delivered packet is at most one interval duration.

A client's performance is quantified by its timely-throughput, which is defined as the average number of successfully delivered packets per interval measured on a long-term basis. Each client is specified to have a rigid timely throughput demand (also called the delivery ratio).

The authors establish an admission control algorithm to evaluate the feasibility of a set of clients (with respect to the above criteria). They further propose two Largest-Deficit-First scheduling mechanisms and prove that these are feasibly optimal in the idea that they can satisfy the needs of every feasible set of clients.

Similar to Max-Weight though there has been theoretical progress in terms of enhancements proposed to the Largest-Deficit-First scheduling policies, there has been no practical implementation of the same.

1.2.3 Unresolved Issues with the Mechanism Propositions

Wireless networks are expected to support both best effort as well real-time traffic. As mentioned earlier, depending on their nature, flows demand specific QoS requisites including minimum bandwidth and maximum delay constraints. The framework proposed to fulfil these requirements should maintain the stability of the queues and guarantee throughput optimality as well.

The Max Weight scheduling based mechanisms proposed in literature (see [5] for a survey) only consider flows with best-effort traffic, thereby not taking into account strict per-packet delay bounds which are critical in real-time flows.

At the same time, although the framework proposed in [20] for Largest-Deficit-First based scheduling also provides a basic insight into how to deal with both elastic and inelastic flows, it lacks any kind of realistic implementation.

In this work, not only do we provide an implementation platform to incorporate both of the above scheduling mechanisms, we also enforce the ability to configure the

MAC policies that are part of these mechanisms real-time via a controller operating remotely.

1.3 Related Work - Policies

1.3.1 Software Defined Networks

Re-configurability has always been a major challenge in the networking area. The recent explosion of mobile devices and cloud-based services have created a need for reconfiguration of flow rules according to changing traffic patterns.

As the network scales, the complexity involved in maintaining it grows exponentially. Deploying fresh services too becomes an arduous task. Software Defined Networking (SDN) is an architecture which tries to address these challenges by decoupling the control and forwarding functions.

SDN is a promising concept that has the potential to reform the way networks are being designed, tested and operated [21]. Apart from enabling network administrators to have a centralized vision of the network, it also provides a standardized interface to configure devices remotely. This provides a platform to drive a large network through a logically centralized controller and to define custom policies as well.

1.3.2 OpenFlow

What began as a research project has now well advanced to become an open standard that enables researchers to deploy their innovations in network switches without forcing them to expose their internal implementations. Building on the well-documented control and data plane abstraction - OpenFlow [2] provides a completely defined communication protocol along with an abstraction of the flow tables.

Combined with a set of associated instructions, each entry in a flow table comprises of a number of fields for a packet to match to. The instruction set involves actions on the packet or modification of the pipeline processing to forward packets to other flow tables.

OpenFlow enables a centralized controller to interact with the switches by adding, deleting or modifying flow table entries in order to perform further processing. The protocol runs over TCP/TLS (Transport Layer Security) connection so that the communication remains private without having to compromise on the security of the whole network.

1.3.3 Unresolved Issues with Policy Reconfiguration using OpenFlow

OpenFlow is intended for reconfiguration of Network Layer protocols by programming the flow tables in Ethernet switches and routers so that the Layer 3 protocols can be tested and controlled completely via software [22, 23]. The prime use cases for SDNs have also been in Network Virtualization [24] or for security purposes [25]. Thus, both OpenFlow and, more generally, the SDN concept have always been targeted towards applications operating at the Network layer. We wish to extend the centralized control using OpenFlow protocol to incorporate policy selection at the MAC and PHY layers.

1.4 Organization

The rest of the thesis is organized as follows. Chapter 2 describes the traffic model and formulates the problem of maximizing the utility for the system in picture. Chapter 3 summarizes related efforts on MAC implementation including tweaking commodity hardware and Software-Defined Radio based approaches. This is followed by a discussion on the design principles of WiMAC and an overview of its 802.11 Application framework. Chapter 4 proceeds to talk about finer execution details including mechanism implementation using WiMAC and policy implementation using OpenFlow. The re-configurable system results are analyzed in Chapter 5 and the optimality of our approach is justified with performance comparisons. Chapter 6 concludes the thesis with an insight into possible extensions in future.

2. SYSTEM MODEL AND PROBLEM STATEMENT

The network is represented by the figure shown in Figure 2.1. There exists a single scheduler on the Access Point with a set of $\mathcal{L}$ downlinks originating from it, one to each of the clients. The links are depicted by l and are number from $1, 2, \cdots, \mathcal{L}$.

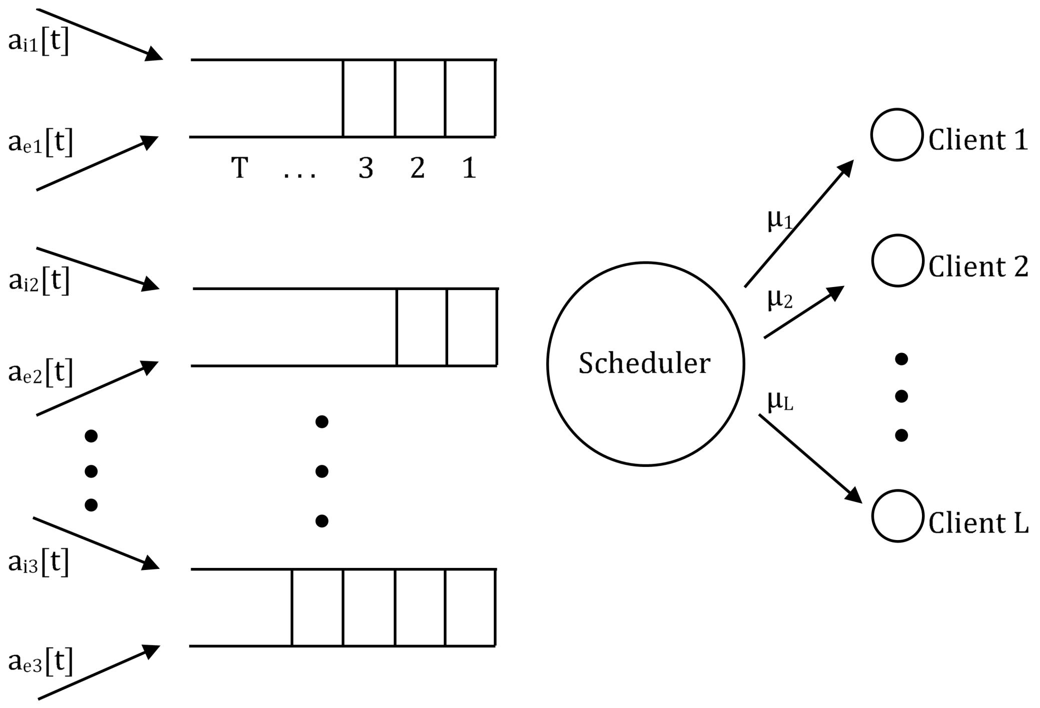

Figure 2.1: Downlink Scheduler

Traffic is taken to be composed of both elastic and inelastic flows, where an inelastic flow is one that has a per packet delay requirement imposed on it. As opposed to this, elastic flows do not have such requirements.

Time is slotted where the length of a time slot is chosen to be large enough to accommodate the time needed for transmitting a packet and all overheads with the least constellation size (BPSK).

A set of T consecutive time slots is grouped into an *interval*. We assume that packet arrivals only occur at the beginning of an interval, and every inelastic packet has a deadline of T time slots. In other words, packets that are generated at the beginning of an interval are only useful if they are delivered no later than the end of that interval. Packets unable to meet the delay bound are considered to have expired and are dropped from the system. Hence by dropping expired packets, it is guaranteed that the delay of every delivered packet is at most one interval duration.

It is required that the loss probability at link $l \in \mathcal{L}$ due to deadline expiry must be no more than q_l. For elastic traffic we correlate a utility function $U_l(x_{el})$ which is a function of the mean elastic arrival rate per interval x_{el}. $U_l(.)$ is assumed to be a concave function.

For a given interval, the vector $a_i = (a_{il})_{l\in\mathcal{L}}$ denotes the number of inelastic packet arrivals at every link, where a_{il} is a random variable with mean x_{il} and variance σ_{il}^2.

Similarly, $a_e = (a_{el})_{l\in\mathcal{L}}$ characterizes the number of elastic packet arrivals at every link in a given interval.

The channel state is assumed to be constant throughout a single interval, independent between two different intervals, and independent of packet arrivals. The vector $c = (c_l)_{l\in\mathcal{L}}$ depicts the number of packets link l can successfully transmit in one time slot of a given interval.

A feasible schedule $s = (s_{il,t}, s_{el,t})$ is one such that $s_{il,t}, s_{el,t}$ denote the number of inelastic and elastic packets respectively that could potentially be scheduled for transmission at link $l \in \mathcal{L}$ and time slot $t \in 1, 2, \cdots, T$. Hence, $s_{il,t} + s_{el,t} > 0$ translates to link l being scheduled to transmit in time slot t of the interval.

We now have the following constraints:

$$\sum_{t=1}^{T} s_{il,t} \le a_{il} \;\forall\; l \in \mathcal{L}$$
$$s_{il,t} + s_{el,t} \le c_l \;\forall\; l \in \mathcal{L} \text{ and } t \in \{1, 2, \cdots, T\}$$

We denote $\mathcal{S}(a_i, c)$ to be the set of all feasible schedules when the arrival state is a_i and the channel state is c.

At the beginning of every interval, a feasible schedule must be chosen to serve the links and decide the number of elastic packets to be injected in the network. Thus, one primary goal is to find a function $Pr(s|a_i, c)$ which is the probability of using schedule $s \in \mathcal{S}(a_i, c)$ when the inelastic arrivals are given by a_i and the channel state is c, subject to the constraint that the loss probability at link $l \in \mathcal{L}$ due to deadline expiry cannot exceed q_l. For the elastic traffic, we wish to select the vector a_e in such a way that we maximize the network utility while keeping the queues stable.

To proceed with the problem formulation, we first define $\mu_i(a_i, c)$ to be the expected number of inelastic packets served. Similarly, $\mu_e(a_i, c)$ denotes the expected number of elastic packets that can be served. Therefore, we have the following constraints:

$$\mu_{il}(a_i, c) \;\le\; \sum_{s \in \mathcal{S}(a_i,c)} \sum_{t=1}^{T} s_{il,t} Pr(s|a_i, c)$$

$$\mu_{el}(a_i, c) \;\le\; \sum_{s \in \mathcal{S}(a_i,c)} \sum_{t=1}^{T} s_{el,t} Pr(s|a_i, c)$$

For mixed traffic therefore, the expected service is given by:

$$\mu_{il} \;\overset{def}{=}\; \sum_{a_i} \sum_{c} \mu_{il}(a_i, c) Pr(c) Pr(a_i)$$

$$\mu_{el} \overset{def}{=} \sum_{a_i} \sum_c \mu_{el}(a_i, c) Pr(c) Pr(a_i)$$

and with the given capacity and QoS constraints: $\mu_{il} \geq x_l(1 - p_l)$ and $x_l \leq \mu_{el}$.

