ENHANCING COEXISTENCE, QUALITY OF SERVICE, AND ENERGY PERFORMANCE IN DYNAMIC SPECTRUM ACCESS NETWORKS

by

Ashwini Kumar

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2011

Doctoral Committee:

Professor Kang G. Shin, Chair
Professsor Edward T. Zellers
Associate Professor Zhuoqing Mao
Assistant Professor Prabal Dutta

To my parents

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my advisor, Professor Kang G. Shin, for his constant encouragement and support during the course of my doctoral study. His guidance, insights, and strong pursuit of excellence have helped my growth tremendously, both professionally and as a person. I also thank my dissertation committee members—Professors Morley Mao, Prabal Dutta, and Ted Zellers for their valuable feedback and suggestions that resulted in a greatly improved PhD work.

I wish to acknowledge my parents who have constantly given me endless love and encouragement at every stage of my life. This dissertation is dedicated to them. I also thank my dear sister and brother-in-law for their support through the highs and lows of this phase. I want to specially mention my nephew little Arnav whose playful frolics always cheered me up during every visit.

I am also grateful to my talented colleagues, whose invaluable comments and critical input improved the quality of my work. I thank the present and former RTCL members, especially Hyoil Kim, Alex Min, Kyu-Han Kim, Xin Hu, Xinyu Zhang, Eugene Chai, and others. I am also thankful to my collaborators and former mentors, Dr. Jianfeng Wang at Philips Research, and, Dr. Dragos Niculescu and Dr. Young-June Choi at NEC Laboratories. I wish them the very best in their careers and life.

Finally, this adventure would not have been possible without the support of my wonderful friends Naveen, Anurag, Trushal, Jayesh, Utsav, and Shantanu. Thanks for sharing this journey with me and making it fun.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

LIST OF ABBREVIATIONS

ApEn	Approximate Entropy
AP	Access Point
BC	Backup Channel
BER	Bit Error Rate
BP	Base Power
BS	Base Station
CASA	Context-Aware Spectrum Agility
CCTT	Channel Closing Transmission Time
CDT	Channel Detection Time
CH	Correspondent Host
CMT	Channel Move Time
CP	Coexistence Period
CR	Cognitive Radio
CRN	Cognitive Radio Network
DEMD	Dynamic Energy Management for DSA
DSA	Dynamic Spectrum Access
DSAN	Dynamic Spectrum Access Network
DSASync	DSA Synchronization
FCC	Federal Communications Commission
GPP	General Purpose Processor

IDT	Incumbent Detection Threshold
ISM	Industrial, Scientific and Medical
LAN	Local Area Network
MAC	Medium Access Control
PCO	Power Consumption Overhead
PD	Probability of Detection
PER	Packet Error Rate
PFA	Probability of False Alarm
PHY	Physical Layer
PU	Primary User
PUG	Primary User Group
QoS	Quality of Service
QPDP	QoS-Provisioned DSA Protocol
RTCP	RTP Control Protocol
RTP	Real-Time Transport Protocol
SA	Spectrum Agility
SDR	Software Defined Radio
SH	Spectrum-agile Host
SNR	Signal to Noise Ratio
SpeCWiFi	Spectrum-Conscious WiFi
SU	Secondary User
SUG	Secondary User Group
SUT	System Under Test
TCP	Transmission Control Protocol
TFP	Transmission Freeze Period
TFR	Traffic Fulfillment Ratio
UDP	User Datagram Protocol

UHF	Ultra High Frequency
UNII	Unlicensed National Information Infrastructure
VHF	Very High Frequency
WLAN	Wireless Local Area Network
WN	Wired Network
WPC	Wireless PHY Chain

ABSTRACT

Dynamic Spectrum Access (DSA) is an upcoming wireless technology which aims to alleviate spectrum-usage inefficiency from the current static spectrum allocation model of wireless communication. However, DSA technology is still in its infancy. Despite its enormous potential for improving wireless networking performance and quality of service (QoS), contemporary DSA is highly ineffective in realizing such advantages to consumer-oriented wireless systems and networks. Therefore, existing DSA technology is unsuitable for actual deployment in its current form. The thesis explores this hypothesis in depth and identifies its important causes, which span three key dimensions—coexistence, QoS, and energy. The thesis proposes novel and practical system-oriented solutions in order to address the issues identified with traditional DSA. "Awareness-cum-adaption" is the central theme across the proposed methods.

First, the thesis focuses on the generic DSA coexistence problem. It presents a dual-mode DSA operation scheme featuring joint sensing and transmission scheduling for safe and efficient time-domain incumbent-unlicensed coexistence. This solution expands the application of DSA to most of the licensed spectrum, which greatly improves the utility and effectiveness of DSA technology. A prototype, called *Spectrum-Conscious WiFi* (SpeCWiFi), is also developed. Second, it provides important insights into the QoS impact of DSA through the case study of a consumer DSA-based wireless network. Third, it presents *Context-Aware Spectrum Agility* (CASA) to address the DSA's QoS issues from a device-centric perspective. Fourth, to tackle the QoS problems at the network level, it proposes a network service framework

called *DSA Synchronization* (DSASync). DSASync consists of algorithms based on buffering and traffic-shaping to effectively manage end-to-end connections in DSA networks. CASA and DSASync together provide a complete solution to application QoS issues associated with existing DSA, and hence, make DSA a highly effective and performance-enhancing technology for consumer wireless networks. Finally, this thesis explores the energy cost of DSA through an empirical analysis, and proposes the *Dynamic Energy Management for DSA* (DEMD) scheme to reduce its energy footprint.

CHAPTER I

Introduction

Rapid proliferation of consumer wireless networks (e.g., Wifi hotspots, cellular and WiMAX networks), together with the increasing popularity of mobile/wireless devices (e.g., laptops, smartphones), is straining the wireless networking performance like never before. Interference due to near-saturation level usage of the wireless medium, especially in the unlicensed Industrial, Scientific, and Medical (ISM) bands, is resulting in wireless network capacity shortfalls and poor end-user quality of service (QoS). The performance problem is further compounded because of a significant increase in bandwidth-intensive and QoS-sensitive network applications or services. Thus, there is a widening demand-supply mismatch in wireless networking performance that can potentially hurt the growth of wireless industry.

The wireless networking research community has responded to this challenge along several dimensions. These include, for example, the approaches for reducing wireless transmission footprint through power control and beamforming; transmission multiplexing using Multiple Input Multiple Output (MIMO) techniques, or development of sophisticated physical layer (PHY) encoding/decoding and error-correction schemes. Unlicensed wireless operation, which involves opportunistic access of the licensed spectrum by unlicensed devices, attacks the performance and QoS performance problem along the spectrum-usage inefficiency dimension.

1.1 Dynamic Spectrum Access: Potential and Background

Recent spectrum surveys [1–3] have shown that certain narrow spectrum regions are now almost fully utilized (e.g, 2.4GHz ISM bands) and are highly congested, while a majority of the licensed spectrum is significantly underutilized—thus, producing the spectrum-usage inefficiency and spectrum scarcity problems. *Dynamic Spectrum Access* (DSA) [4], also called *Spectrum Agility* (SA), is a new wireless networking paradigm that aims to address these problems through unlicensed wireless operation. DSA relies on opportunistic exploitation of licensed channels by unlicensed devices, called *Secondary Users* (SUs). Such unlicensed accesses must occur when authorized licensee devices, called *Primary Users* (PUs), are not concurrently accessing the channel, i.e., during *spectrum white spaces*. DSA paradigm is motivated from the aforementioned spectrum surveys, that show existence of abundant spectrum white spaces along both frequency and time dimensions. Many licensed channels are seen to be severely underutilized (average <50% across 20MHz-3GHz spectrum), even in urban areas.

The potential benefits of spectrum-agile operations has led regulatory bodies, like FCC in the USA, to move towards opening TV channels for DSA [5, 6]. A growing interest is being witnessed in the development of DSA-based wireless products, especially for TV bands [7–10].

DSA has much broader scope and promise than that in the TV bands. Fundamentally, DSA is not limited to a particular spectrum region, but would also involve opportunistic switching of channels between different spectrum regions (e.g., WiMAX bands to TV bands). We take this more general view of DSA in which, apart from switching channels, transmission/reception mechanisms (or the MAC-PHY protocols) may also need to be dynamically updated, e.g., in order to accommodate different wireless characteristics of various spectrum bands. Realizing this potential, efforts have begun to advance DSA to other licensed bands, in addition to TV spec-

trum [11, 12].

Despite some recent progress, DSA technology is still in an early stage of development. Several challenges (both technical and economic) remain before DSA becomes mature and viable enough to be rolled out in consumer wireless systems and networks. This thesis looks into several of such technical issues that are roadblocks to the evolution of DSA and proposes their solutions. To better understand such issues, we next provide a design overview of DSA.

DSA, as a networking module, spans physical and link layers, as shown in Fig. 1.1. The physical layer aspect is captured by *Software Defined Radio* (SDR) or *Cognitive Radio* (CR) [13–16], which provides the necessary radio capability for DSA. More relevant to this thesis are the higher-layer MAC aspects that manage DSA operation over SDRs/CRs. There have been numerous DSA protocol proposals in literature [17–20]. The process for DSA standardization has also begun [21, 22]. However, at the time of writing this thesis, there is no consensus in the research community on a standard and well-accepted DSA protocol. Thus, instead of selecting one protocol proposal and disregarding others, we consider DSA from a *functional abstraction* viewpoint in this thesis.

Our study of several DSA protocols reveals that currently proposed DSA protocols share certain key functions that must be performed to achieve DSA. This observation enabled us to create an abstract function model of DSA which forms the basis of solutions developed in this thesis. Our approach ensures the generality of our proposed methods and analysis, despite the evolving nature of DSA research. It permits as to study and understand the behavior of fundamental DSA components, without incurring incompatibility problems or selectivity bias from choosing one (or a few) specific DSA protocols.

From a functional perspective, DSA consists of three components: *spectrum sens-*

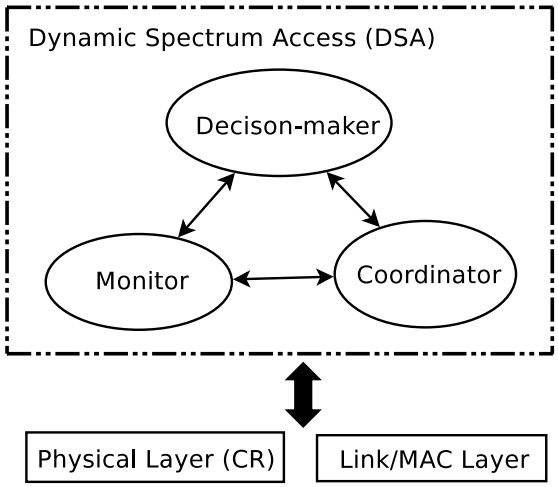


Figure 1.1: Functional model of a typical DSA protocol.

ing, *spectrum-use decision making*, and *coordination* (see Fig. 1.1).[1]

The *spectrum sensing* component scans channels in the spectrum, and acquires relevant time-variant characteristics for each channel. For instance, a DSA protocol typically maintains a list of channels together with their average spectrum white-spaces. This list is referred to as the *Spectrum Opportunity Map* (SOM). Spectrum sensing involves scheduling of *quiet periods* (QPs) during which data packets cannot be transmitted/received, in order to observe a channel.

The *spectrum-use decision making* component determines the channel for secondary devices to use, and invokes the channel-switching and coordination procedure, if needed. This component analyzes the information (i.e., SOM) gathered by the sensing component. For instance, it is invoked when an incumbent signal is detected on the current channel.

The *coordination* component orchestrates DSA decisions in a multi-node DSA network. For instance, in ad-hoc DSA networks, the coordination component ensures that the SUs are on the same channel, thus maintaining their inter-communication. Many of the proposed DSA protocols use *control channels* in order to accomplish this

[1]These functions are realized through the underlying MAC (e.g., for coordination) and PHY schemes (e.g., for sensing).

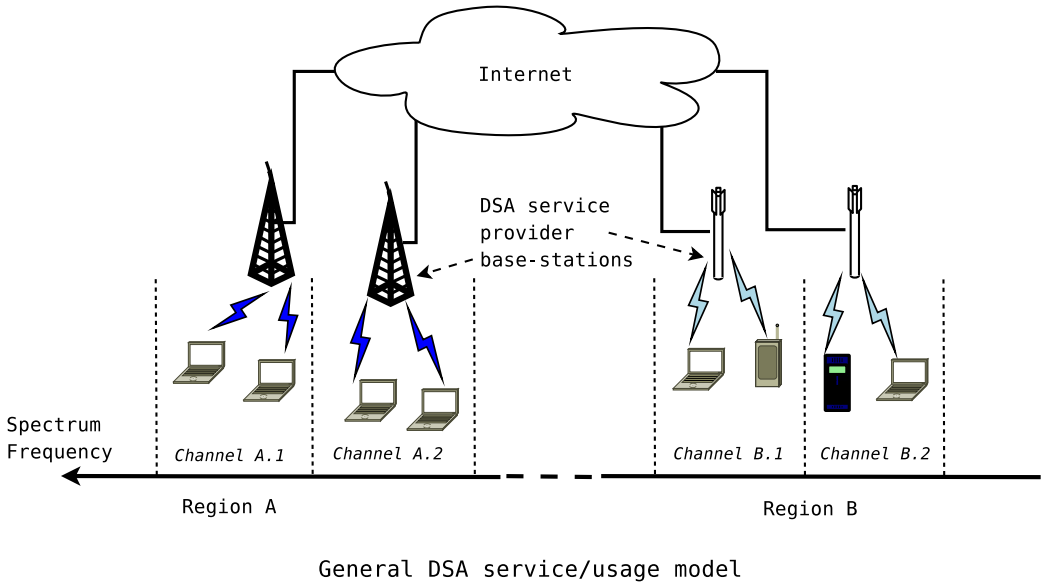


Figure 1.2: Client wireless devices in first-/last-mile wireless access networks constitute the system model.

coordination.

As seen from its abstract function model, DSA's main design objective is to gain as much spectral resources as possible. For more details on the design and techniques involved in the aforementioned DSA components, interested readers are pointed to the references in [4, 17–19, 23].

1.2 System Model

We consider unlicensed wireless operation in the secondary wireless service market model, as described below. The service structure for the secondary market is still evolving, but is expected to closely resemble the currently existing consumer wireless service model (e.g., cellular service), as their service infrastructure already exists with proven effectiveness and success.

Device: The system model consists of general-purpose computing devices, each equipped with one DSA-capable wireless interface. In general, the wireless interface would be composed of a highly-reconfigurable SDR/CR [24,25] together with a highly-

tunable antenna [15, 26]. A basic prototype of a CR, albeit with limited capabilities, is the SDR platform USRP2 [27].

We consider a single such CR-based data interface scenario because of its cost-size-design advantage and simplicity of analysis. However, the issues identified in this dissertation (and their proposed solutions) also apply to multi-interface (or multi-CR) devices. The SU devices may utilize the same interface for spectrum sensing or have separate specialized *spectrum-sensor* hardware to minimize interruptions to data transfers during sensing. External sensing infrastructure (e.g., a separate sensor network or a sensing server) can also be used for collecting information on spectrum conditions.

Network: A SU device, by making use of its DSA capability, can migrate to a licensed channel where it can connect to the secondary gateway for available network services. Thus, the SU is a client in a first/last-mile wireless access network (or an edge access network), as shown in Fig. 1.2. Such a network is also referred to as a *Dynamic Spectrum Access Network* (DSAN), *Cognitive Radio Network* (CRN), or a *Secondary User Group* (SUG). Like an individual SU client device, it is also possible that a SUG can dynamically tune to another licensed channel, if allowed by its DSA protocol. Note that there may be a co-located *Primary User Group* (PUG) on the same channel as that of the SUG. During channel-access, a SU device will be part of a one-hop WLAN (like a Wifi hotspot) and must share the opportunities with other SUs, as shown in the expanded view in Fig. 1.3.

Example: A typical scenario would consist of a PDA-like wireless device, which is in the range of multiple edge access networks on different channels, possibly of different types (e.g., 802.11 WLANs, WiMAX, or cellular). One or more of the networks can be *primary* to the device (i.e., authorized to use it at any time), while other networks are available on a secondary basis and can be accessed opportunistically (if unused by their own licensed devices) by exploiting the device's DSA capability.

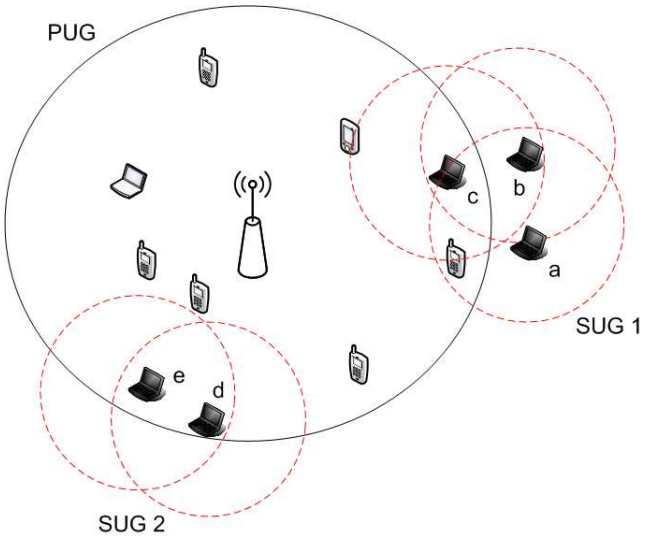


Figure 1.3: The PU network typically consists of multiple clients associated with a base-station. The solid circle denotes the transmission range of the PU base station while dashed circles show the range of the SUs.

Remarks: With the aforementioned model, this thesis addresses issues from both device- and network-centric perspectives, thus yielding more balanced and effective solutions. Although our research focus is on generic one-hop point-to-multipoint DSA networks, fully ad-hoc/distributed DSANs can also benefit from the solutions proposed in this thesis.

1.3 Challenges and Limitations in Contemporary DSA

Following is the main observation about existing DSA technology that forms the hypothesis of this thesis.

Despite an enormous potential for improving wireless networking performance and quality-of-service (QoS), contemporary DSA is highly ineffective in providing such advantages to consumer-oriented wireless systems and networks. Narrow scope of application, together with associated overheads and negative side-effects, outweigh DSA's benefits, and can even degrade normal wireless communication performance. Therefore, existing DSA technology is unsuitable for actual deployment in its current

form.

The thesis explores and validates this hypothesis in detail. Further, it shows that this problem with contemporary DSA is mainly a combined outcome of technical shortcomings in three key areas of wireless communication—coexistence, QoS, and energy.

1.3.1 Poor Time-Domain Coexistence

Time-domain coexistence of Primary Users (PUs) with Secondary Users (SUs), referred to as *DSA coexistence*, is a necessary function for DSA. "DSA coexistence" must be provided when the licensed channel has already been selected by the SUs and should then be utilized. DSA coexistence must balance two conflicting objectives—(a) guarantee safe operations for the PUs, (b) ensure efficient utilization of spectrum white spaces by the SUs. In a broader context, effective DSA coexistence represents an important step towards universal DSA and non-intrusive coexistence of heterogeneous wireless technologies.

Current DSA research lacks adequate coverage of DSA coexistence, as its focus has exclusively been on TV bands (or UHF/VHF spectrum) on account of FCC's establishment of rules [5,6] that open up certain TV channels for DSA. For TV bands, DSA coexistence simply constitutes halting unlicensed transmissions within a specific time-frame when a PU activity is detected—IEEE 802.22 standard draft [21] ceases SU transmission within 2 seconds, as required by regulatory guidelines. In this thesis, we show that this form of DSA coexistence is ineffective in most of the licensed channels because of the underlying diversity in incumbent channel-access characteristics. A significant fraction of the licensed spectrum is characterized by short-term ON/OFF incumbent behavior, unlike TV channels [3]. Thus, existing reactive approach to DSA coexistence precludes the applicability of DSA for a majority of the licensed spectrum.

It is important to differentiate between DSA coexistence (or PU–SU coexistence), from self-coexistence (or SU–SU coexistence). Self-coexistence is simpler to provide and existing DSA standard proposals, like 802.22 [21], outline several mechanisms for this. Self-coexistence in DSA has also been studied by Sparta [28] and Flex [29], among other works.

Related Work: At a high level, our motivation for this problem arises from the performance benefits associated with coexistence of heterogeneous wireless technologies. Various protocol-specific coexistence methods have been suggested in literature. For example, WiMAX and Wifi coexistence in [30], and, coexistence of ZigBee, Wifi and Bluetooth wireless systems in [31, 32], among others. Further, coexistence has also been considered in the context of 802.11 amendments (802.11g/n) [33, 34] for backward compatibility reasons. The obvious disadvantage with such approaches is that they are specific to certain wireless standards and spectrum, resulting in narrow scope and unscalability of the solution.

The key constraint of coexistence based on unlicensed operation is not considered in the aforementioned approaches. We argue that coexistence through DSA provides a more general approach to heterogeneous coexistence leading to a wider scope of application and usefulness. DSA coexistence can be used with a wide range of protocols, and across the wireless spectrum.

Our DSA coexistence problem formalization (detailed in Section 2.3) is related to [35, 36] that identifies 802.11 as a coexistence system in the time domain. But again, they did not focus on DSA coexistence. While the formalized DSA coexistence problem is found to be related to the well-known *exploration vs. exploitation* class of problems, it entails unique constraints of DSA, like mandatory time-bound safety for PUs. In fact, the problem is found to be closer to restless multi-armed bandit version, and is shown to be PSPACE-hard in general [37]. Thus, optimal solutions can be found only for very limited situations with several highly restrictive assumptions [38].

Thus, approximate solutions have to be employed in practice. Along this direction, our proposed solution features joint transmission and sensing scheduling based on PU channel-usage pattern estimation.

Our proposed solution for the DSA coexistence problem involves estimation of the incumbent's channel-usage pattern (details in Section 2.4). Some existing work in DSA research have used this approach, e.g., [3, 39], although towards different objectives. Our estimation scheme is based on Approximate Entropy (ApEn) [40], which differs from existing techniques in being generic. For example, it makes no restrictive assumptions like i.i.d. ON and OFF durations for PUs as in [39].

Further, a significant benefit of our ApEn-based estimation approach is its amenability to online deployment due to its very low overhead. Hence, it can provide a high degree of adaptivity in real time. While [3] proposes a 2-D frequent pattern mining algorithm to predict channel availability, it cannot be used online as it entails significant computational overhead and requires a long duration of training data (at least 2 hours). The computational overhead incurred also precludes other pattern matching techniques like autocorrelation and FFT from being used in our proposed solution.

Note that ApEn was introduced as a measure of system complexity in [40]. An experimental study of the usefulness of ApEn metric in the context of DSA was also presented in [41]. However, its focus was on improving the reliability of spectrum sensing rather than DSA coexistence.

A recent study [42] has shown some possibility of achieving full duplex communication on a single wireless channel, i.e., transmit and receive packets at the same time. This would resolve the basic tradeoff inherent in DSA, i.e., "exploration vs. exploitation" of licensed channels. Thus, the DSA coexistence problem (and the solution) would be simplified. However, [42] imposes restrictions on transmit power, bandwidth, and size, thus introducing several limitations to its practical feasibility. For example, the proposed approach can cancel only up to 80dB (typically 60dB) and

will not work with transmit power more than 20dBm. DSA regulations from FCC [6] require PU detection very close to noise floor (-116dBm), which would be impossible to support with this method.