The objective is to maximize the following for a given vector $w \in \mathbb{R}_+^{|\mathcal{L}|}$ which weights the share of channel bandwidth to be allocated for the inelastic flows:

$$\max_{\substack{\mu_i(a_i,c),\, \mu_e(a_i,c) \\ \mu_i(a_i,c),\, \mu_e(a_i,c)}} \sum_{l \in \mathcal{L}} U_l(x_{el}) + w_l \mu_{il}$$

3. WIMAC: A TESTBED FOR PROGRAMMABLE MAC

Due to the advent and widespread use of emerging applications such as multimedia streaming, the traffic served by Wi-Fi has been rising exponentially. To cope with the continued increase and in some sense to stay ahead, wireless networks need to keep evolving not only to satisfy the throughput requirements but to also meet other crucial performance metrics such as low-latency & power consumption.

3.1 Background on Wireless Platforms

3.1.1 Need for MAC Prototyping

A wide variety of MAC protocols have been proposed in the academia in order to face the above mentioned challenges. But if you take a close look at the number of these propositions actually making it to the deployment stage, the number is surprisingly minimal [26, 27, 28, 29, 30].

This can be attributed to the lack of experimentation of the proposed protocols without which the performance guarantees provided would solely be based on simulation events alone. Needless to say, such results are inadequate when it comes to implementing them in real-world scenarios.

3.1.2 Tweaking Commodity Hardware

With ever-changing wireless standards and protocols, there has been a conscious shift towards a programmatic approach for designing and implementing wireless radios. In commercial wireless network interface cards, MAC layer operations are interlinked with the configuration of driver suites that are specific to individual chipsets, such as the ath family for Atheros chipsets and the rt2x00 family for Ralink chipsets.

However, these drivers are limited in terms of functionality and have no access to the basic MAC parameters, such as packet format and frame timing. To address this shortcoming, propositions like SoftMAC [31], MadMAC [32] and FlexMAC [33] follow design strategies that involve building a software platform to implement customized MAC protocols on commercial 802.11 hardware. However, despite the added functionality, having been built on commodity hardware, these platforms suffer from limited scope of redefinable methods and fail to support protocols outside the Carrier Sense Multiple Access (CSMA)-based category.

Following an alternate approach, Doerr et al. [34] propose MultiMAC to enable switching between multiple MAC protocols to achieve higher performance results in a varying environment. Despite the protocol-level adaptability, this platform too does not offer the required flexibility in MAC implementation.

3.1.3 Where does the MAC Layer fall Short?

Wireless protocols require different experimentation strategies depending on their location on the OSI stack.

PHY layer protocols are subject to stringent timing requirements of throughput sensitive flows. As is the case in Software Defined Radios (SDRs), with dedicated FPGAs or DSPs PHY layer experimentation can be conducted to meet the required specifications to establish realistic performance measures.

The Network Layer has to consider complex topologies in order to derive reachability information to make appropriate routing decisions. Experimentation of the Network Layer is achieved using abstraction in the software domain as is the case with OpenFlow [2]. This greatly aids in the effectuation of the network layer protocols which are being proposed.

Differing from the above two layers, implementing MAC protocols presents its own challenges. Since the MAC layer serves as the junction between the PHY and other high level layers, it is required to interface with the PHY closely and smoothly. As a result of this, commodity products usually have MAC functionality unified with the PHY components on a Network Interface Card (NIC). Implementing a new MAC protocol hence might need development from scratch thereby significantly prolonging the time needed for experimental testing.

3.1.4 Software Defined Radio based Wireless Platforms

To reduce the prototyping time for MAC protocols, and at the same time not compromise on flexibility, Software-Defined Radio (SDR) is a promising solution. In general, SDRs consist of three major parts: RF front end, baseband hardware and a software host. Since hardware development takes longer than software, it is desirable to perform as many functions as possible in the software domain to suit rapid development.

To achieve this flexibility, a good deal of wireless platforms have adopted the Software Defined Radio paradigm. The most popular example is the GNU Radio [35] which focuses most of its implementation in the software domain although taking a hit in terms of latency as discussed in [36].

In the family of FPGA-based software radio platforms, Wireless Open Access Research Platform (WARP) [37] is a popular high-performance hardware platform for research purposes. By using FPGA and PowerPC core, WARP enables full access to both MAC and PHY functions thereby achieving high flexibility. However, most FPGA-based platforms including WARP focus primarily on flexibility and latency performance, and therefore do not guarantee quick prototyping.

3.2 WiMAC - Key Design Principles

Hence we are in need of a rapid and flexible prototyping platform that validates the deployment of the proposed MAC protocols by accelerating the testing phase.

WiMAC [1] is a general-purpose wireless testbed for the rapid prototyping of user-definable MAC protocols. WiMAC is built around the idea that per-packet decisions, which we referred to as a mechanism require fast hardware supported decision making, while larger timescale decisions on the choice of mechanisms, which we referred to as policy selection, can be done via software. Built on a mechanism-policy separation framework, WiMAC achieves independence of software from hardware by revamping the usual tight-coupling between the MAC and PHY layers.

The key design principles for such a platform are:

1. MAC vs PHY decoupling: In many cases, the MAC and PHY layers are very tightly coupled. This is due to the stringent timing constraints required on the real-time communication between the two. Any change brought into one of them will necessitate a change in the other with minimal exceptions. This makes it tedious to implement MAC protocols that are exceedingly different from the existing ones. Hence to ensure rapid implementation capability, decoupling between MAC and PHY is essential.

2. Designing MAC protocols in software: This would not only increase the flexibility in terms of protocol design, it would also enable implementation of a larger class of algorithms since its generally easier to develop in software than hardware. Additionally, this would also expedite the experimentation process itself. However, not all parts of the implementation can be moved to software, since we still have to meet the fine constraints on the processing time as required by the MAC protocols.

3. Supporting dynamic protocol-configuration changes at runtime: This feature comes into play when incorporating real-time policy updates. Another scenario where this would prove instrumental is when we deal with implementing multiple MAC protocols at runtime. This would mean comparative experimentation since we can switch between different MAC protocols to observe their relative behavior when exposed to the same set of conditions.

3.3 802.11 Application Framework - Overview

We proceed to describe the functional blocks of WiMAC. The WiMAC enables support for both LTE as well as 802.11. At this point, since we are only concerned with implementing the system over Wi-Fi, we restrict ourselves to the 802.11 application framework.

The framework consists of functional elements of the PHY and MAC layers (as illustrated in Figure 3.1) of a single station implemented using LabVIEW Communications. It runs on the Xilinx Kintex-7 FPGA (USRP) and Intel x64 general-purpose processor (Host), which are integrated with the RF and analog front ends of the NI software defined radio (SDR) hardware.

The framework supports majority of the PHY and MAC functionalities from the 802.11a and 802.11ac with compliance to the 802.11 standards.

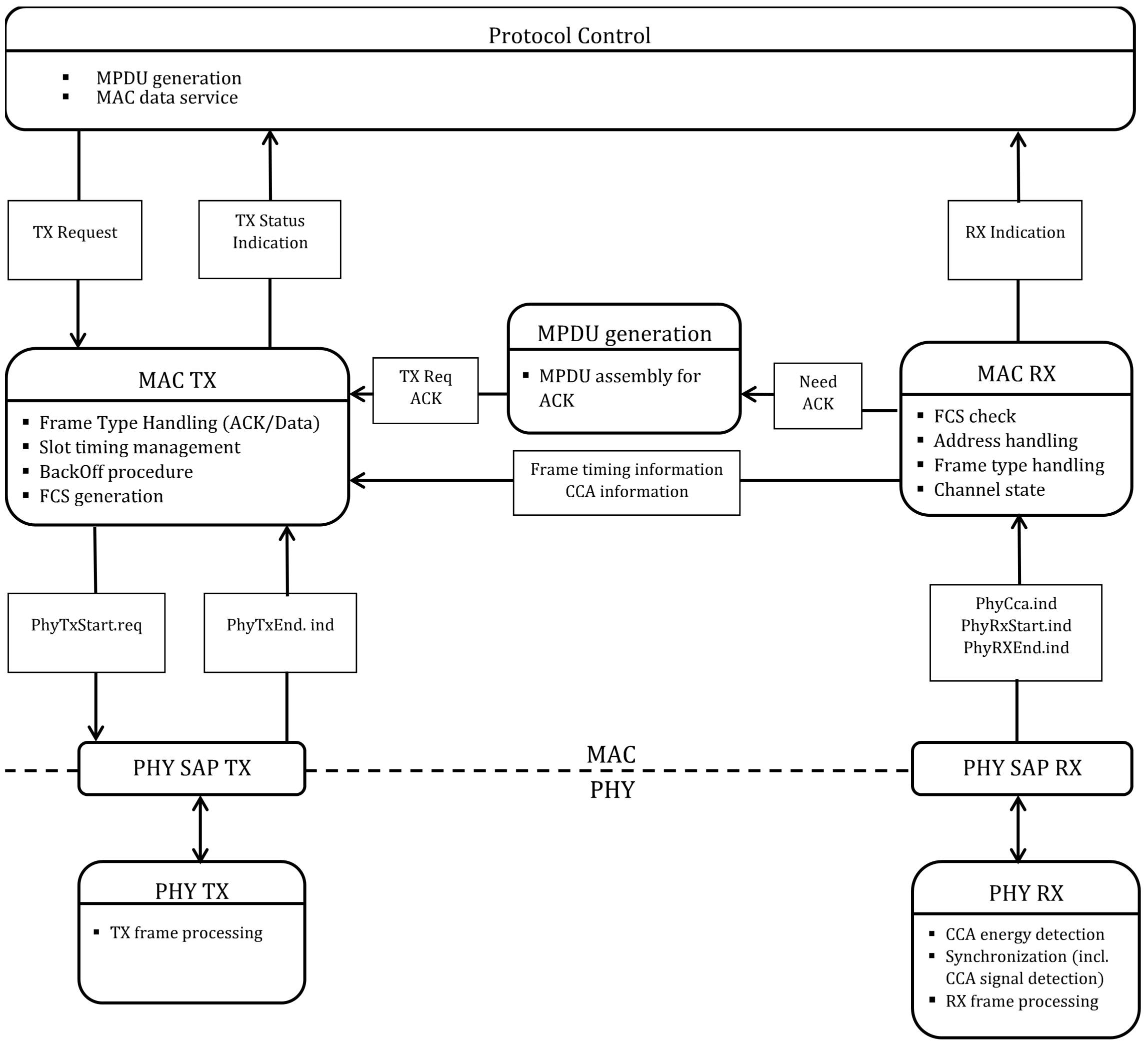

Figure 3.1: Interoperability Between MAC and PHY Blocks in the Application Framework

3.3.1 Functional Split - Host vs. Target

The functionality of the framework is split between the Host (software) and the USRP Target (hardware) keenly considering their salient features. This is depicted in Figure 3.2. Software allows for flexibility of design, accommodates incorporation of a wider range of algorithms owing to larger memory and reduces the prototyping time itself. Hardware helps meet the quick response constraints imposed by the MAC protocols.

The USRP target is hence associated with the lower MAC and baseband PHY layer algorithms as well as control capabilities for the RF front-end.

The host on the other hand, encompasses upper MAC layer functionality including protocol control.