1.3.2 Lack of Adequate QoS Support and Management

Despite its potential advantages, DSA is a fundamentally disruptive technology because of the unwarranted side-effects of its functional components. Key functions involved in DSA, like spectrum sensing or channel switches, can cause delays and disruptions to the application, thereby introducing QoS degradation and reduced bandwidth to application traffic. Minimizing the interference to incumbents is the key requirement in opportunistic usage of the licensed spectrum. Thus, any PU activity adds to the interruptions suffered by applications running on SUs. Further, DSA may also result in link capacity fluctuations, for example, due to a reduction in frequency-width or less-efficient MAC-PHY schemes on a new channel. Session handovers, terminations and re-establishments may occur, exacerbating application QoS degradation. Thus, QoS-provisioning schemes are a must in DSA to sustain and enhance the end-user experience.

Current DSA research incorporates channel-awareness, but does not consider application QoS support as an important objective. This thesis provides both device-centric and network-centric mechanisms for QoS support and management in DSA.

Related Work: Though almost all of the prior work in DSA incorporates channel-awareness, most of it does not consider DSA's impact on application-layer QoS requirements. Certain recent efforts attempt to account for QoS and context-awareness in DSA [11, 43, 44] using either Reinforcement Learning or Game Theory. However, the QoS considered in these proposals is not the high-level application QoS metrics, but link-level metrics like SNR and BER [11] or number of successful link-layer transmissions [44]. Hence, their QoS context awareness is very narrow. Applica-

tion QoS demands (like goodput, delay, and jitter) have higher-level semantics and their dependency on link-level parameters is not considered in such prior work, which we rigorously define through our proposed semantic matching approach (details in Section 4.5).

Apart from narrow QoS context, the adaptive responses in [11, 43–45] are also limited and just include channel selection and spectrum sharing schemes—typically for every packet transmission which incurs very high overhead. No adaptation of DSA's fundamental parameters (like sensing duration) is considered. Finally, the implementation and practical issues are neither analyzed nor evaluated. On the other hand, our proposed solution (details in Section 4.3) has low operational overhead while providing significant QoS benefits to consumer applications in realistic settings.

Our approach is inspired by the benefits shown for adaptation based on application behavior in several other system optimization strategies, including power management of wireless interfaces [46] and wireless network selection [47]. In principle, our solution adopts a similar approach, providing a cross-layer framework in order to provide application QoS hints to the underlying DSA protocol which can adapt intelligently to accommodate application QoS requirements.

1.3.3 High Energy Cost

As noted earlier, DSA technology requires a highly configurable radio with widely tunable antenna. Hence, conventional hardware radios (analog or ASIC based) are not suitable for this purpose. SDRs/CRs [16, 24, 48, 49] have been identified as the enabling radio platform for DSA.

Apart from other benefits, a key advantage of SDRs is software-based signal processing on the *General Purpose Processor* (GPP), which ensures dynamic reconfigurability of the radio. This feature is especially suited for DSA. However, signal processing on GPP is a computationally intensive process which can incur unaccept-

ably high energy overhead. Our empirical analysis of DSA over SDRs (details in Chapter VI) confirms the existence of this problem. Energy consumption overhead associated with DSA (on SDR platform) is found to be very high and unsustainable with contemporary DSA.

Related Work: Though several DSA protocols have been proposed in literature, none of them consider energy overhead of the underlying SDR platform as a key design criteria. The performance problem of SDR has witnessed some attention recently [50, 51]. But the energy aspect has not been studied. On the other hand, there has been some work in the computer architecture community towards designing energy-efficient processors for signal processing [52, 53]. However, they do not constitute true SDRs because of their focus on specialized processors (like DSPs) targeted for wireless protocol execution, rather than on off-the-shelf GPPs (as in [50] or [51]).

Our proposed solution to lower the energy overhead with DSA is based on adaptive wireless PHY chain management (details in Section 6.6). Adaptive rate modulation is part of a number of wireless protocols, e.g., 802.11 [33]. Also, adaptive channel-width has recently been proposed to improve communication performance [23, 54]. However, the objective of these approaches is performance, rather than energy. With the advent of SDRs, fully adaptive wireless PHY chain is now feasible, which forms the basis of our solution.

1.4 Research Goals: QoS-aware DSA with Broader Scope and Lower Energy Footprint

Motivated by the issues identified in contemporary DSA, as discussed in Section 1.3, this dissertation has the following research objectives.

- **Enabling generic DSA coexistence:** For wide applicability of DSA across diverse licensed channels, time domain "licensed-with-unlicensed" coexistence

is necessary. The goal is to provide practical and efficient awareness of incumbent channel-access characteristics and consequent scheduling of unlicensed transmissions.

- **Characterizing QoS over DSA:** DSA is a new technology for sharing the wireless spectrum, and its application QoS impact has not been studied sufficiently. This thesis aims to analyze the application QoS behavior in consumer-oriented DSA-based wireless systems. More specifically, we focus on a home wireless network (which features DSA) as our study platform. This study also intends to provide a basic QoS-provisioning guideline featuring important QoS-improving techniques that are particularly effective in the context of DSA.

- **Optimizing DSA based on application needs:** User-perceived QoS must be enhanced when using DSA. To achieve this goal, the underlying DSA protocol must be sufficiently intelligent to match dynamic application QoS needs. This thesis targets development of a practical and low-overhead framework for application QoS support and management, together with the adaptation algorithms.

- **Improving end-to-end connection performance:** With most wireless networks serving as edge access networks for services on the Internet, the end-to-end networking performance is very important. Device-centric QoS support is not sufficient here. To ensure improved performance for connection streams in DSA networks, the goal is to provide a simple and non-intrusive network management solution that is also compatible with the existing networking protocols.

- **Analysis and optimization of energy usage:** Energy overhead of DSA is a major roadblock in deployment of DSA technology. This thesis aims at

quantifying the energy consumption behavior of DSA, and developing energy-usage optimization approach for the same.

1.5 Research Contributions

In accordance with the aforementioned goals, we design, implement, and evaluate approaches that improve DSA along coexistence, QoS, and energy fronts. The central theme across our proposed solutions is to incorporate the principle of *awareness-based adaptation*. Practicality, together with low operational and deployment overheads, are the key advantages of the methods presented in this thesis. Their feasibility is demonstrated through software prototype development and testing using off-the-shelf wireless networking products.

1.5.1 Enhancing DSA Coexistence

This thesis advances the state-of-the-art DSA coexistence by exploiting white spaces in time-domain in a comprehensive manner for unlicensed access across the licensed spectrum. The approach involves awareness and estimation of incumbent channel-access patterns using Approximate Entropy (ApEn) [40]. Unlicensed devices schedule their own transmissions based on the estimated incumbent pattern, resulting in efficient spectrum white-space usage by the SUs while maintaining a high degree of safety for PU transmissions. An implementation of the proposed DSA coexistence protocol, called *Spectrum-Conscious Wifi* (SpeCWiFi), is also developed and evaluated.

The proposed DSA coexistence mechanism and deployment framework addresses three key DSA coexistence challenges: (a) low-overhead awareness and estimation of incumbent characteristics, (b) safe and efficient adaptation through secondary transmission scheduling, and, (c) distributed medium access coordination among SUs.

1.5.2 Analysis of DSA's QoS Impact

This thesis characterizes the application QoS issues with DSA through a case study of HDTV-streaming over a home DSAN. The study platform features stringent QoS requirements that will need to be fulfilled by DSANs. Our analysis reveals the adverse impact on application QoS due to unwanted side-effects resulting from several fundamental DSA operations. Further, we also propose a basic design guideline for DSA protocols (at MAC level) to minimize their direct QoS impact. A QoS-aware DSA MAC protocol, called *QoS-Provisioned DSA Protocol* (QPDP), is also developed for illustration and evaluation. This case study provides a valuable insight into impact of DSA on application QoS, which has been lacking so far in the realm of DSA research.

1.5.3 Enabling Application-Awareness in DSA

Since application performance dictates end-user experience, this thesis develops a novel and practical framework for application-awareness in DSA in order to improve application QoS. The proposed framework, called *Context-Aware Spectrum Agility* (CASA), consists of low-overhead aggregation of fundamental QoS requirements (e.g., bandwidth needed) that are directly affected by DSA's negative side-effects. The concept of *semantic context alignment* is developed to match the higher-layer QoS abstractions to low-level DSA parameters. In the process, several semantic dependency equations are developed. Finally, a rewards-based adaptation algorithm, called the *CASA Algorithm* is proposed to make DSA application-aware and dynamically adaptive to QoS demands.

CASA framework provides two major QoS-enhancing benefits for DSA: (a) device-level masking of DSA side-effects, and, (b) dynamic adaptation to variations in spectrum conditions relative to application requirements.

1.5.4 End-to-end Connection Management in DSANs

Application-awareness (through CASA) makes DSA more intelligent and proactive in providing good application QoS at the end device. However, this solution does not address adverse DSA impacts on end-to-end connections. Hence, this thesis proposes network-level management of DSA. We design efficient algorithms (based on buffering and traffic-shaping), together with an easily-deployed architecture in edge DSANs for end-to-end connection management. We call the proposed framework as *DSA Synchronization* (DSASync). DSASync targets the transport layer as transport protocols form the first end-to-end layer in the protocol stack, with most impact on connection quality. DSASync comprises techniques for managing both TCP and UDP connection streams. To ensure compatibility and semantic consistency, DSASync exploits the built-in flow and congestion control mechanisms in TCP. Since UDP is connectionless, DSASync attempts to leverage the information carried in higher layer packet headers.

DSASync features three key benefits for end-to-end connections in edge DSANs: (a) network-level masking of DSA side-effects, (b) adaptive transport layer traffic management, and, (c) compatibility with the existing networks by maintaining end-to-end semantic consistency of TCP and UDP.

1.5.5 Managing Energy Cost

This thesis provides an empirical study of the energy overhead incurred when using SDRs, and develops an energy profile that quantifies the relative impact of different factors involved in energy consumption. Since SDR is the key enabling platform for DSA, this analysis also provides important insights into how to reduce the energy consumption involved in realizing DSA. Based on the developed energy profile, we develop an energy optimization scheme called *Dynamic Energy Management in DSA* (DEMD) in order to provide energy-efficient DSA operation. The key compo-

nent of DEMD is the *DEMD Algorithm* that introduces adaptive wireless PHY chain management over the SDR to lower energy usage, whenever feasible.

The empirical analysis and proposed DEMD solution features two important contributions: (a) quantify the relative energy impact of different MAC-PHY components in DSA design, (b) provide an effective mechanism to reduce energy overhead of DSA (and SDR-based wireless communication in general), while still maintaining necessary performance.

1.6 Organization of the Dissertation

Rest of the thesis is organized as follows. Chapter II presents our approach for efficient and safe DSA coexistence, together with a description and evaluation of the prototype SpeCWiFi. Chapter III provides details on our case study on application QoS in a consumer DSAN. CASA, the framework for application-aware operations for DSA, is described in Chapter IV. DSASync, the framework for end-to-end connection stream management in DSANs, is discussed in Chapter V. DEMD, the approach for adaptive energy management of DSA over SDRs is described in Chapter VI. The thesis concludes with Chapter VII.

CHAPTER II

Safe and Efficient Time-Domain Coexistence of Unlicensed and Licensed Spectrum Users

2.1 Introduction

As discussed in Section 1.3, we postulate that the advantages and scope of DSA can be significantly enhanced by effective harnessing of spectrum white spaces in time-domain. We view a time-domain approach as complementary to frequency-domain methods in providing a complete unlicensed access solution.