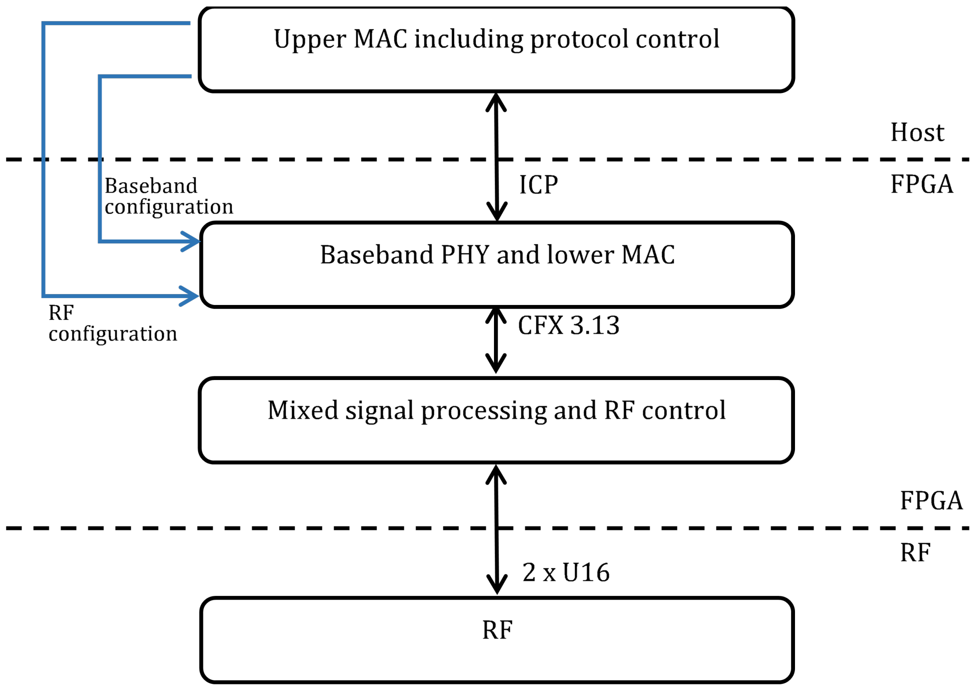
Upper MAC including protocol control
Host
FPGA
ICP
Baseband configuration
Baseband PHY and lower MAC
RF configuration
CFX 3.13
Mixed signal processing and RF control
FPGA
RF
2 x U16
RF

Figure 3.2: Functional Split: Host-FPGA-RF

The data path between the host and the USRP target is realized using a message-based interface communication protocol (ICP), which is described in Section 3.3.3. The control path between the host and USRP target is implemented using LabVIEW controls and indicators. The baseband RF configuration is achieved using target-specific driver VIs.

While the framework achieves the decoupling between hardware and software, it does not compromise on the cross-layer design aspect - which is crucial for better performance.

3.3.2 Host Architecture

We now take a closer look at how the Host deals with upper MAC layer control.

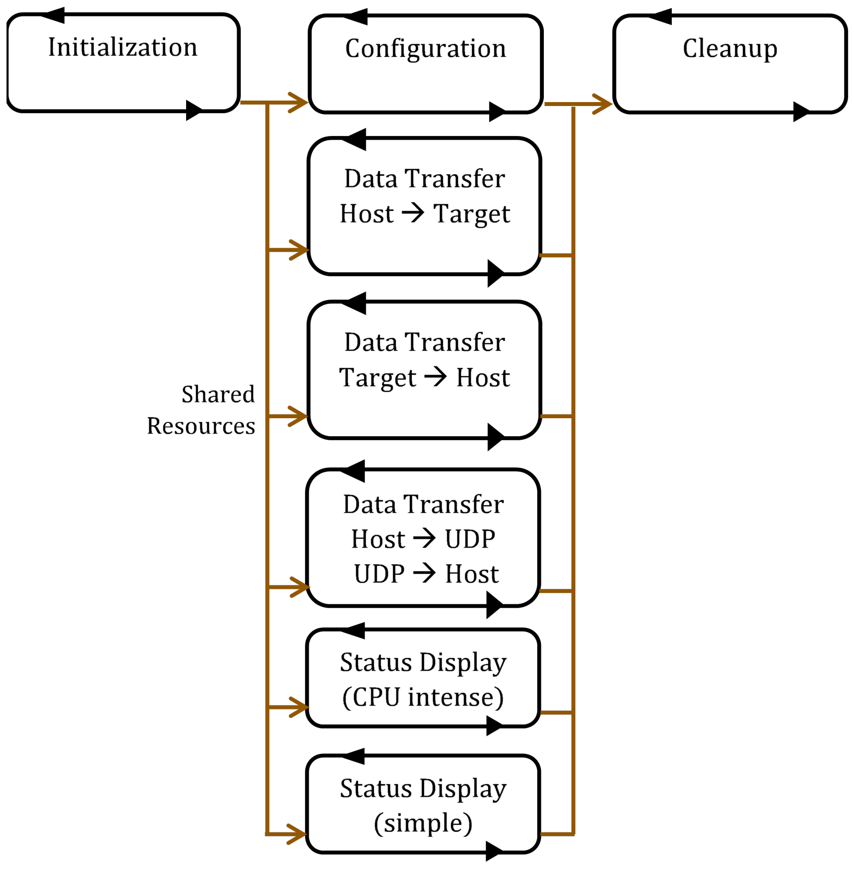

Figure 3.3: Schematic Showing the Host Architecture

As shown in Figure 3.3, the host is composed of six concurrent threads which together cover the tasks of configuration, data exchange, and status display.

The system status is globally shared using a session cluster that stores handles to devices and host-based queues. The system state is constantly updated by the parallel procedures, each of which modify the state information using the handles obtained from the session cluster.

3.3.3 Interprocess Communication Protocol (ICP)

ICP is used to transfer data between the host and target. While the structure itself is fixed, the length of the messages can be varied to be sent over a channel with a bytestream transmission capability, such as the DMA FIFOs. We modify the payload portion of this protocol to include additional fields that aid our development process.

The ICP packet format has two variants - one to transfer data to Target TX (Figure 3.4) and the second to receive data from the Target RX (Figure 3.5).

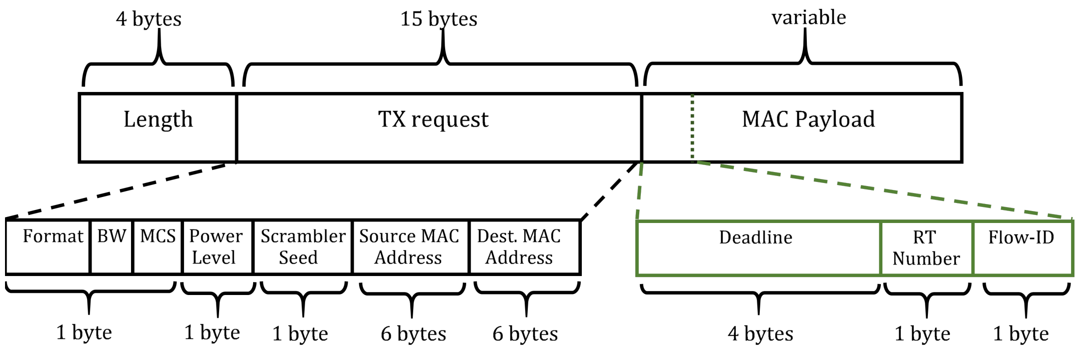

Figure 3.4: ICP TX Packet Format

The ICP TX structure serializes the TX Request into the header which is followed by the MAC payload. The payload is prefixed with custom fields as illustrated. The first 4 bytes represent the deadline of the packet which is followed by 1 byte of Re-Transmission Number (to limit the Re-transmissions as need be) and finally tailed by 1 byte of flow-ID. Each of these fields are prefixed when the packet arrives via the UDP receiver port. All except the flow-ID are detached before the packet is transferred from Host to the USRP Target. The flow-ID is a field that will be critical to identify the nature of flow at the receiver (and thereby forward to the corresponding application).

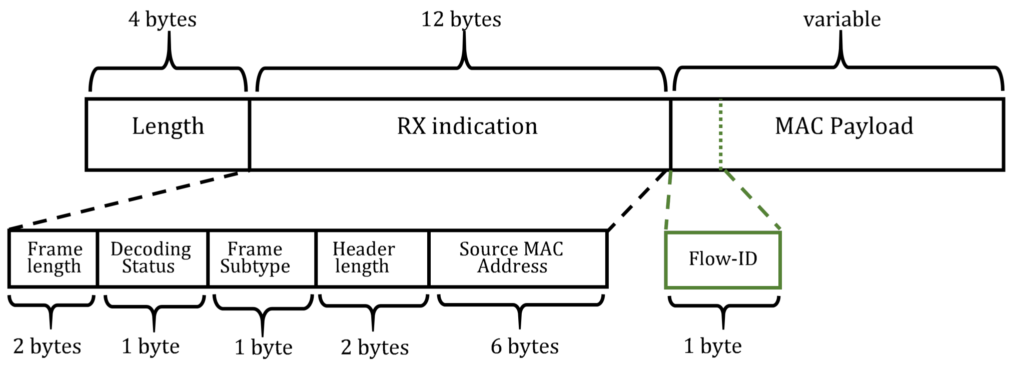

Figure 3.5: ICP RX Packet Format

The ICP RX structure serializes the RX indication which is followed by the MAC payload (MPDU without header information). The first byte of the MAC payload is the flow-ID which is prefixed at the transmitter. Based on this, the Receiver-Host will map the arriving flow to the corresponding application.

4. A SOFTWARE DEFINED CONTROL APPROACH ON WIMAC

4.1 Implementation Architecture

The implementation architecture is represented in Figure 4.1:

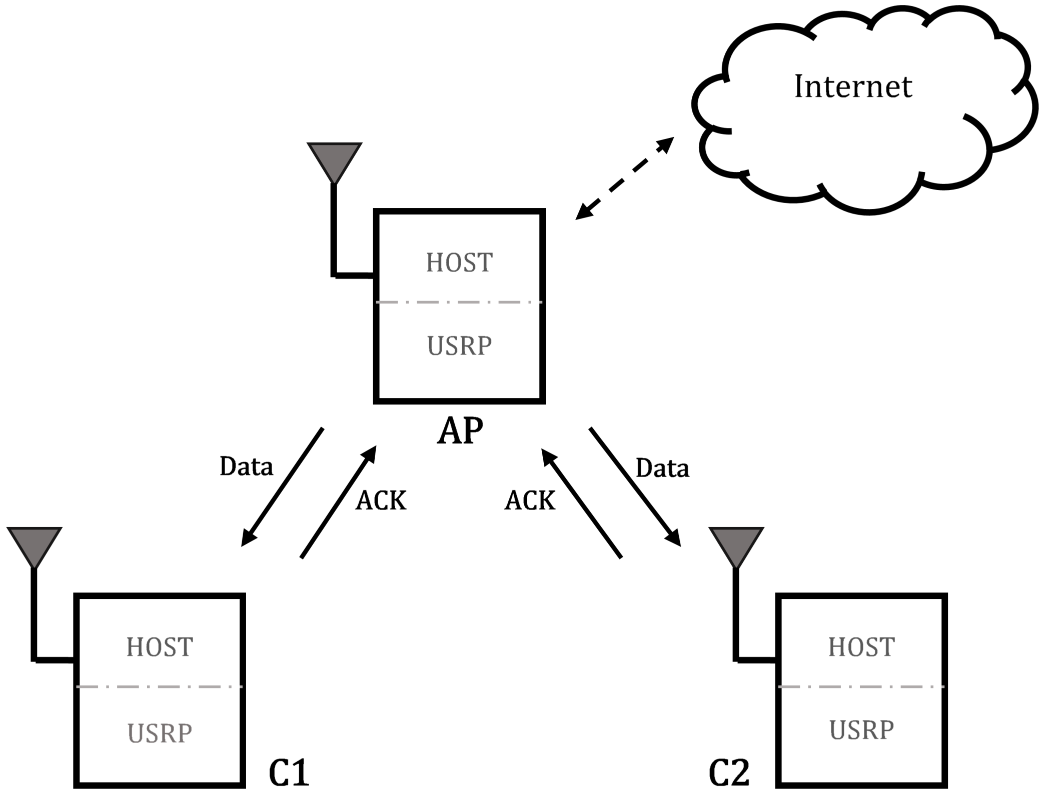

Figure 4.1: Implementation Architecture

Each station of this network is a WiMAC node (picture depicted in Figure 4.2), which consists of 2 parts - the Host (which runs LabVIEW on Windows) and the USRP (Universal Software Radio Peripheral by National Instruments). The WiMAC has the capability to talk to any host-based application via UDP.