DSA coexistence problem: This chapter explores the problem of *DSA co-existence*, defined as *time-domain unlicensed coexistence* of PUs with SUs. Sharing of spectrum white spaces in frequency-domain is assumed to be handled through other higher-level DSA functions, such as a spectrum-management component [17,55]. Thus, by this definition, "DSA coexistence" comes into play only when the channel[1] has already been selected by the SUs and should then be utilized, i.e., after the frequency-domain aspect of unlicensed access has been established.

Inadequacy of current DSA coexistence: Existing DSA research features very basic type of DSA coexistence scheme, as it has been applied only to TV bands.

[1]We use the word "channel" to denote any portion of a licensed spectrum region. It may consist of one or more physical channels.

Currently, DSA coexistence implies ceasing unlicensed transmissions within a specified time-limit when a PU activity is detected (2 seconds in IEEE 802.22 standard draft [21]). This strategy is suitable only for licensed channels characterized by *slow-varying* incumbent behavior in time-domain. For example, TV transmissions are either ON or OFF for long durations (on the order of hours) at a time. Thus, when a TV signal is detected, it is expected to remain on the channel for a long time, and vice versa. Hence, the best coexistence strategy is to find a new vacant channel, trivially achieving good coexistence between PUs and SUs.

This form of DSA coexistence, however, is ineffective in most of licensed spectrum because of the underlying diversity in incumbent channel-access characteristics. For instance, SUs should be able to coexist effectively alongside PUs that produce channel-access patterns varying on a small time-scale. For example, the WiMAX-based IEEE 802.16h draft [56] explicitly proposes spectrum white spaces of $\sim$10ms duration, depending on traffic load. Such small but frequent time-domain white spaces are also seen in other spectrum bands like satellite and cellular service channels [3]. In channels characterized by *fast-varying* incumbent behavior, PUs access channels very frequently, but each access lasts for a short time, resulting in very small ON/OFF durations. Even though each PU–SU transmission overlap is controlled to be within the specified regulatory time-limit, the accumulated interference to PUs over a long term can be significant, and hence, unacceptable to licensees.

Summary of proposed approach: This work advances the conventional DSA coexistence[2] by exploring the feasibility and addressing the challenge of exploiting white spaces in time-domain in a comprehensive manner, thus enabling unlicensed access across diverse spectrum bands. We first formalize the DSA coexistence problem which must take into account the two key requirements:

1. *PU-safety:* PUs must be protected from the interference generated by SUs;

[2]Throughout this chapter, by "DSA coexistence" or "coexistence," we mean time-domain PU–SU coexistence, unless mentioned otherwise.

2. *SU-efficiency:* SUs should be able to maximize the utilization of available white spaces.

PU-safety is an important issue as licensees are extremely concerned about interference from SUs, and hence reluctant to permit unlicensed users in their channels. Regulatory guidelines from the US FCC have mandated incumbent protection in the TV channels [5, 6]. PU-safety and SU-efficiency inherently conflict with each other, because increase in channel utilization by SUs leads to an increase in interference to PUs. This tradeoff is more pronounced in fast-varying channels. To precisely quantify this tradeoff, we define a new metric, called the *Coexistence Goodness Factor* (CGF). CGF is a comprehensive metric to evaluate the effectiveness of DSA coexistence protocols in any licensed spectrum region.

In order to quantify the run-time performance of DSA coexistence, we define a CGF-based multi-objective function. While optimization of the proposed objective function is simple to do in theory, it is difficult to do in a real deployment. For practical optimization of the objective function, we propose an intelligent dual-mode DSA coexistence protocol, with *Safe Mode* (SM) and *Aggressive Mode* (AM). The proposed scheme features joint transmission-cum-sensing scheduling based on *Approximate Entropy* (ApEn) [40] estimate of PUs' channel-access pattern. We also develop a proof-of-concept prototype on 802.11 MAC, called *Spectrum-Conscious WiFi* (or SpeCWiFi for short). SpeCWiFi is evaluated on a testbed in our department, by developing an implementation based on MadWifi device driver [57].

Organization: This chapter is organized as follows. We present the preliminaries in Section 2.2, and formalize the DSA coexistence problem in Section 2.3. The proposed coexistence solution is presented in Sections 2.4 and 2.5, with implementation details in Section 2.6. Evaluation of SpeCWiFi is described in Section 2.7. The chapter concludes with Section 2.8.

2.2 Preliminaries

2.2.1 Assumptions and Notations

We assume that all SU devices in the SUG have a similar view on PU activity. This is reasonable because DSA protocols ensure that all SUs are on same footing w.r.t. spectrum conditions, e.g., via control channel sensing exchanges [18, 19]. However, this assumption is not strictly necessary. We also briefly address how to handle the case when such a coordination function is absent—resulting in SUs possibly having divergent views on incumbent activity.

We do not assume any type of coordination between the PUs and the SUs during their coexistence. This ensures that the proposed DSA Coexistence scheme can be widely deployed, even alongside legacy PU systems, and be more acceptable to licensee wireless operators.

The key terms used in this work are defined below.

- *Incumbent Detection Threshold (IDT)*: Weakest PU signal strength detectable by SUs.
- *Channel Detection Time (CDT)*: Maximum duration within which SUs must detect PU signals and halt their own transmissions.
- *Coexistence Period (CP)*: The duration during which a SUG coexists with the PUGs.

2.2.2 Role of Spectrum Sensing

Although the focus of this work is not spectrum sensing,[3] its role must be emphasized in the context of DSA coexistence. The effectiveness of DSA coexistence is highly dependent on how correct and timely is its knowledge about the underlying

[3]Only "in-band" spectrum sensing is relevant here.

channel conditions, especially PU activity. High-fidelity spectrum sensing requires *Quiet Periods* (QPs), during which no SU should engage in any transmission. A QP may vary from less than 1ms (using just energy-detection in high frequency bands) to 100ms or more (using feature-detection) [58]. With advances in sensing technology, QP duration is expected to decrease in future. Still, QPs induce a significant overhead in DSA if scheduled frequently. Our solution would only benefit from advances in the field of spectrum sensing.

Thus, the underlying tradeoff between spectrum sensing and performance necessitates intelligent scheduling of transmission and sensing, which is a key design goal of our DSA coexistence. It can be seen that the aforementioned tradeoff belongs to the well-known class *exploration vs. exploitation* problem, which has been studied in many other areas in computer science, statistics, and maths. However, the key constraint of *PU-safety* in a dynamically variable resource environment (i.e., the variability in spectrum white spaces) is a unique twist to this problem in DSA domain, which makes it a PSPACE-hard problem [37].

Though an external sensing infrastructure (e.g., sensing sensor network, or offline channel geolocation databases as used in [59] and recommended in [6] for TV spectrum) simplifies the sensing task, it must still be performed in real time for multiple reasons, e.g., to dynamically check for spectrum white spaces, detecting unexpected incumbent transmissions, and coexistence. While fixed database based sensing and DSA may work reasonably well for slow-varying TV spectrum, it can be very restrictive in wireless channels used for consumer communication, as channel utilization cannot be changed dynamically on demand or network load, even for PUs. Typically, such consumer wireless channels are also fast-varying in terms of availability and duration of white spaces.

2.3 Problem Statement

2.3.1 Motivation

As discussed in Section 1.1, spectrum surveys [2,3] show the existence of abundant time-domain spectrum white spaces, even in densely populated urban areas. However, current DSA coexistence schemes are not sophisticated enough to exploit most of these opportunities because of tremendous variability in their temporal characteristics. This aspect makes generic DSA coexistence problem is challenging because it must account for diversity in incumbent channel-access behavior. Further, any DSA coexistence solution must also account for the stringent requirements for incumbent protection.

While it is easy to achieve near-optimal balance of PU-Safety and SU-Efficiency criteria in slow-varying channels (such as UHF bands), it is difficult to accomplish in fast-varying channels. The key insight here is that interfering PU-SU transmission overlaps occur at a much higher rate in fast-varying channels, because of the burstiness of incumbent activity—PUs access the channel more frequently with much smaller ON and OFF durations. We demonstrate this problem through simple experiments.

2.3.2 Experimental Verification

To verify the existence and impact of SU-efficiency vs. PU-safety tradeoff, we conducted simple experiments using our implementation of PU and SU emulators (described in Appendix B). The PU follows ON/OFF channel access behavior with each ON/OFF duration being exponentially random. We changed the PU's average ON/OFF duration while keeping the the average availability of channel white spaces at 50%. We point out that ON/OFF model is a well-accepted model for incumbent behavior in DSA research [4, 39], and is also utilized in our problem formulation (Section 2.3).

In this setup, our DSA MAC implementation mirrors the current DSA coexistence

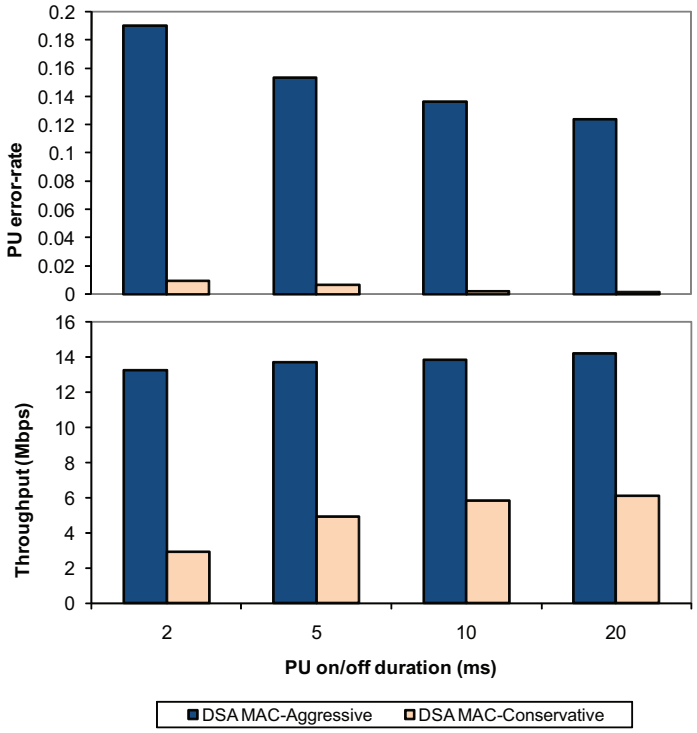


Figure 2.1: Performance with current DSA coexistence model.

strategy followed in the UHF spectrum: it halts transmission when the PU signal is detected (within a time-limit), and attempts to transmit when sensing declares the medium to be free of incumbents. To show that any straightforward modification (e.g., reducing time-limit for stopping secondary transmission) of this simple coexistence strategy is insufficient in fast-varying channels, we designed two types of DSA MAC—aggressive and conservative. Both MAC types differ in backoff time before starting transmission during an incumbent-free period, but are similar in all other respects.

In the aggressive MAC, SUs access the channel with a smaller backoff time. In this case, as shown in Fig. 2.3.2, interference to the PUs can be as much as 19% of its channel-access time. In contrast to the aggressive MAC, SUs access the channel with a large backoff time when using the conservative MAC. If a conservative medium-access approach is taken to reduce interference to the PUs, the error-rate would be much more acceptable ($< 2\%$). However, SU-efficiency then degrades by more than

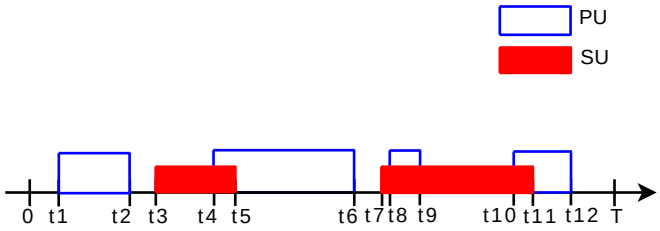


Figure 2.2: The blocks represent the medium access durations by either SU (shaded) or PU (unshaded).