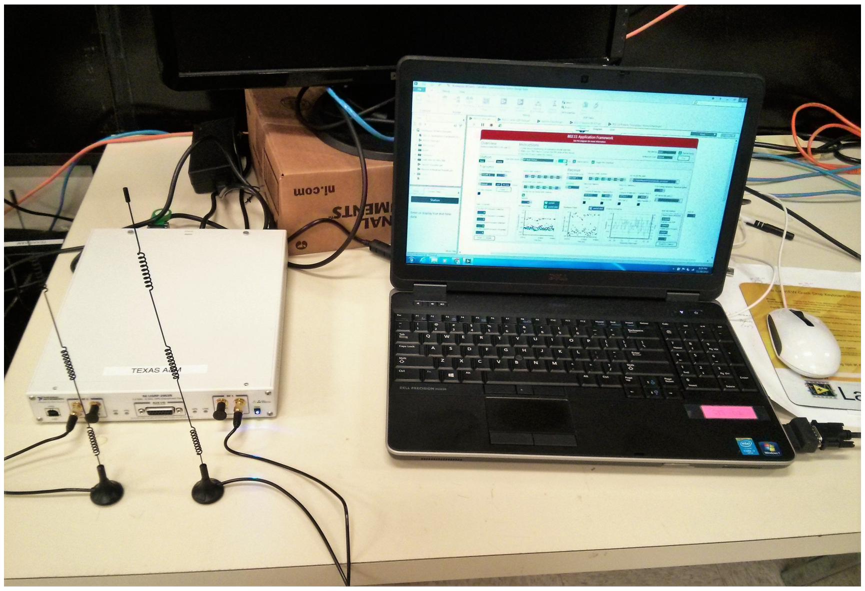

Figure 4.2: A WiMAC Station with a Host (right) and USRP (left)

The flows originate from (applications on) the AP and are intended for the (corresponding peer applications on) clients.

The AP-Host is the gateway to the Internet and could also be potentially connected to other networks via its Ethernet/Wi-Fi interface chipsets.

We choose a VLC video streaming application at the AP-Host to be the source of real-time flows, while a simple UDP file transfer application is the source of non real-time flows.

At each Client-Host we have a VLC receiver application as well as a UDP File receiver application to obtain the traffic generated at the Host. Packets received are appropriately forwarded to the respective application based on the flowID that we prefix as part of the header.

As introduced in Section 1, we implement mechanisms on the WiMAC and policies using OpenFlow.

4.2 Mechanism Implementation on WiMAC

Taking another look at the high-level overview of the system at the AP depicted in Figure 1.1, we observe that the mechanism implementation is done at three levels to enable reconfigurability. We now take a close look mechanism implementation at each of these levels on the WiMAC.

4.2.1 Reconfigurable MAC

Deficit based queuing has been proved to be optimal for delay sensitive traffic. This mechanism will be applicable for the inelastic real-time flows.

Packet scheduling using the Max Weight Algorithm has been established to be throughput optimal. This mechanism will be applicable for the elastic non real-time flows.

The solution to the problem stated in Section 2 would result in a joint congestion control and scheduling algorithm which equitably allocates spectrum resources to achieve the fairness objectives of both elastic and inelastic flows [38]. We find the capacity of our system at first and allocate total resources to about 80% of the capacity.

Based on this solution, at the beginning of each interval we give strict priority to the inelastic flows over elastic flows and make sure that the inelastic flows are rate controlled to prevent starvation of the elastic flows. Once all the inelastic packets that arrived at the frame boundary are transmitted, we proceed to schedule the elastic flows.

4.2.1.1 Deficit Based Queues - Implementation

Packet arrivals are uniform across both flows and occur at frame boundaries (at the beginning of each interval). For each inelastic flow, as packets arrive, the deadlines are prefixed as part of the header. Packets remain in the queue for as long as they are valid.

Since the deadline equals size of the interval, we guarantee that packets coming in at the beginning of the interval are transmitted by the end of the interval (as long as the packet arrivals are within the capacity region).

Each of the flows has a timely throughput requirement imposed on it by its intended client. This requirement is termed as the Delivery Ratio (q_l) for that flow. The Delivery Ratio is an indicator of how much to penalize the flow in case of packet expiry. The cumulative penalty suffered by each flow is indicated by its deficit field.

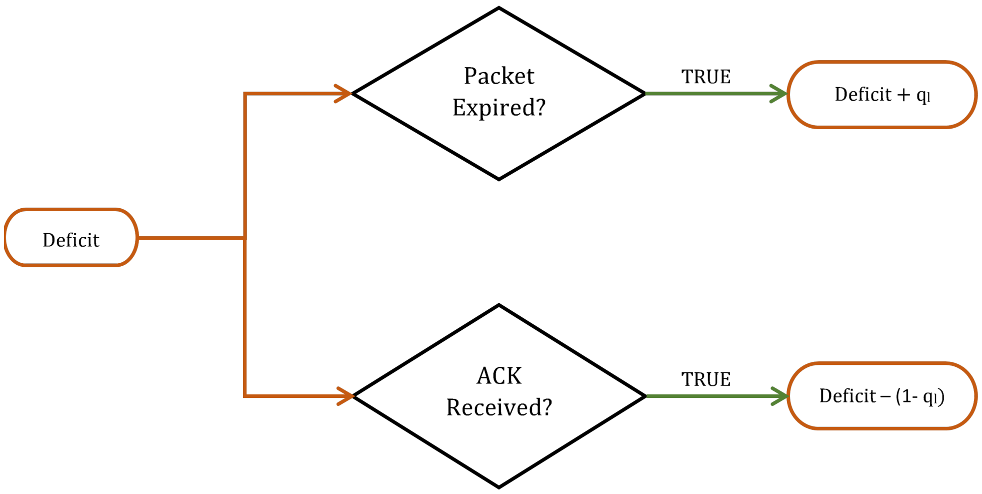

Figure 4.3: Deficit Update

The updating of deficit is illustrated in Figure 4.3. When a packet's deadline expires we increment deficit of the corresponding flow by q_l and drop the packet. When the packet is successfully transmitted (i.e upon reception of ACK) we decrease deficit by $(1 - q_l)$. This is consistent with the weighted delivery deficit described in [19].

As depicted in Figure 4.4, at each slot, we give priority to the flow with more deficit in order to equalize the penalties. Before transmission, we check the packet deadline and transmit/drop the packet as the case maybe. Simultaneously, we poll the Head-Of-Line packet in the unscheduled queue and drop it on expiry.

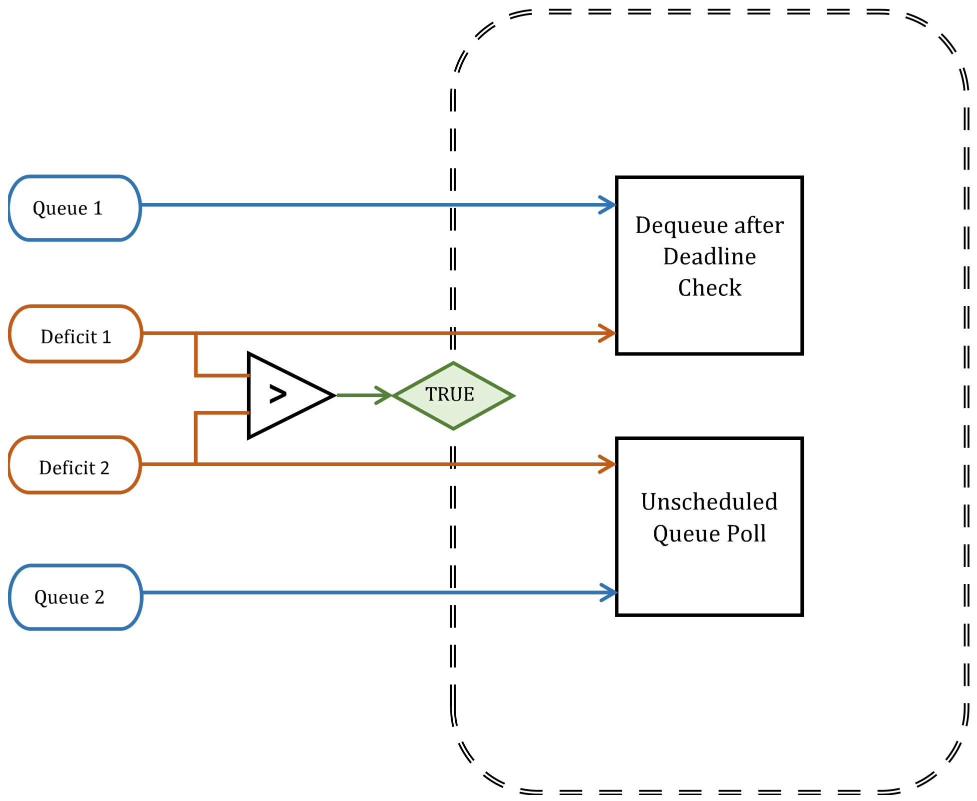

Figure 4.4: Deficit Based Scheduling

If the transmission of a packet isn't successful (indicated by ACK timeout), we retransmit the packet as long as the packet is still alive. A point worth noting here is that we don't increment the deficit in this case i.e., we don't penalize for unsuccessful/lossy transmissions.

As long as the packet arrivals are within the capacity region the deficits are always bounded and the system is stable.

4.2.1.2 Longest Queue First - Implementation

In the case of elastic flows, the scheduling is much more straightforward since there are no delay constraints imposed. At each slot, we check if the Deficit Based Queues are empty (zero packets in each of them). If yes, we schedule the elastic flow with the longest queue length as shown in Figure 4.5.