75% as compared to the aggressive MAC.

In this experiment, the average fraction of available channel white space (the OFF periods) is the same (50%) for both fast-varying (e.g., PU ON/OFF=2ms/2ms) and relatively slow-varying channels (e.g., PU ON/OFF=20ms/20ms). The key difference is that it is much more fragmented in time-domain for the case of fast-varying channels. With current DSA coexistence schemes, such fast-varying channels will always be seen by SUs as occupied by the PUs and hence un-utilized, even though the channel is free 50% of the time. Clearly, conventional DSA coexistence, in both aggressive and conservative forms, is ineffective in balancing between PU-safety and SU-efficiency. In practice, there would be no choice but to use the conservative MAC, as PU-safety is paramount.

An option for the SUs is to switch to a different channel, as currently followed in UHF spectrum. However, as mentioned earlier, this is a task of frequency-domain spectrum management, which is orthogonal to time-domain DSA coexistence considered here. Nevertheless, changing channels would not eliminate the aforementioned problem, if similar PU behavior is exhibited on the new channel. Note that closely-spaced channels in the licensed spectrum, in general, exhibit similar characteristics as they are used for similar services.

2.3.3 Mathematical Formulation

The following formulations are in the context of a DSA coexistence duration, say $[0, T]$ shown in Fig. 2.2.

CGF description: To mathematically quantify the tradeoff between PU-safety and SU-efficiency—the two key requirements of DSA coexistence, we first model the active channel-usage intervals (CUIs) as a set of ordered pairs:

$$CUI = \{(t_i, t_j)\text{: channel use from } t_i \text{ to } t_j, t_i < t_j\}. \tag{2.1}$$

Each element of the set CUI represents a finite time interval when the channel was utilized. Also, if $(t_a, t_b) \in CUI$ and $(t_c, t_d) \in CUI$, then $t_a < t_c \implies t_b < t_c$, and vice versa.

We define two parameters:

- $I_{ps}(CUI_{PU}, CUI_{SU})$: *PU–SU Interference Factor*, or the maximum fraction of PUs' transmission time interfered by SUs' transmissions, $0 \leq I_{ps} \leq 1$.
- $U_s(CUI_{PU}, CUI_{SU})$: *SUs' Channel Utilization Factor*, or the fraction of time utilized by SUs ($0 \leq U_s \leq 1$).

For instance, in Fig. 2.2,

$$\begin{aligned} I_{ps} &= \frac{(t_5 - t_4) + (t_9 - t_8) + (t_{11} - t_{10})}{(t_2 - t_1) + (t_6 - t_4) + (t_9 - t_8) + (t_{12} - t_{10})}, \\ U_s &= \frac{(t_5 - t_3) + (t_{11} - t_7)}{T}. \end{aligned} \tag{2.2}$$

The *Coexistence Goodness Factor* (CGF) is a 2-dimensional metric defined as

$$CGF(CUI_{PU}, CUI_{SU}) = (I_{ps}, 1 - U_s). \tag{2.3}$$

CGF incorporates the cumulative effect of both PU-safety and SU-efficiency factors

during CP. CGF is much more comprehensive than, and a significant improvement over, the traditional DSA coexistence metric of absolute time-bound for limiting interference to PUs. The improvement comes for two reasons—(a) PU-safety is extended to include the cumulative impact of SU transmissions throughout coexistence period; (b) Overall SU-efficiency is also incorporated. Thus, a CGF-based DSA Coexistence protocol design will widen the spectrum for applying DSA, and be significantly more attractive to both PUs and SUs.

Note on PU–SU interference model: One of the fundamental characteristics of wireless communication is that detecting collisions while transmitting is extremely difficult (without external aid) due to very high transmit energy density around the transmitter. Thus, it is virtually impossible to ensure perfect PU-safety in DSA in current wireless communication model. Licensee network operators (or spectrum owners), who have paid huge amount of money for their ownership, would certainly prefer zero interference guarantee at all times from SUs. However, this is infeasible to provide in practice within reasonable cost margin with state-of-art technology. Realizing this, even the current regulatory guidelines for DSA in TV spectrum allows for up to 2 seconds of overlap anytime an incumbent access begins, which is acceptable to TV spectrum operators. Still, minimizing interference to PUs (even beyond stringent regulatory guidelines) must be a key goal for any DSA coexistence scheme to make DSA more acceptable and useful. CGF metric incorporates this objective comprehensively.

Channel White-Space Utilization Problem: The goal of a DSA coexistence is to minimize CGF. Since the two individual objectives in CGF conflict with each other, we formulate the following multi-objective optimization problem (MOP). For CUI_{PU} during $[0, T]$,

$$\min_{CUI_{SU}=\{(t_i,t_j):0\leq t_i<t_j\leq T\}} CGF(CUI_{PU}, CUI_{SU}). \tag{2.4}$$

The solution space of the above MOP consists of the possible channel-access schedules for the SU network. Eq. (2.4) can be shown to be Pareto-optimal, with the optimal value depending on CUI_{PU}. The optimal value for I_{ps} objective is always 0. The ideal solution corresponds to perfect usage of the channel by the SUs—100% utilization of the spectrum white spaces on the channel, with zero SU transmission during PU ON periods. In theory, it is simple to solve the optimization problem in Eq. (2.4) using the well-known MOP optimization techniques like the Aggregate Objective Function (AOF) or Normal Boundary Intersection (NBI) method. Thus, based on the optimized solution, a SU network can schedule its upcoming transmissions such that the CGF vector during the coexistence interval is minimum.

Need for a practical approach to DSA coexistence: In practice, it is difficult to optimize Eq. (2.4) for a number of reasons, arising due to lack of real-time knowledge about operating parameters during deployment. First, CUI_{PU} is most likely unknown to the SUs (PU–SU cooperation cannot be assumed). Second, DSA is inherently inefficient as it needs to schedule sensing (and other events) that prevents full utilization of channel white spaces. These two constraints actually make the DSA coexistence problem PSPACE-hard [37]—ruling out any low overhead optimal solution. Third, SUs need to schedule their transmissions in real time which is affected by other channel-related factors, thus preventing maximal utilization of spectrum white spaces.

2.4 SpeCWiFi

Our practical solution to the channel white-space utilization problem involves incumbents' channel-usage pattern estimation with joint sensing and transmission scheduling.

We present our approach in the context of SpeCWiFi—which enhances the 802.11 DCF with our DSA Coexistence scheme, while maintaining its distributed operation

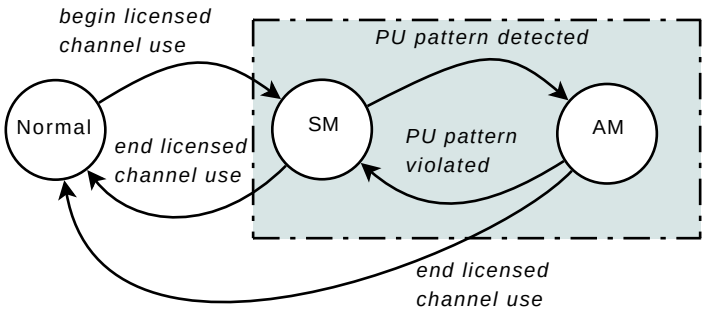


Figure 2.3: State-transition diagram of DSA coexistence.

semantics. SpeCWiFi serves as a concrete illustration of how to apply our proposed DSA coexistence protocol. Note that our goal is not to build a full DSA MAC protocol as in [17,18]. Rather, our focus is on the DSA coexistence problem, which is typically a component of the DSA MAC.[4]

Further, it must be emphasized that *the proposed DSA coexistence is generic* and can be incorporated as part of *any* DSA protocol even though we develop it for WiFi-type system for implementation and evaluation tractability. Also, for SpeCWiFi, SU–SU coexistence (or self-coexistence) is handled by the existing CSMA/CA mechanism in the 802.11 MAC.

2.4.1 Adaptive Dual-mode Licensed Operation

We propose a dual-mode DSA coexistence scheme, consisting of *Safe Mode* (SM) and *Aggressive Mode* (AM), as shown in Fig. 2.3. Normal mode corresponds to operations in unlicensed (or *home*) channels, where DSA coexistence does not apply.

2.4.2 Safe Mode (SM)

SM is the default mode of SUs when operating in licensed channels. Once a SU network enters a licensed channel, it starts operations in SM, and may switch to AM

[4]Issues such as when SpeCWiFi should switch between licensed and unlicensed operations or which channel to select, are unrelated to the PU–SU coexistence problem addressed here.

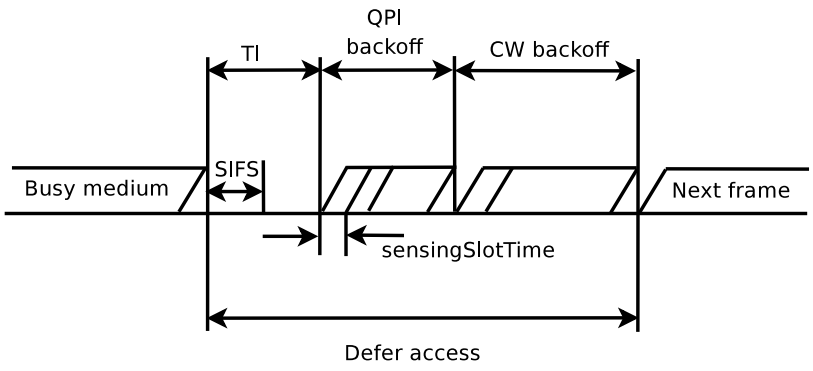


Figure 2.4: Channel access in the Safe Mode of SpeCWiFi.

when a PU channel-usage pattern is detected, as shown in Fig. 2.3. Conversely, the SU network will switch from AM to SM, if the expected PU channel-usage pattern is violated. The philosophy behind SM is to "transmit less, observe more." This allows SUs to continuously gather sensing information without too many time-gaps. A high-quality sensing time-series is crucial to the determination of PU channel-usage patterns (see Sections 2.4.4 and 2.4.5).

We define an *Atomic Packet Exchange* (APE) as a sequence of frame exchanges resulting in a complete transfer of a set of MSDUs from the sender. In SM, time-consuming APEs such as burst-type exchanges and prioritized access are prohibited in order to prevent SUs from using the licensed channel for a long time in one stretch. Regular APEs are allowed, with the condition that the APE duration must confirm to the DSA regulatory guidelines on incumbent detection.

Every APE is followed by a *Quiet Period Interval* (QPI), before the channel can be accessed for the next APE. Similar to 802.11 contention window variation [33], QPI varies according to:

$$QPI = QPW \times sensingSlotTime \tag{2.5}$$

where QPW is the quiet period window and takes an integer value in the range over the interval [1, QPW_{max}]. The minimum duration adequate for high-fidelity spectrum

sensing is indicated by *sensingSlotTime*, and its value is a fixed input derived from the sensing technology used.

QPW (or equivalently QPI) is varied based on recent sensing observations in order to adaptively balance the SU's need for sensing opportunities vs. data transmission. The initial value of QPW is QPW_{max}. For every QPI result indicating the PU's absence, QPW is reduced by half. Once QPW reaches 1, it remains at this value until it is reset. Thus, even in SM, data transmission can be frequent when PUs are not observed on the channel for a long time. If a PU is detected during the QPI, QPW is reset to QPW_{max}. QPI must then be re-initialized. Thus, a recent PU detection makes SUs wait longer before attempting to transmit even when the medium may be sensed to be currently free, as the PUs could likely be engaged in an ongoing communication session.