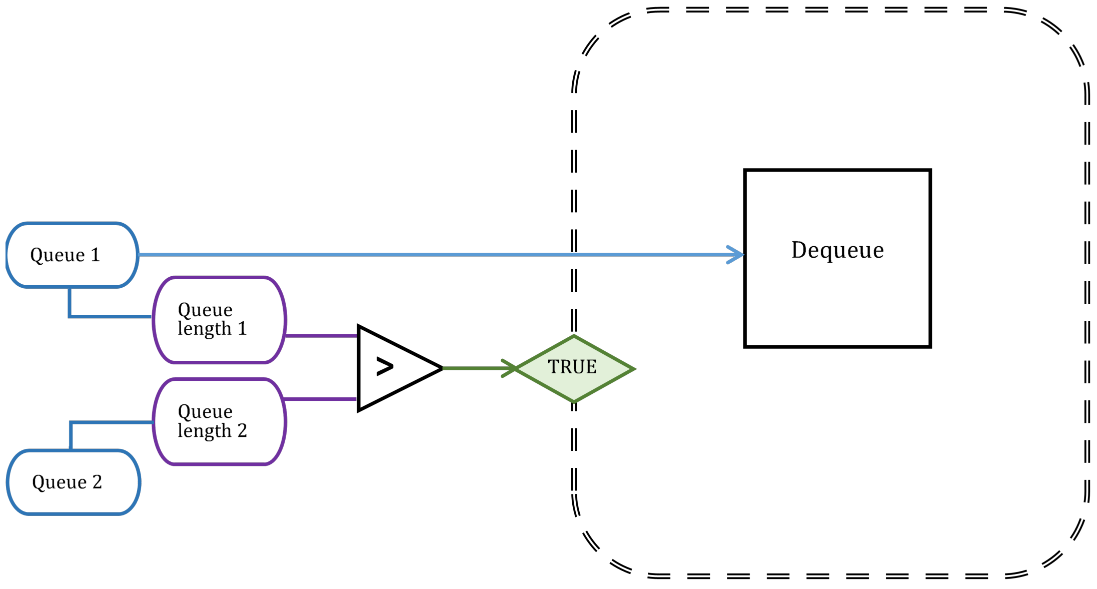

Figure 4.5: Longest Queue First Scheduling

We re-transmit a fixed number of times since the elastic packets don't have deadlines associated with them. This will prevent a single elastic flow monopolizing the resources of the entire system when its corresponding channel conditions happen to be poor.

4.2.2 Reconfigurable PHY

Since the clients themselves are located at different spots, their individual channel conditions could well differ. The Channel State of a client can be quantified by observing its block error rate - or essentially the number of successful transmissions (denoted by ACKs). This Channel State Information (CSI) is an indicator of the channel quality. A higher quality channel is to be given more precedence to minimize resource wastage as well as improve the overall system performance.

In order to make sure that the spectrum is well utilized, we use the CSI of each client to:

1. Adjust its modulation scheme

2. Weight its scheduling parameters

The CSI updating itself is done using an exponentially weight moving average of the following form:

$$CSI_{upto\ [T]} \quad = \quad (1-\beta)CSI_{upto\ [T-1]} + (\beta)CSI_{at\ [T]}$$

Where $CSI_{at\ [T]} = \mathbb{1}_{ACK}$ and β is pruned to suit our scenario. Based on the range of values in which the CSI lies, we use Table 4.1 to decide which modulation scheme to use. Since the CSI values typically don't vary rapidly, the updated modulation scheme will either be a step-up or a step-down from the one deployed previously.

$CSI_{upto\ [T]}$	Modulation Scheme (MCS)
0-0.1	BPSK (3/4)
0.1-0.2	QPSK (1/2)
0.2-0.3	QPSK (3/4)
0.3-0.7	16 QAM (1/2)
0.7-0.8	16 QAM (3/4)
0.8-0.9	64 QAM (2/3)
0.9-1	64 QAM (3/4)

Table 4.1: CSI Ranges to MCS Look-up.

Using the updated modulation scheme and a lookup table (shown in Table 4.2), we weight the scheduling parameters (deficits or queue lengths) accordingly. This is consistent with the $(1/p_n)$ described in [19].

Modulation Scheme (MCS)	Weight
BPSK (3/4)	1
QPSK (1/2)	1.3
QPSK (3/4)	1.8
16 QAM (1/2)	2.3
16 QAM (3/4)	3
64 QAM (2/3)	3.6
64 QAM (3/4)	3.9

Table 4.2: MCS to Weight Look-up.

By choosing a modulation scheme, the net maximum throughput values observed are correspondingly recorded (under nearly perfect channel conditions in an anechoic chamber). These throughput values are then normalized with respect to the BPSK base case to express them as weights in the look-up table.

4.2.3 Reconfigurable Antennas

The WiMAC's antenna setup can be controlled via an external unit that permits configuration of antenna parameters. A picture of the Antenna setup is shown in Figure 4.6. The antenna reconfiguration is done over large timescales at the aggregates of packets.

The antennas used have two reconfiguration states viz. Vertical and Horizontal polarization. These states are controlled by an Arduino microcontroller which in turn is serially controlled by a local computer (typically an Intel NUC). More specifically, applying a bias to the diode which in the datapath to the antennas, the polarization states can be altered in the timescale of microseconds.

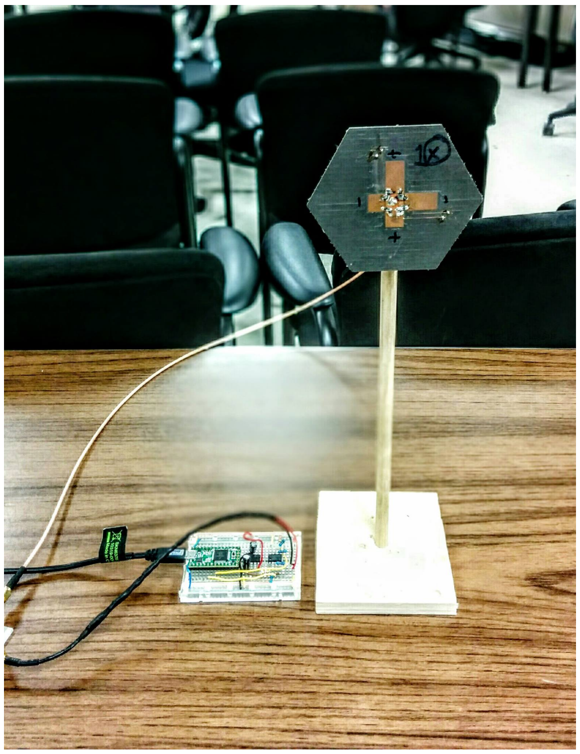
Figure 4.6: The Reconfigurable Antenna Setup

4.3 Policy Implementation using OpenFlow

We now extend this setup to make the AP OpenFlow compatible to execute specific policy schemes operating over an aggregate of packets.

We utilize OpenFlow's Experimenter messages feature which enables the exchange of user-defined messages between an SDN Controller and Device. The nature of the message is specified using an Experimenter type included as part of header. This field allows us to define different categories of custom commands/responses.

The Controller packs the command (in pre-decided format), creates an experimenter message and sends it to the Device. The Device unpacks the message, performs the appropriate operation (based on experimenter type) and sends the results obtained back to the Controller (as an experimenter message).

We deploy a RYU SDN Controller which communicates to the Softswitch (SDN Receiver/Device) via OpenFlow. The Softswitch in turn relays these messages to the AP Host via UDP send and receive. The remote controller will now gain the ability to configure policies (real-time) on the following components of the AP:

1. Reconfigurable MAC: MAC mechanisms and their configurations can be chosen based on a dynamically operating bidding system. Policies would include:
   - Determining which flow should be coupled with what mechanism (real-time or throughput maximizing).
   - Assigning appropriate Delivery Ratios to the real-time flows of clients.
   - Proportionally allocate the spectrum resources between the two types of flows to maximize utility.
2. Reconfigurable PHY: USRP Radio parameters (TX power level, constellation etc.) based on channel conditions.

3. Reconfigurable Antennas: The WiMAC's antenna setup via an external antenna control unit that permits configuration of the radiation pattern of the antennas.

5. RESULTS

The primary focus of this work is to implement custom MAC mechanisms on a wireless testbed to facilitate quick prototyping. Though a combination of real-time and non real-time flows has never been implemented, elastic flows alone have been implemented in [14]. While these mechanisms are very diverse, WiMAC makes the experimentation easily performable through the decoupling framework.

5.1 Implementation Specifications

For the experiments, we use NI USRP 2153R, which is an FPGA-based SDR by National Instruments. The experiments are run at the center frequency of 2.5 GHz. Subcarrier format follows the IEEE 802.11a 20 MHz standards. To reduce interference from external sources, we manually set the transmit power to be 10 dBm as well as perform our experiments in an anechoic chamber. Figure 5.3 shows a line diagram of the experimental setup while Figures 5.1 and 5.2 show snapshots of the setup from within the chamber.

We take the slot length to be the time taken by BPSK to transmit a packet including all overheads. We have determined the total capacity of the system to be of the order of 16 Mbps and we operate at around 13 Mbps. We choose interval width to be 50 ms and regulate inelastic packet arrivals to only at the beginning of each interval for both the clients. Elastic packets are non-deterministic and follow poisson arrivals at each interval with rate λ. We have 30 packets of inelastic flows and $\lambda = 10$ packets per interval for elastic flows arriving for Client 1. To induce assortment of flows, for Client 2 we have 10 packets of inelastic flows and $\lambda = 30$ packets per interval for elastic flows. Each packet is of the order of 1400 bytes. We specify the Delivery Ratio of Client 1 to be 0.95 and Client 2 to be 0.85.

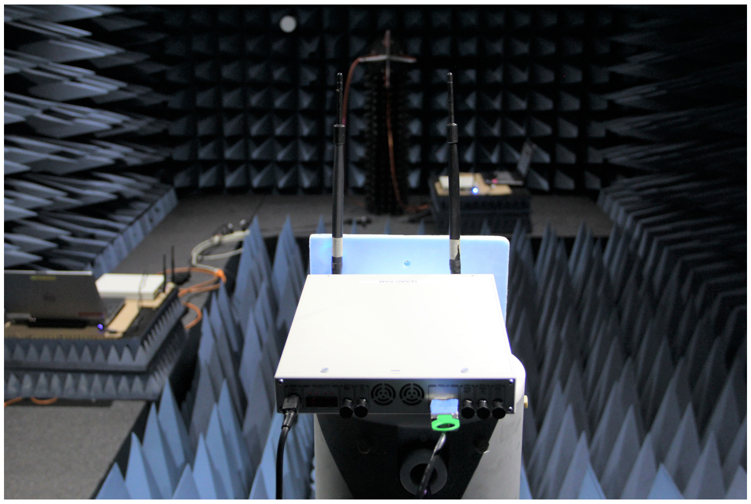

Figure 5.1: Experimental Setup - View from AP

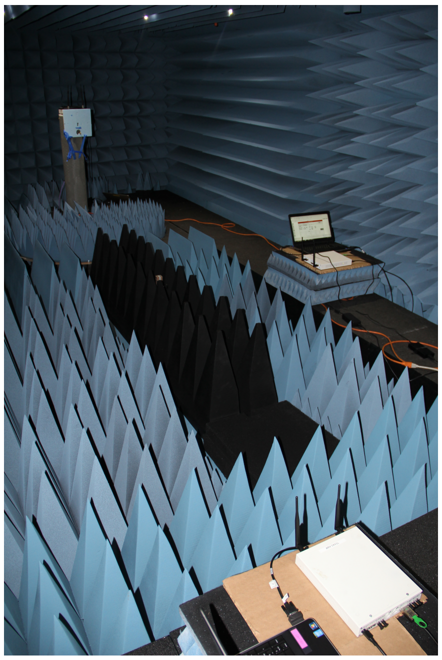

Figure 5.2: Experimental Setup - View from Client 2

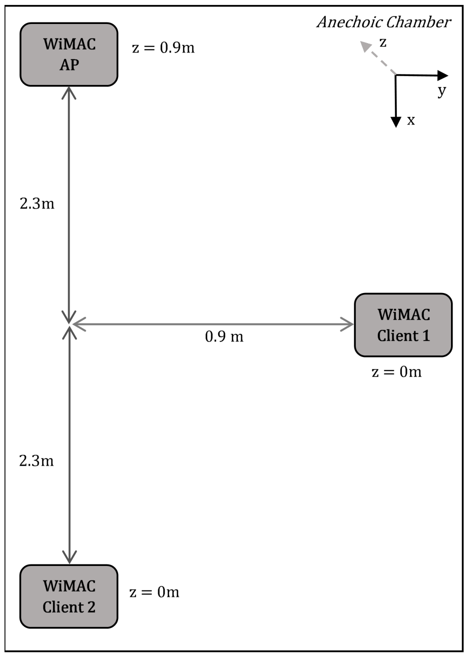

Figure 5.3: Experimental Setup - Line Diagram

5.2 Reconfigurable MAC

We have implemented the wireless system with the architecture illustrated in Figure 4.1. This incorporates real-time flows scheduled by Deficit-Based-Queuing and non real-time flows scheduled by Longest-Queue-First.