In wireless networks, after every packet transmission, a sufficient turnaround time is required for decoding and resetting interfaces (e.g., SIFS in 802.11). In SpeCWiFi, QPI follows the *Turnaround Interval* ($TI > SIFS$) after each APE, and proceeds with the CW backoff period as shown in Fig. 2.4. Further, although QPI is calculated individually by the SU nodes (using Eq. (2.5)), they converge to the same value and are scheduled at the same time. This is because all the nodes of a SUG have a consistent view about channel conditions in terms of PU detection (see Section 1.2), and also follow the same channel access model. Typically, the coordination component in most DSA MAC-PHY protocols ensures that nodes have a similar view of the channel (e.g., via control channel exchanges as in [18, 19], or beacons as in [21]). By leveraging this feature, the proposed SM operation is applicable to multiple collision domains.

We briefly address the rare case when explicit spectrum coordination is absent from the parent DSA MAC protocol in Section 2.5.

2.4.3 Aggressive Mode (AM)

In case there is a pattern of PUs' ON and OFF durations, SUs may not need to waste QPIs that could otherwise be utilized for data transmission. To achieve better CGF, AM attempts to exploit occurrence of PU channel-access pattern, which can exist due to: (a) explicit design in the incumbent wireless protocol to facilitate unlicensed coexistence (as in 802.16h [56] or upcoming 4G systems), (b) traffic characteristics in the incumbent network (as in TDMA-based licensed networks where some slots go unused [41]), (c) channel reservation semantics of the licensed channel's owner/operator [60]. The goal is to exploit even very short-term patterns (lasting few seconds).

In contrast to SM, the principle of AM is to "transmit more, observe minimally." In AM, the channel-usage pattern of PUs is known based on sensing observations gathered in SM. Details on how to estimate the PUs' usage pattern will be provided in Section 2.4.4.

In AM, QPIs are scheduled at frequency f_{qpi} in order to ensure the periodic sensing required to ascertain any out-of-pattern PU traffic. The frequency f_{qpi} must conform to the regulatory time guidelines in terms of detecting any PU transmission. Any unexpected detection of PU traffic will result in the PU channel-usage pattern violation, and the SUs switch back to SM. Since QPI is scheduled relatively infrequently in AM, the QPW value is fixed at QPW_{max} allowing maximum duration for each QPI to enable more reliable sensing.

2.4.4 Estimation of PU Channel-Usage Pattern

The sensing component of DSA provides information on whether incumbent activity has been detected at various time instants on the licensed channel. Using bits 1 (to indicate the PU presence) and 0 (to indicate the PU absence), the sensing observations can be represented as a binary time-series:

$$s = [s_1, s_2, \ldots, s_i, s_{i+1}, \ldots], s_i \in \{0, 1\}.$$

The series s has a bounded number of elements (N) over a finite time-window. An element s_i of the series corresponds to the sensing observation taken at time instant t_i. Series s constitutes the input available for PU channel-usage pattern detection.

Many well-known techniques (e.g., genetic algorithms) have been used for pattern recognition and trend analysis, especially in data-mining and machine learning. However, they are quite complex to implement at device level, and involve significant resource overhead. Further, such approaches typically require a high degree of training and thus cannot be deployed online, such as 2-D frequent pattern mining proposed in [3]. We ruled out other high-overhead techniques here (like FFT and autocorrelation), as MAC processing needs to be agile, real time, and must operate on limited memory and computational power.

Instead, we make use of *Approximate Entropy* (ApEn) [40,61] for pattern recognition. ApEn is a measure of regularity (or irregularity) present in a discrete sequence, e.g., binary sequences like s. Given a small number of observations, ApEn can be used to classify complex systems including deterministic and stochastic processes, without any additional information about system behavior. Hence, ApEn measure is well suited for analyzing PU channel-usage behavior. ApEn has been shown to be useful in diverse contexts, e.g., cardiovascular data analysis [62] and spectrum sensing [41]. A brief description of ApEn is provided in Appendix A.

2.4.4.1 Algorithms

Our pattern recognition method is based on parametrized decision-making. We present Algorithms 1 and 2 that jointly accomplish this task.

Algorithm 1 encodes efficient calculation of ApEn values for the sensing time-series s. The output of Algorithm 1 is $ApEn[L_{max}]$, an array of ApEn values. Algorithm 2 takes this array together with $ApEn_{thresh}$ as input parameters to decide, based on

Algorithm 1 ApEn calculations for s

Require: $s = [s_1, s_2, \ldots, s_N]$, L_{max}
Require: Maximum expected pattern length L_{max}
Ensure: $L_{max} + 1 \leq N$

Declare ApEn array $ApEn[L_{max}]$ {0-indexed}
Declare logarithmic correlation array $\Phi[L_{max}]$
$\Phi[0] \leftarrow 0$ {Initialize boundary condition}
$L \leftarrow 1$ {Initialize pattern length to 1}
while $L \leq L_{max} + 1$ **do**
  $\Phi[L] \leftarrow 0$
  Declare distance array $d[N - L + 1][N - L + 1]$
  for $i \leftarrow 1$ to $N - L + 1$ **do**
    for $j \leftarrow 1$ to i **do**
      if $i = j$ **then**
        $d[i][j] \leftarrow 0$
      else
        $d[i][j] \leftarrow \max_{k \leftarrow 1,2,\ldots,L}[|s_{i+k-1} - s_{j+k-1)}|]$
        $d[j][i] \leftarrow d[i][j]$ {Using symmetry of $d[i][j]$}
      end if
    end for
  end for
  Declare correlation vector $C[N - L + 1]$
  for $i \leftarrow 1$ to $N - L + 1$ **do**
    $C[i] \leftarrow 0$
    for $j \leftarrow 1$ to $N - L + 1$ **do**
      if $i \neq j$ && $d[i][j] \leq 0$ **then**
        $C[i] \leftarrow C[i] + \frac{1}{N-L+1}$ {Calculate $C_i^L(0)$}
      end if
    end for
    $\Phi[L] \leftarrow \Phi[L] + \frac{C[i]}{N-L+1}$ {Calculate $\Phi^L(0)$}
  end for
  $ApEn[L-1] \leftarrow \Phi[L-1] - \Phi[L]$ {Calculate $ApEn(L-1, 0, N)$}
  $L \leftarrow L + 1$
end while

comparisons with $ApEn_{thresh}$, whether a pattern is present in s. If $ApEn(L, 0, N) \leq ApEn_{thresh}$, a pattern of length L is present in series s. The length of the best recognized pattern is the output of Algorithm 2.

Algorithm 2 Pattern recognition decision-making

Require: $ApEn[L_{max}]$, $ApEn_{thresh}$

$ApEn_{min} \leftarrow \infty$ {Initialization and boundary cases}
$L_{pattern} \leftarrow -1$
$Found \leftarrow FALSE$
for $i \leftarrow 1$ to L_{max} **do**
  if $ApEn[i] \leq ApEn_{thresh} \&\& ApEn[i] \leq ApEn_{min}$ **then**
    $ApEn_{min} \leftarrow ApEn[i]$
    $L_{pattern} \leftarrow i$
    $Found \leftarrow TRUE$
  end if
end for
return $Found, L_{pattern}$

2.4.4.2 Correctness and Complexity

Algorithm 1 is easily proved correct as it is based on Eq. (A.1). The asymptotic worst-case runtime complexity of Algorithm 1 is $O(L_{max}^2 N^2 + L_{max}^3)$. When the length of the detected pattern is small, L_{max} is effectively a small constant. Hence, the asymptotic runtime is simply $O(N^2)$. Alternatively, if the length of the pattern is large (or $L_{max} \sim N$), then the runtime is $O(N^4)$. The asymptotic space complexity of Algorithm 1 is $O(N^2 + L_{max}^2)$, or $O(N^2)$ in the worst case.

The correctness of Algorithm 2 is easily proved by using contradiction. Algorithm 2 is a linear-time algorithm with the worst-case complexity as $O(L_{max})$. For small values of L_{max}, it is essentially a constant-time algorithm. Also, if $L_{max} \sim N$, then the runtime complexity is $O(N)$. The asymptotic space complexity of Algorithm 2 is the same as its runtime complexity.

In conclusion, both algorithms have simple polynomial asymptotic runtime and space requirements. The impact of design parameters is evaluated in Section 2.7.

2.4.4.3 Applying the algorithms to AM

Given the PU-channel usage pattern of length $L_{pattern}$ (from Algorithm 2), the SUs estimate the start-time and duration of each PU-free and PU-busy periods (relative

to the current time) as follows.

Let t_i be the time instant when sensing observation s_i was made. The key insight is that the most recent sensing observations $S_{pattern} = \{s_1, s_2, \ldots, s_{L_{pattern}}\}$ will repeat over the next $T_{pattern} = t_{L_{pattern}} - t_1$ time-interval. Thus, the PU-free/busy periods are estimated based on the elements of $S_{pattern}$ and the difference between their observation time. For instance, if the values of successive pattern elements $\{s_i, s_{i+1}\}$ are $\{0, 0\}$, then PU-free duration of $t_i - t_{i+1}$ is predicted. Similar is the case for $\{1, 1\}$ where PU-busy period is predicted. For observations of type $\{0, 1\}$ or $\{1, 0\}$, the transition is assumed to be midway between the individual observation times. Here, the exact transition times depend on the sensing frequency and sensing duration, and may not be known. Thus, any conflict during the initial phase of such transitions are ignored.

As an estimated pattern may not be 100% accurate, there must be a reasonable margin for error-tolerance. Any mismatch in prediction and expectation of PU-free/busy period results in a probabilistic switch to SM based on $ApEn_{thresh}$ value. Thus, if the fraction of mismatches observed surpasses $ApEn_{thresh}$, the SUs switch back to SM.

2.4.5 Sensing: Quality vs. Quantity Tradeoff

As noted earlier in Section 2.2.2, sensing information is crucial for DSA coexistence, especially for the PU channel-usage pattern estimation. Apart from a reliable sensing technology, the overall sensing data should be of sufficiently high granularity to detect patterns over short durations. Thus, frequent scheduling of sensing quiet periods is key to high quality sensing information.

In SM, the sensing is quite frequent—after every APE, and hence, PU activity can be captured at fine granularity. However, frequent sensing quiet periods can incur a high overhead for communication. Thus, sensing is scheduled on a *as-needed* basis

when the PU channel-usage pattern is known. In this way, the dual-mode DSA Coexistence ensures a proper tradeoff of sensing information quality with performance.

2.4.6 Remarks

Practical considerations: In practice, additional minor changes to DSA MAC protocols may be required in order to incorporate our approach. For instance, to facilitate mode management (e.g., during node association to an AP), the DSA MAC should include mode information in control packets. Our SpeCWiFi implementation includes mode information in the MAC header.

Why pattern detection is useful and applicable?: As discussed earlier, a reliable estimate of incumbent channel-usage pattern and corresponding transmission scheduling can dramatically improve SU-efficiency and PU-safety metrics. This is especially important for channels exhibiting spectrum white spaces of very small scale (order of few milliseconds), where traditional reactive coexistence approach does not scale. Studies, such as [3], have shown that licensed channels (20MHz–3GHz) is characterized by highly correlated time-domain white spaces in numerous spectrum regions (on average > 0.7), which again enhances the usefulness of pattern estimation. Majority of such patterns are seen to last for several minutes or more. Besides the implicit usage pattern present in current licensed spectrum, certain upcoming licensed protocols (like IEEE 802.16h) feature explicit regular white spaces.

On interference to PUs: Note that it is extremely difficult to guarantee zero interference to PUs due to variability and lack of sufficient information during deployment, as discussed in Section 2.3. The PU channel-usage pattern may change after some time, causing some interference while the mode is adjusted to SM. Extremely small time-scale PU transmissions may also be missed. Further, even in SM, there may be some interference to incumbents, especially if the incumbent channel-access behavior is highly random.

Therefore, our proposed DSA coexistence scheme tries to achieve the next-best goal, i.e., minimizing such interferences. In the process, our solution can provide the average operating benchmark levels for wireless operators and consumers. In our evaluation (see Section 2.7), average interference to incumbents is found to be less than 2% of coexistence duration when PU channel utilization is 50%. Similarly, this approach may not provide 100% SU-efficiency in practice, but it is found to be sufficiently high in all cases ($> 90\%$ on average).

2.5 Boundary Environment

Albeit not the primary focus of this work, we briefly discuss safeguards to deal with the scenario where the underlying DSA MAC-PHY protocol does not provide sensing information coordination feature. Thus, SUs in the network may conflict on the incumbent's presence. This could occur, for example, in a *boundary environment* condition where the incumbent signal strength is very weak around the SU network such that certain SUs can sense the PU signal, while others cannot. Also, channel sensing errors cannot be avoided at all times, leading to such discrepancies.

Typically, this is handled through explicit coordination functions in a DSA MAC protocol. Most protocols use a control channel to disseminate control information. In the absence of an explicit coordination, we propose the following conflict resolution policy.

Considering the importance of PU-safety, conflicts between AM and SM are resolved in favor of SM. For instance, a node (if in AM mode) switches to SM when it receives (or observes) any packet indicating that the sender is in SM. Packets should carry mode information in order to quickly identify and resolve conflicts when they occur.

As a further enhancement, on-demand control packets (similar to RTS/CTS in 802.11) can be used by SUs (including passive SUs) for quick indication of the PU's

presence, with minor impact on PU communication. A SU node broadcasts a *PU On* (PUO) packet when the PU is sensed on the channel, followed by a *PU Ceased* (PUC) packet when the PU ends the transmission. To avoid overheads, PUO-PUC packet exchange must only be undertaken when a SU node detects (or expects) any SU transmission while it has knowledge of simultaneously ongoing PU activity on the channel—indicating a conflict among nodes in the SUG. Further, a randomized delay should be used to prevent multiple SUs from broadcasting PUOs at the same time.

The aforementioned coordination approach exploits the observation that it is very likely for SU control packets to be received without errors. This is because the PU signal strength is quite low (or absent for some SUs), especially in a boundary environment, and SU signal strength will be comparatively much higher.

2.6 Implementation

We have implemented SpeCWiFi by augmenting the open-source 802.11 driver MadWifi (madwifi-0.9.4) [57] to develop a software prototype that operates Atheros 802.11 wireless cards. The SpeCWiFi implementation consists of its state machine together with its access model (see Figs. 2.3 and 2.4). The QoS control bits in the standard 802.11 MAC header are used to carry the DSA coexistence mode information. RTS/CTS packets emulate PUO/PUC packets that are used in a boundary environment (see Section 2.5). More details on the implementation can be found in Appendix B

Ideally, many of the SpeCWiFi features should have a hardware systems-on-chip implementation for precise and real-time MAC operations, similar to the current network interface cards available in the market. However, the absence of any suitably-priced hardware with required performance led us to implement SpeCWiFi in the driver. Thus, SpeCWiFi operations may not be as quick as they can be, since the driver has to share the processor with other processes. While the implementation

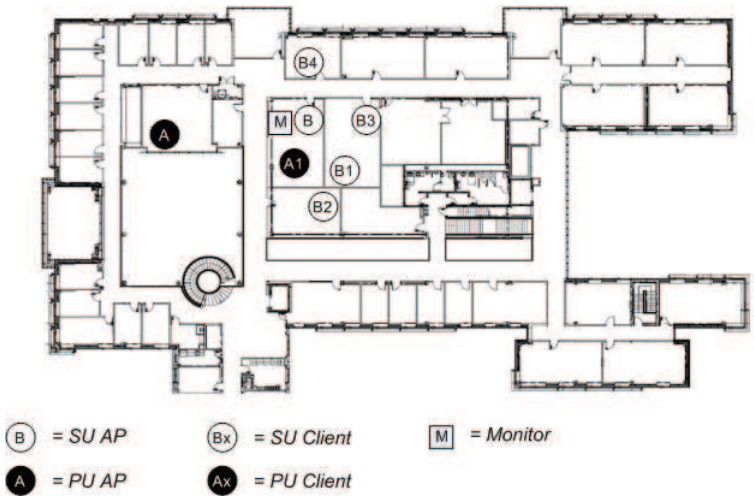


Figure 2.5: SpeCWiFi testbed in the building floor.

platform is not the best for SpeCWiFi, it is found to be more than adequate for analyzing performance trends in realistic deployments.

Again, we emphasize that even though our prototype implementation is based on a WiFi-type platform, the proposed DSA coexistence is applicable to any DSA protocol and across most licensed spectrum.

2.7 Evaluation

2.7.1 Testbed Setup

Our testbed was set up according to the system model (see Fig. 1.3). The setup consists of one SU network and one PU network. Note that single PU network is sufficient to emulate full range of incumbent channel activities and patterns. Similarly, single SU network (consisting of multiple devices) is sufficient, as our focus is on PU–SU coexistence rather than self-coexistence. As mentioned earlier, self-coexistence (SU-SU coexistence) is easily achieved through existing mechanisms, e.g., DCF in 802.11 MAC. For microbenchmarks, only a single AP–client pair is active in the SU network, while multiple pairs are used for macrobenchmark experiments. The machines are Dell Inspiron 600m laptops with Linksys A+G wireless cards, and running

Ubuntu 8.04 Linux (2.6.24-23 kernel). We used 802.11a channel 36 (5.18GHz) for our experiments as it was found to be free of other interfering devices in our experimental setting.

`Netperf` with saturated UDP stream is used to fully stress out the prototype implementation in microbenchmark experiments, while traces are utilized to study coexistence performance in a more realistic setting. The default sensing granularity (or *sensingSlotTime*) used is 1ms and the default PU pattern consists of exponentially random ON/OFF = 5ms/5ms (on average) durations.[5] Although there is a regular pattern in PU channel-usage (reflecting the temporally correlated white spaces in licensed spectrum), the ON/OFF periods are themselves random. Other default parameter values are: $N = 100$, $L_{max} = 50$, $ApEn_{thresh} = 0.1$, $f_{qpi} = 0.5s^{-1}$, and $QPW_{max} = 10$.

2.7.2 Performance Metrics

CGF (in terms of I_{ps} and U_s[6]) is the main metric of evaluation of the proposed coexistence schemes via SpeCWiFi. We also show the overall end-to-end performance in terms of throughput achieved, where needed.

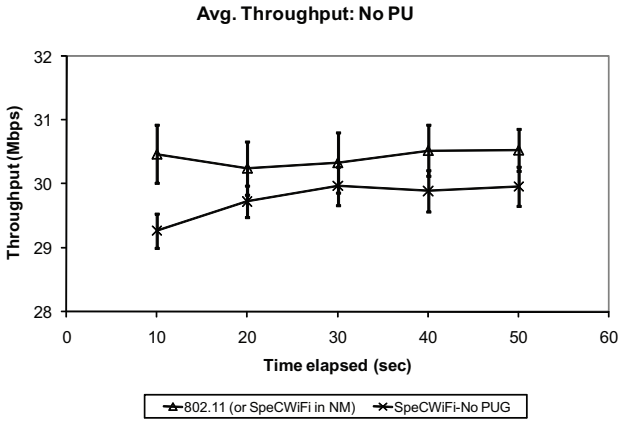
2.7.3 Results and Discussion

In every CGF comparison plot, $U_{s_{max}}$ represents the maximum SU utilization and $I_{ps_{min}}$ represents the minimum interference to PUs over the coexistence duration in that setup. Achievement of $U_{s_{max}}$ and $I_{ps_{min}}$ implies perfect DSA coexistence. Note that $I_{ps_{min}}$ is always 0.

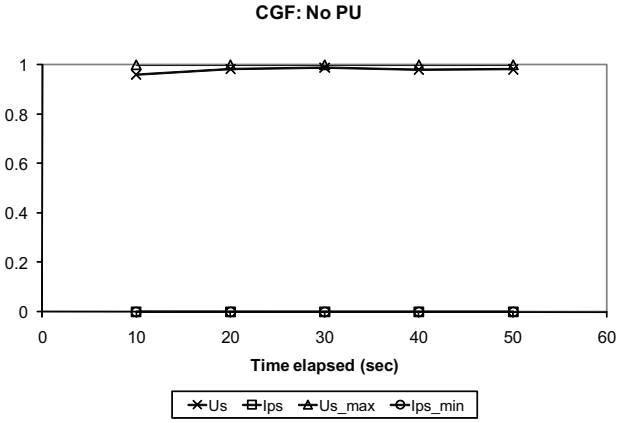
First, we discuss the important microbenchmark experiments and their results.

[5]Results for uniformly random and log-normally distributed ON/OFF durations are found to be similar, and omitted due to lack of space.

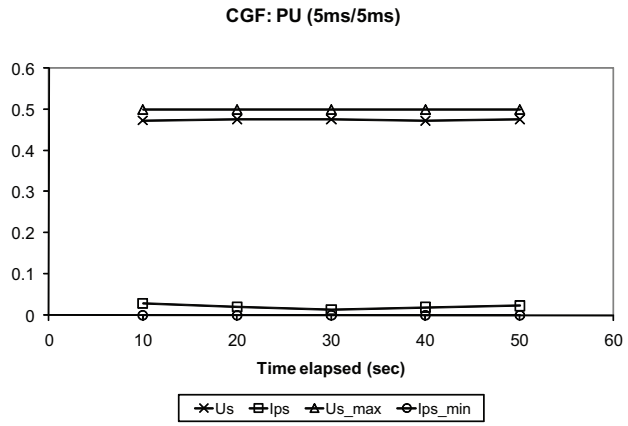
[6]We plot U_s instead of $1 - U_s$ to simplify plot visualization and to show its direct correlation to the throughput.



(a) Throughput (No PU)



(b) CGF (No PU)



(c) CGF (with PU)

Figure 2.6: Performance variation over time.

2.7.3.1 Overhead characterization

To capture the run-time overhead, the testbed was run with PU turned off. This scenario also establishes the performance benchmark when PUs are not present on a

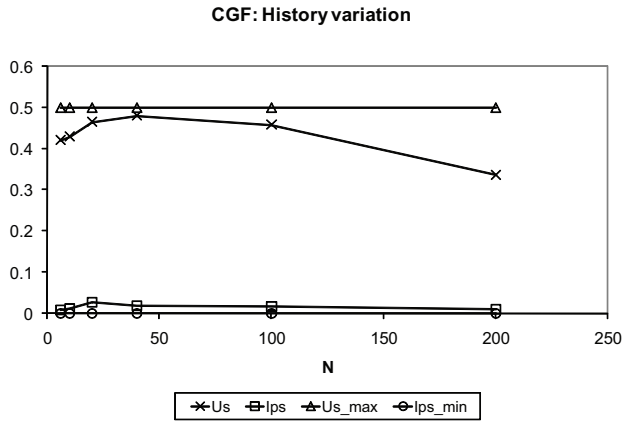
licensed channel for long durations (e.g., in slow-varying channels). Fig. 2.6(a) shows the average throughput obtained with a 95% confidence interval over every 10s for the first 50s. The initial throughput is slightly lower than the later values for SpeCWiFi, as it starts out in SM and takes few hundred ms to switch to AM. The throughput quickly stabilizes to around 29.5Mbps as seen from the graph. Overall, SpeCWiFi throughput is 0.6Mbps less than native 802.11 throughput in an empty channel. This loss in performance ($\approx 2\%$) is attributed to the computational overhead, e.g., timer routines and ApEn calculations. As pointed out earlier, the overhead should be lower in a hardware-based implementation where the access model (including sensing computations) does not have to contend for the processor time.