5.2.1 Experimental Evaluation of the Optimal MAC System

For the first set of results, we expose both clients to similar channel conditions. As is expected, we get a throughput of about 6.5 Mbps at each Client. Since both channels are good, we do not observe packet loss. Since we are operating within the capacity region, the system is stable, deficits for the inelastic flows are almost always at 0 and the queue lengths for the elastic flows maintain their state and don't build up to enormous amounts.

Since we use a VLC streaming application to be the source of real-time traffic, we observe the corresponding video feed at each Client-Host with a slight delay. The file transfer application too progresses steadily.

Next, we compare the performance of our optimal mechanism with other based on the metrics explained in the next section.

5.2.2 Performance Comparison with Existing MAC Algorithms

For comparison purposes, the two key mechanisms discussed in Section 2 are chosen to be benchmarks.

We implement the following MAC mechanisms in our current system of elastic and inelastic flows:

1. Utility maximization using Max-Weight and Deficit Based (proposed & optimal)

2. Max Weight Scheduling for all queues

3. Deficit Based Queueing for all queues

4. Randomized Scheduler

Metrics to gauge the performance of each mechanism would be:

- RX Throughput: average number of packets successfully received at the client measured in bits per second.

- Queue lengths: Of elastic flow measured each time the state of the system changes.

- Attained Delivery Ratio: Of both elastic and inelastic flows for each client.

  The attained delivery ratio for inelastic flows signifies the fraction of packets satisfying the delay requirement. This can be quantified as (1 - Drop probability) where the Drop probability is defined as $a_{i-exp}/(a_i + a_e)$ where a_{i-exp} refers to the packets expired in the interval and $a_i + a_e$ refers to the total packets that arrived at the beginning of the interval.

- Queue Deficits: Of inelastic flow measured each time the state of the system changes.

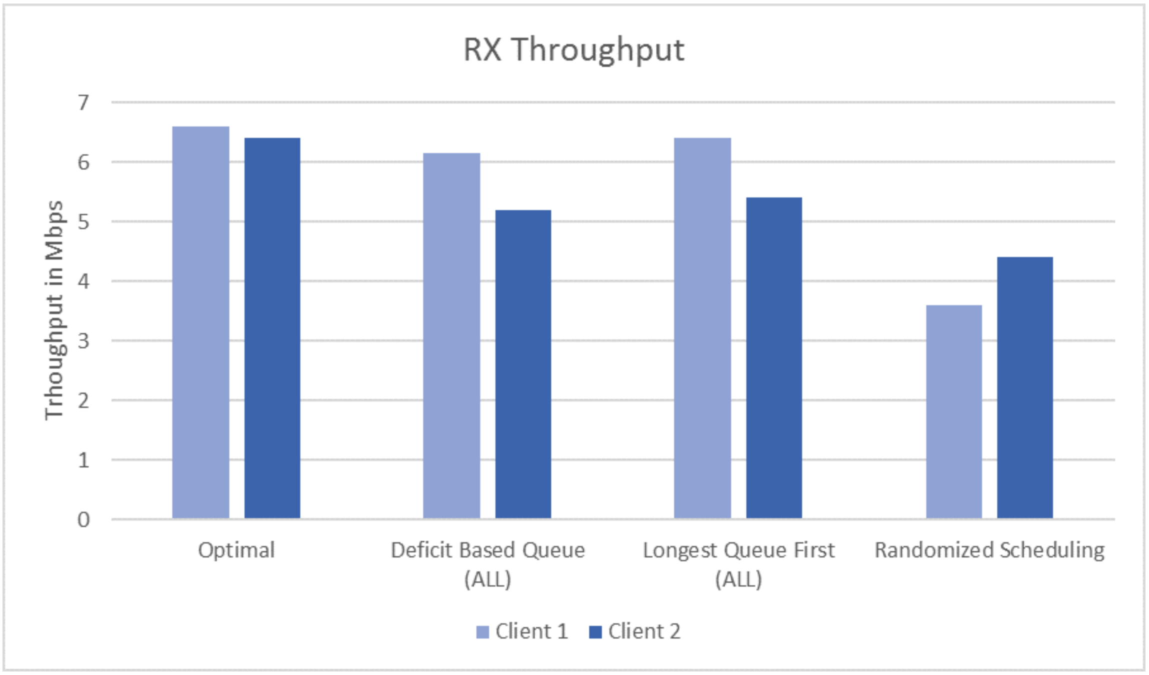

Figure 5.4: RX Throughput Comparison

Figure 5.4 shows the average throughput achieved by each of the MAC mechanisms for Clients 1 and 2. Client 2 has a higher throughput than Client 1 owing to the greater Delivery Ratio assigned to it. Evidently, our optimal mechanism performs much better than the others by delivering a net throughput of above 13 Mbps. The Deficit Based Queues and Longest Queue First only manage to reach around 11-12 Mbps.

A good throughput is essential to sustain a system of both elastic and inelastic flows. More so when the inelastic flow consists of high definition video packets.

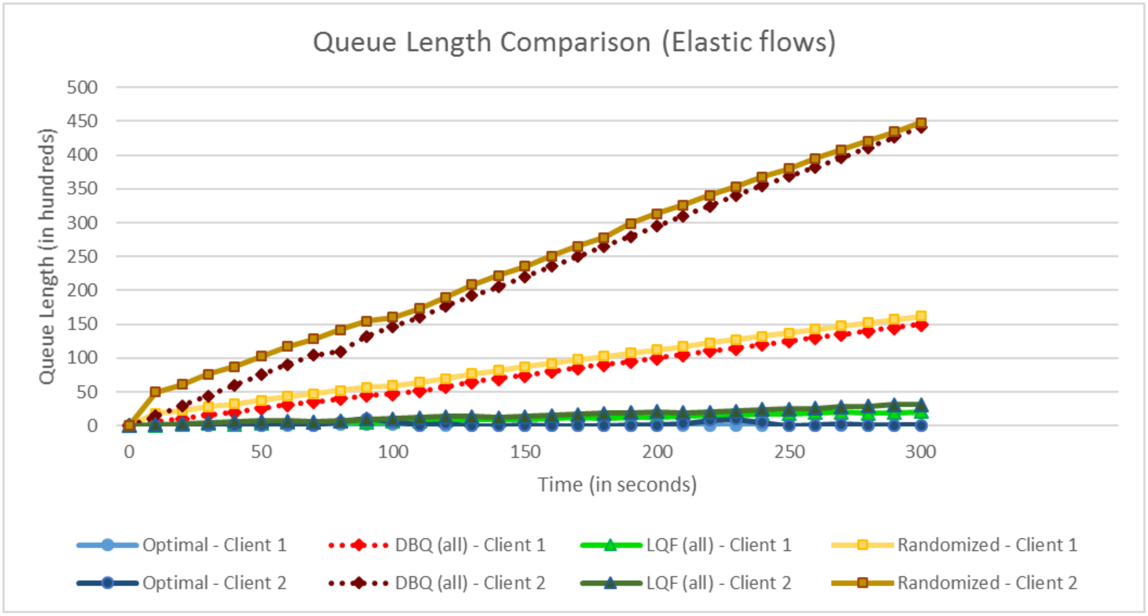

Figure 5.5: Queue Length Comparison of Elastic Flows

Next in Figure 5.5, we take a look at the queue lengths of the elastic flows of both the clients as time progresses. While the Optimal mechanism maintains queue lengths at the order of less than 100 packets for each client, the Longest Queue First mechanism is less successful since it also gives equal weight to the queue lengths of inelastic flows (which is unnecessary). Deficit Based Queueing clearly trails behind since it pays no heed to queue lengths while scheduling the flows.

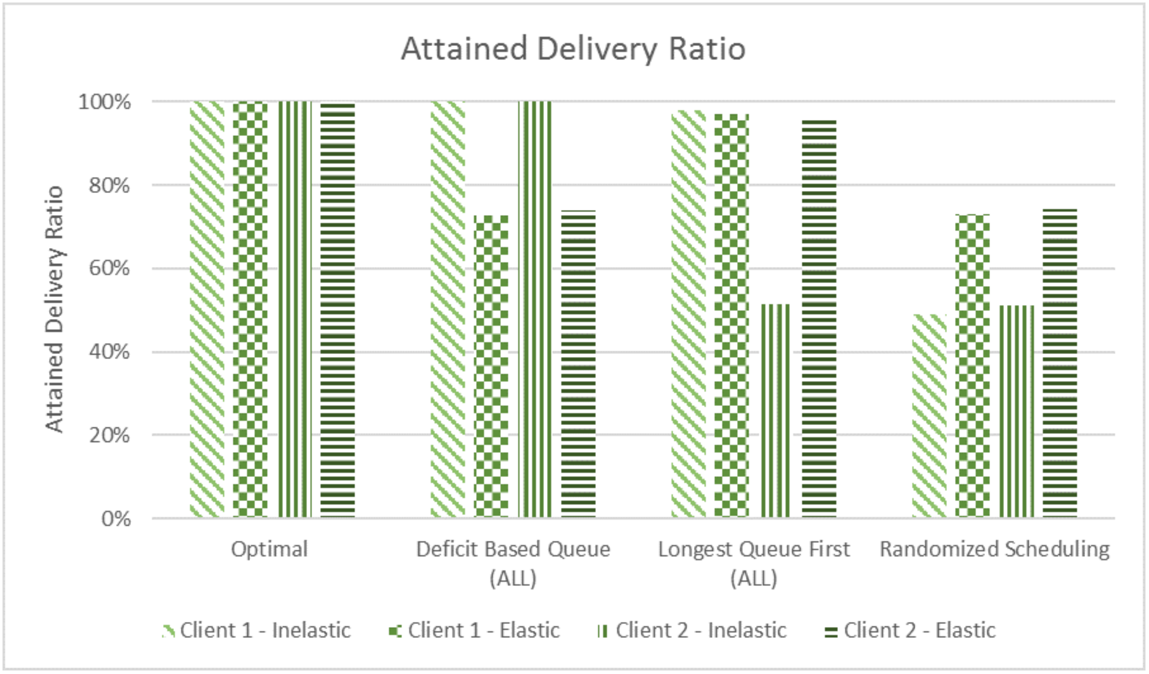

Figure 5.6: Attained Delivery Ratio Comparison

As illustrated in Figure 5.6, the Optimal mechanism achieves 100% of the delivery ratio for all four flows, which is more than the required numbers for the clients (85% and 95% respectively).

Although Deficit Based Queuing meets these numbers for inelastic flows, it only attains around 75% delivery ratio for the elastic flows. This is due to the randomized scheduling done between the elastic flows. The elastic packet arrivals per interval at the two clients differ hugely (10 for Client 1 vs. 30 for Client 2). Since randomization pays no heed to queue lengths, it schedules Client 1 (elastic) even after it runs dry.

Longest Queue First just manages to meet the optimal ratios for Client 1's flows as well as Client 2's elastic flow. It takes a performance hit for the inelastic flow of Client 2 owing to its queue length. Since inelastic packet arrivals are just 10 per interval for Client 2, its always given a lower scheduling priority which increases the chances of the head of line packets being dropped more often due to deadline expiry.

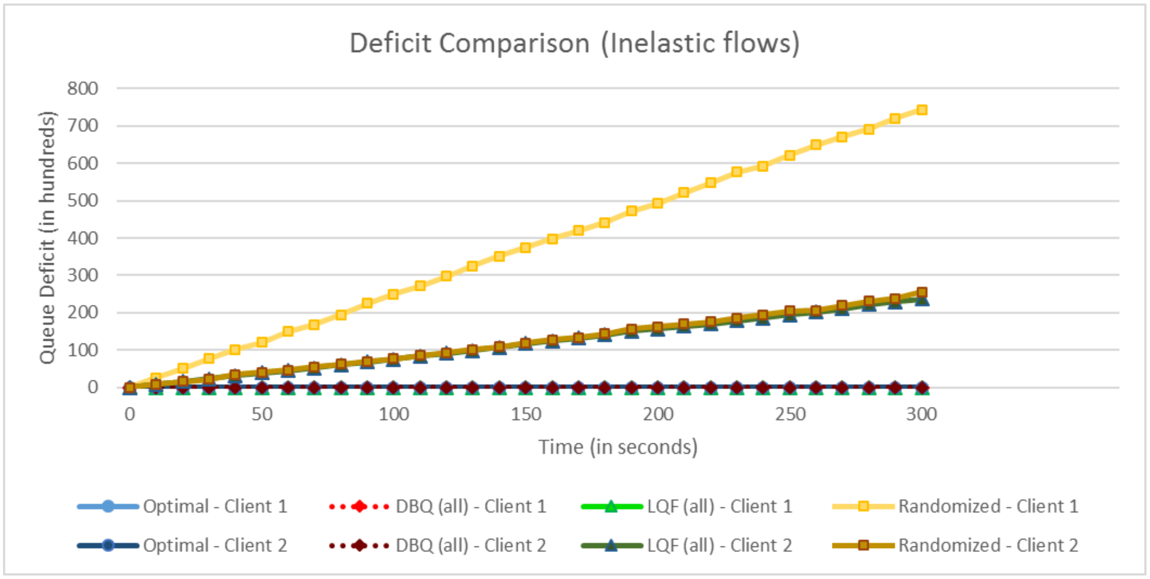

Figure 5.7: Deficits Comparison of Inelastic Flows

Finally in Figure 5.7, we compare the deficit numbers for the inelastic flows of each client as time progresses. Both the Optimal mechanism and Deficit Based Queueing have close to zero deficits for each inelastic flow.

While deploying Longest Queue First, the deficit numbers continue to increase slowly and are never stabilized unlike the above two mechanisms.

In all the above comparison metrics, the Randomized scheduler performs the worst since its neither aware of queue lengths nor of the presence of deadlines and delivery ratios to be met.

5.3 Reconfigurable PHY

Following the MAC, we focus on enabling reconfigurability at the PHY layer by analyzing the system performance under variable channel conditions. This is primarily to demonstrate that the incorporation of Channel State Indicators (with corresponding modifications to the weights and modulation scheme as discussed in the Section 4.2.2) results in a marked improvement in performance.

Client 1 is left as is, while Client 2 is faced with a much lower received signal strength and hence a poorer channel condition. This is achieved by decreasing the receiver antenna gain at Client 2 as well as decreasing the transmit power at the AP (for Client 2 in particular).

Without considering the channel state of the system - Client 2, (due to its static modulation scheme) not only performs poorly, but also uses up the spectrum resources that could have been put to better use if allocated to Client 1. The performance gain due to incorporation of the Channel State Indicators (CSI) is illustrated in the plots below (Figures 5.8 and 5.9).

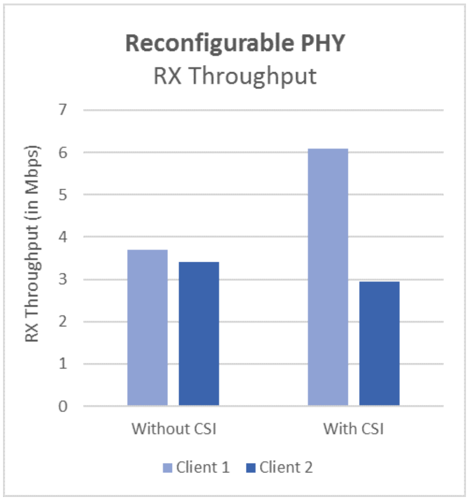

Figure 5.8: RX Throughput with Channel State Indication

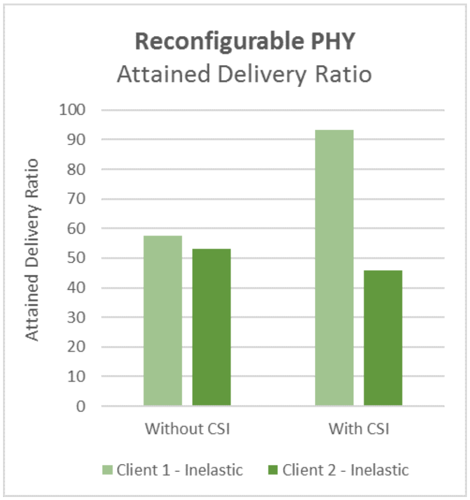

Figure 5.9: Attained Delivery Ratio with Channel State Indication

5.4 Reconfigurable Antennas

Next, we look at the system behavior with varying antenna configuration states. The AP and Client 1 maintain their current state of Vertical Polarization with the existing omnidirectional antenna.

Client 2 however is affixed with a re-configurable directional antenna with two polarization states:

1. Vertical (Co-polarization)

2. Horizontal (Cross-polarization)

The system performance due to varying antenna configuration states is illustrated in the plots below (Figures 5.10 and 5.11).

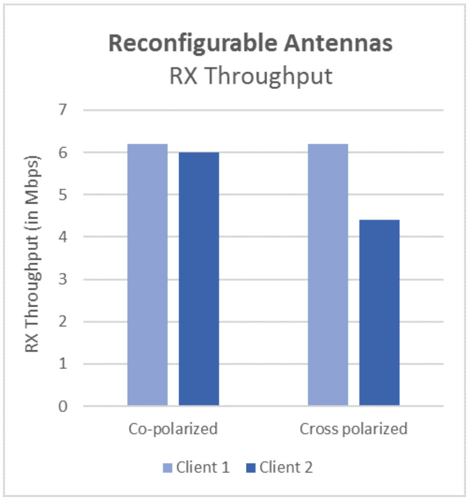

Figure 5.10: RX Throughput with Antenna Polarization

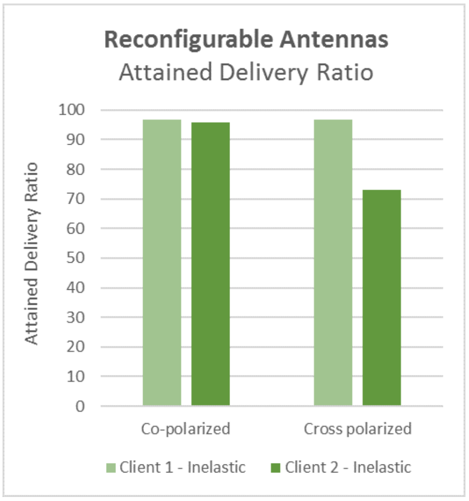

Figure 5.11: Attained Delivery Ratio with Antenna Polarization

5.5 Policy Implementation

We finally look at implementing policies on our testbed. We vary arrival rates (a_l) and delivery ratios (q_l) in the timescale of seconds to compare how the Optimal system performs as compared to the Randomized scheduler. The Randomized scheduler operates similar to an 802.11 based AP in the sense that all flows are treated identically irrespective of their nature.

In the following experiment, we change these parameters every 90 seconds. The variance with timeframes is depicted in Table 5.1.

Timeframe (sec) / Parameter	0 - 90	90 - 180	180 - 270	270 - 360
a_{i1}	35	30	40	25
$\lambda(a_{e1})$	10	10	20	5
a_{i2}	10	10	20	5
$\lambda(a_{e2})$	35	30	40	25
q_{l1}	0.95	0.9	0.9	0.85
q_{l2}	0.85	0.8	0.9	0.95

Table 5.1: Parameter Specifications at each Timeframe (in seconds)

The corresponding comparison results are illustrated as under (in Tables 5.2, 5.3 and Figures 5.12, 5.13). Despite the real-time variance and exceeding the capacity of the system (in timeframe 180 - 270), the optimal mechanism outperforms the randomized scheduler in the key aggregates quantifying the QoS.

Timeframe / MAC mechanism	0 - 90	90 - 180	180 - 270	270 - 360
Optimal - Client 1	6.6	5.8	6.6	4.9
Optimal - Client 2	6.6	5.8	6.6	4.9
Randomized - Client 1	3.6	3.2	3.9	2.4
Randomized - Client 2	4.4	3.9	4.6	2.9

Table 5.2: Throughput (in Mbps) Variance with Time

Timeframe / MAC mechanism	0 - 90	90 - 180	180 - 270	270 - 360
Optimal - Client 1	100	100	86.7	100
Optimal - Client 2	100	100	100	100
Randomized - Client 1	48.9	55.2	42.8	62.3
Randomized - Client 2	51.1	51.8	59.3	68.7

Table 5.3: Attained Delivery Ratio (in %) of Inelastic Flows with Time

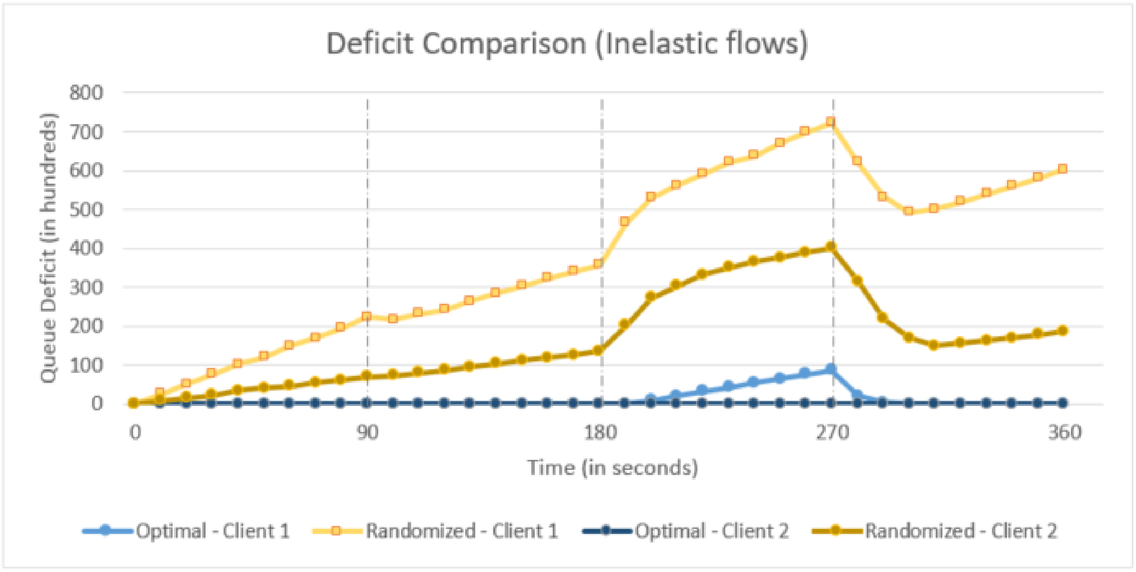

Figure 5.12: Deficits Comparison of Inelastic Flows with Time

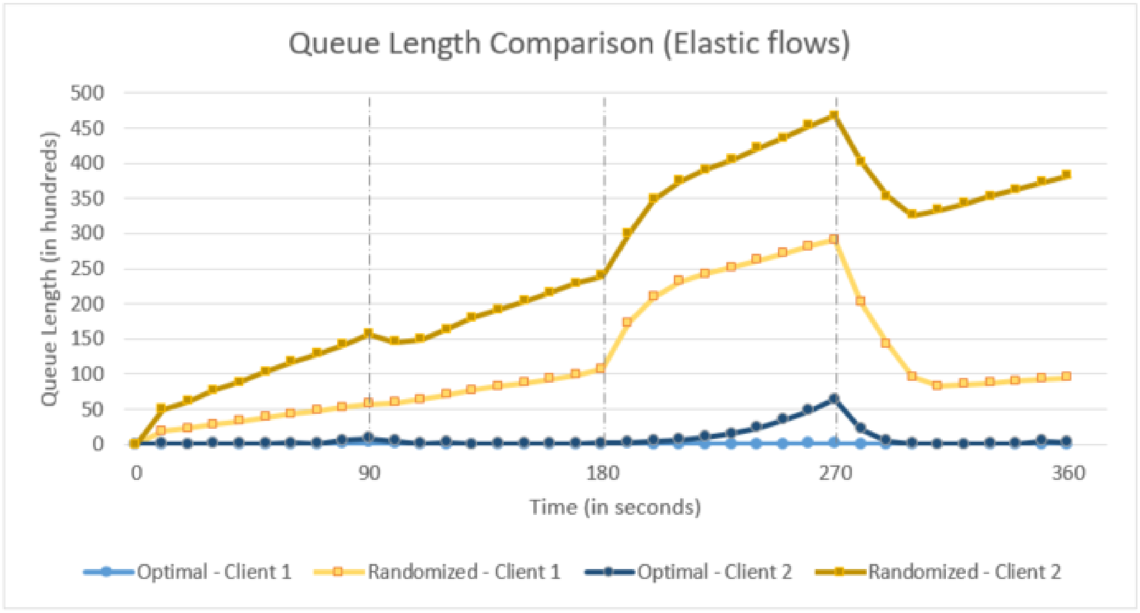

Figure 5.13: Queue Length Comparison of Elastic Flows with Time

6. CONCLUSION

Existing efforts towards MAC implementation do attain protocol-level adaptability but lack in terms of either implementation flexibility or prototyping agility. We remove the tight coupling between the MAC and PHY to achieve rapid MAC prototyping via the WiMAC testbed. We promote the design of MAC protocols in software thereby supporting dynamic protocol changes at runtime.

In this work, we developed the idea of mechanism-policy separation. The per-packet decisions (mechanisms) were implemented on the WiMAC while the larger timescale decisions (policies) were implemented using OpenFlow.

The implementation considers both best-effort and delay sensitive traffic and uses a combination of the appropriate protocols (Max-Weight and Deficit Based Queueing) to achieve optimality. The optimal protocol outperforms the other protocols in consideration with respect to the key aggregates that define the Quality-of-Service. Execution of policy schemes using a centralized controller also enables the system to achieve reconfigurability at multiple levels.

The current implementation focuses on a single-hop downlink scheduling based wireless network. It will be worthwhile to extend this to a multi-hop setup in order to analyze the dynamics which are less-deterministic. Another aspect that we intend to inspect in the near future would be enabling reconfiguration of radiation patterns using the antennas on a per-packet basis. This could be feasible with the control-functionality at the USRP-Target. Since the USRP-Target takes care of the actual packet transmissions, possible issues with respect to synchronization can be avoided.

In this work, we only consider some basic policy schemes to suit our implementation purposes. The idea of executing policy changes with a centralized controller opens a vast set of possibilities for exploration.

As demonstrated in our results, the combination of mechanisms (at the PHY-MAC levels) with the given set of policy schemes is what results in a QoS aggregate. It is however not clear as to what the optimal combinations are when the flows are more varied and heterogeneous. Hence, coming up with an intelligent model to sample the space of mechanism-combinations and their corresponding QoS aggregates and thereby resolving the optimal combination is also an interesting direction worth investigating.

REFERENCES

[1] S. Yau, L. Ge, P.-C. Hsieh, I. Hou, S. Cui, P. Kumar, A. Ekbal, N. Kundargi, *et al.*, “WiMAC: Rapid Implementation Platform for User Definable MAC Protocols Through Separation,” in *ACM SIGCOMM Computer Communication Review*, vol. 45, pp. 109–110, ACM, 2015.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling innovation in campus networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[3] L. Tassiulas and A. Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks,” *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[4] M. J. Neely, E. Modiano, and C. E. Rohrs, “Dynamic power allocation and routing for time-varying wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 89–103, 2005.

[5] X. Lin, N. B. Shroff, and R. Srikant, “A tutorial on cross-layer optimization in wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1452–1463, 2006.

[6] A. Eryilmaz and R. Srikant, “Joint congestion control, routing, and MAC for stability and fairness in wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1514–1524, 2006.

[7] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource allocation and cross-layer control in wireless networks.* Now Publishers Inc, 2006.

[8] A. Ganti, E. Modiano, and J. N. Tsitsiklis, "Optimal transmission scheduling in symmetric communication models with intermittent connectivity," *IEEE Transactions on Information Theory*, vol. 53, no. 3, pp. 998–1008, 2007.

[9] M. J. Neely, "Order optimal delay for opportunistic scheduling in multi-user wireless uplinks and downlinks," *IEEE/ACM Transactions on Networking (TON)*, vol. 16, no. 5, pp. 1188–1199, 2008.

[10] J. Ni and R. Srikant, "Distributed CSMA/CA algorithms for achieving maximum throughput in wireless networks," in *Information Theory and Applications Workshop*, Citeseer, 2009.

[11] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 18, no. 3, pp. 960–972, 2010.

[12] J. Liu, Y. Yi, A. Proutiere, M. Chiang, and H. V. Poor, "Maximizing utility via random access without message passing," *Mountain View, CA, Tech. Rep*, vol. 128, 2008.

[13] J. Ni, B. Tan, and R. Srikant, "Q-CSMA: Queue-length-based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 825–836, 2012.

[14] A. Sridharan, S. Moeller, and B. Krishnamachari, "Implementing backpressure-based rate control in wireless networks," in *Information Theory and Applications Workshop, 2009*, pp. 341–345, IEEE, 2009.

[15] J. Lee, J. Lee, Y. Yi, S. Chong, A. Proutière, and M. Chiang, “Implementing utility-optimal CSMA,” in *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pp. 102–111, IEEE, 2009.

[16] B. Nardelli, J. Lee, K. Lee, Y. Yi, S. Chong, E. W. Knightly, and M. Chiang, “Experimental evaluation of optimal CSMA,” in *INFOCOM, 2011 Proceedings IEEE*, pp. 1188–1196, IEEE, 2011.

[17] T. Stockhammer, H. Jenkac, and G. Kuhn, “Streaming video over variable bit-rate wireless channels,” *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 268–277, 2004.

[18] S. H. Kang and A. Zakhor, “Packet scheduling algorithm for wireless video streaming,” in *International Packet Video Workshop*, vol. 2002, 2002.

[19] Q. Li and M. Van der Schaar, “Providing adaptive QoS to layered video over wireless local area networks through real-time retry limit adaptation,” *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 278–290, 2004.

[20] I.-H. Hou and P. R. Kumar, “Packets with Deadlines: A Framework for Real-Time Wireless Networks,” *Synthesis Lectures on Communication Networks*, vol. 6, no. 1, pp. 1–116, 2013.

[21] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[22] A. Devlic, W. John, and P. Sköldström, “A use-case based analysis of network management functions in the ONF SDN model,” in *2012 European Workshop on Software Defined Networking*, pp. 85–90, IEEE, 2012.

[23] M. Jarschel, T. Zinner, T. Hoßfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for SDN," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 210–217, 2014.

[24] M. Yang, Y. Li, D. Jin, L. Zeng, X. Wu, and A. V. Vasilakos, "Software-defined and virtualized future mobile and wireless networks: A survey," *Mobile Networks and Applications*, vol. 20, no. 1, pp. 4–18, 2015.

[25] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 623–654, 2015.

[26] W. Hu, H. Yousefi'zadeh, and X. Li, "Load adaptive MAC: A hybrid MAC protocol for MIMO SDR MANETs," *IEEE Transactions on Wireless Communications*, vol. 10, no. 11, pp. 3924–3933, 2011.

[27] H.-S. W. So, J. Walrand, and J. Mo, "McMAC: A parallel rendezvous multi-channel MAC protocol," in *2007 IEEE Wireless Communications and Networking Conference*, pp. 334–339, IEEE, 2007.

[28] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson, "PW-MAC: An energy-efficient predictive-wakeup MAC protocol for wireless sensor networks," in *INFOCOM, 2011 Proceedings IEEE*, pp. 1305–1313, IEEE, 2011.

[29] T. Van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pp. 171–180, ACM, 2003.

[30] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *INFOCOM 2002. Twenty-First Annual Joint Con-*

ference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 3, pp. 1567–1576, IEEE, 2002.

[31] M. Neufeld, J. Fifield, C. Doerr, A. Sheth, and D. Grunwald, "SoftMAC - Flexible Wireless Research Platform," *Hot Topics in Networking (HotNets-IV)*, 2005.

[32] A. Sharma, M. Tiwari, and H. Zheng, "MadMAC: Building a reconfiguration radio testbed using commodity 802.11 hardware," in *Networking Technologies for Software Defined Radio Networks, 2006. SDR'06. 1st IEEE Workshop on*, pp. 78–83, IEEE, 2006.

[33] M.-H. Lu, P. Steenkiste, and T. Chen, "FlexMAC: A wireless protocol development and evaluation platform based on commodity hardware," in *Proceedings of the Third ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, pp. 105–106, ACM, 2008.

[34] C. Doerr, M. Neufeld, J. Fifield, T. Weingart, D. C. Sicker, and D. Grunwald, "MultiMAC - An adaptive MAC framework for dynamic radio networking," in *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005.*, pp. 548–555, IEEE, 2005.

[35] E. Blossom, "GNU Radio: Tools for exploring the radio frequency spectrum," *Linux Journal*, vol. 2004, no. 122, p. 4, 2004.

[36] T. Schmid, O. Sekkat, and M. B. Srivastava, "An experimental study of network performance impact of increased latency in software defined radios," in *Proceedings of the Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, pp. 59–66, ACM, 2007.

[37] A. Khattab, J. Camp, C. Hunter, P. Murphy, A. Sabharwal, and E. W. Knightly, "WARP: A flexible platform for clean-slate wireless medium access protocol design," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 12, no. 1, pp. 56–58, 2008.

[38] J. J. Jaramillo and R. Srikant, "Optimal Scheduling for Fair Resource Allocation in Ad Hoc Networks With Elastic and Inelastic Traffic," *IEEE/ACM Transactions on Networking*, vol. 19, no. 4, pp. 1125–1136, 2011.