Fig. 2.6(b) shows the same result as in Fig. 2.6(a), but in terms of average CGF. U_s is found to be very close ($\approx 98\%$) to $U_{s_{max}}$, while I_{ps} is trivially 0.

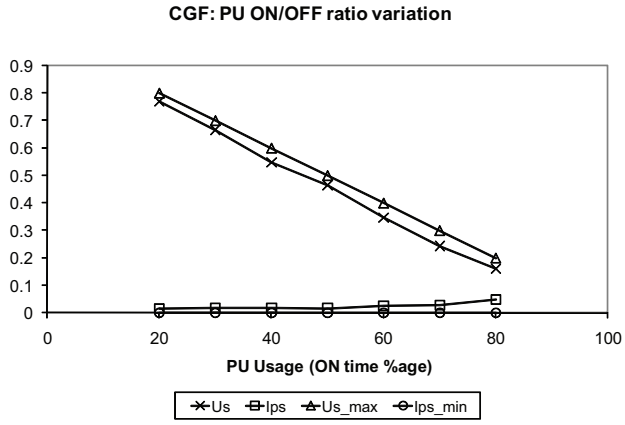
Fig. 2.6(c) shows the average CGF seen over every 10s interval, when PU is present on the channel with 50% channel-usage (average PU ON/OFF=5ms/5ms). SpeCWiFi manages to achieve an average of 96% utilization ($U_s \approx 0.48$ or 14.4Mbps throughput) of the available channel white-spaces, with less than 2% rate of interference to PUs.

2.7.3.2 Impact of parameters

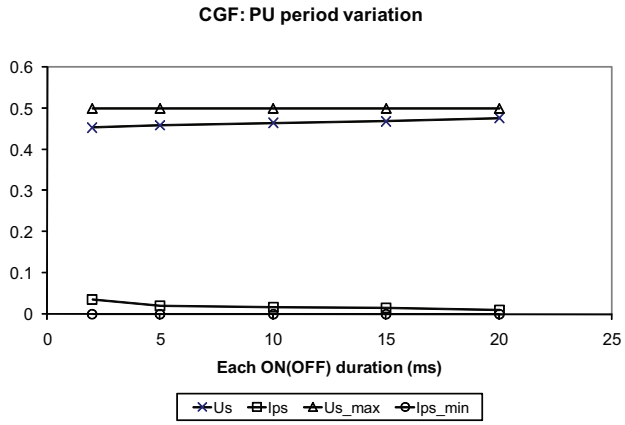
Fig. 2.7(a) shows the impact of history size $N(= 2L_{max})$ on CGF. The sensing granularity is 1ms. For very low values for history ($N < 20$), CGF is found to be significantly lower—with high I_{ps} and low U_s. The main reason for this is an insufficient history window to capture the full PU pattern (5ms/5ms ON/OFF), resulting in frequent mode switches. Similarly, the performance is poor when the history size is large (beyond ≈150). However, in this case we observe that the computational overhead is the bottleneck, highlighting the importance of low computational complexity of pattern estimation algorithms. The optimal value for history size in terms of L_{max}, as seen from Fig. 2.7(a), is around the size of the PU pattern. The optimal



(a) History Window



(b) PU channel-usage



(c) PU pattern

Figure 2.7: Impact of parameters and incumbent behavior I.

history size must be greater than the total PU pattern window (≈20ms in this case). Thus, prior knowledge about the range of expected incumbent patterns in a spectrum

region (e.g., through an external database as in [63]) may be useful in dynamically updating L_{max}.

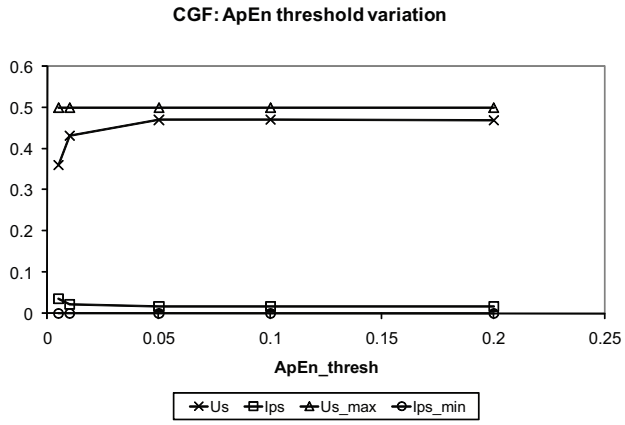
Fig. 2.7(b) shows the impact of varying PU channel-usage pattern in terms of its ON time percentage. One ON+OFF pattern lasts for 20ms. SpeCWiFi is shown to be consistent in achieving high U_s and low I_{ps}. I_{ps} is seen to be somewhat higher when the PU ON time fraction is very high. This observation suggests that in such cases, it may be better to switch to a freer channel. However, such a decision is up to the spectrum management of DSA, which is outside of the scope of this chapter.

Fig. 2.7(c) shows the impact of varying PU period in terms of its ON/OFF duration, while keeping the overall PU channel-usage fixed at 50%. The graph shows that CGF is better when each ON/OFF duration is larger. However, SpeCWiFi is able to keep CGF low ($U_s = 0.44$, $I_{ps} = 0.04$) even in the presence of very fast-varying PU with average ON/OFF period of 2ms. Clearly, SpeCWiFi improves coexistence performance significantly in comparison with the traditional DSA (see Fig. 2.3.2)—85% improvement in PU-safety and 150% gain in SU-efficiency.

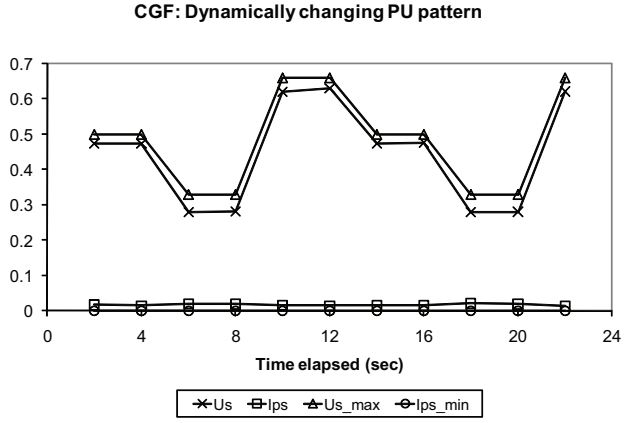
Fig. 2.8(a) shows the impact of $ApEn_{thresh}$ parameter. It shows that when a reliable PU pattern (5ms/5ms in this case) exists, there is little effect of varying the ApEn threshold. According to our observations, when there is high regularity in the sensing time-series, ApEn values tend to be very low (< 0.01), while rising rapidly on irregularity, which reflects its logarithmic nature. This fact is visible in the graph, where performance degrades only when $ApEn_{thresh}$ values are very small. In such a setting, slight fluctuations in sensing time-series may lead to mode switches (and associated overheads). Hence, keeping $ApEn_{thresh}$ value very low is not recommended.

2.7.3.3 Dynamic and random PU pattern

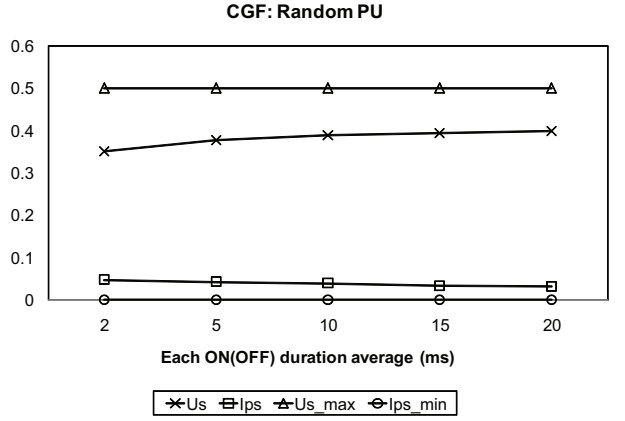
Fig. 2.8(b) shows the adaptability of SpeCWiFi when the PU channel-usage pattern changes dynamically. In this scenario, the average PU ON/OFF duration changes



(a) $ApEn_{thresh}$



(b) Dynamic PU



(c) Random PU

Figure 2.8: Impact of parameters and incumbent behavior II.

from 5ms/5ms to 10ms/5ms to 5ms/10ms every 4 seconds in a cyclic fashion. This result shows that SpeCWiFi can quickly adapt to incumbent pattern changes.

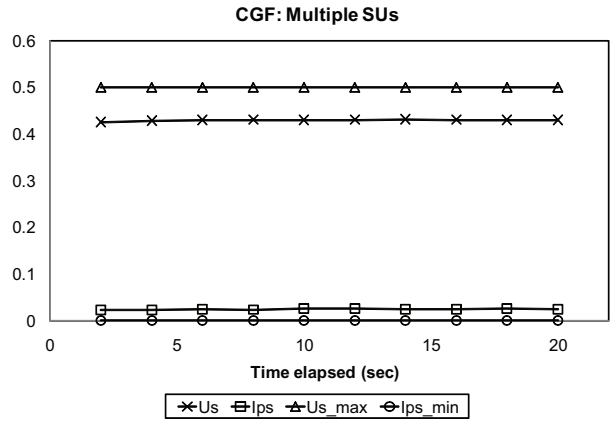


Figure 2.9: Performance with multiple nodes in SU network.

Despite the proposed DSA coexistence scheme exploiting the existence of pattern in PU behavior, it manages to achieve good performance in the presence of PUs with no fixed channel-access pattern, i.e., with random time-domain white-spaces. Fig. 2.8(c) shows SpeCWiFi performance when the PU channel-usage pattern itself is uniformly random, i.e., the exponential distribution parameters for average ON/OFF interval are not fixed but uniformly random between 0-10ms (average = 5ms) for each ON or OFF duration. As seen from the graph, SU-efficiency is lower as compared to periodic PU scenarios. SpeCWiFi is unable to find a consistent pattern from its sensing observations and hence, remains mostly in SM. Also, PU-SU collisions increase when incumbent exhibits random behavior. However, SpeCWiFi still achieves good PU-safety ($I_{ps} < 4\%$), with reasonable SU-efficiency (average $U_s \approx 0.40$ or 80%). Similar results are obtained when the incumbent accesses the channel using other distributions to randomize the access pattern.

2.7.3.4 Macrobenchmarks and real network traces

Previous experiments provided SpeCWiFi performance microbenchmarks with a single sender-receiver pair. Fig. 2.9 shows SpeCWiFi performance when there are 5 contending SU transmitters (4 clients+1 AP) constituting the SU network, with aver-

age PU ON/OFF pattern as 5ms/5ms. This experiment is intended to test distributed operation of the proposed DSA coexistence scheme. Self-coexistence among SUs is managed through CSMA/CA (DCF). During the experiment 2 clients were mobile, and sometimes moved beyond the range of PU transmission for short durations (∼6s) to generate the boundary environment situation. As seen from the graph, the overall PU-safety is similar to the single transmitter scenario. The overall SU-efficiency is slightly lower, mainly because of handling the frequent boundary situations when the SU network switches to SM.

For a similar topology setup as above, Fig. 2.10 shows the average CGF using publicly available traces (from actual deployments of Wifi/WiMAX systems) to generate real network traffic load as well as licensed channel characteristics. Here, SUs generate packets for 30s based on snapshots of the traces for OSDI 2006 WiFi traces [64]. The packets are backlogged at the transmitter to produce a saturated stream at the channel-level in order to stress our coexistence scheme. SpeCWiFi manages to keep PU interference low (<1.8% of channel-time) with high rate of utilization (≈85%), in the presence of periodic incumbents (PU ON/OFF = 5ms/5ms or 10ms/10ms), as seen in Fig. 2.10.

To emulate consumer licensed network, we generate two scenarios of PU channel activity—(a) based on WiFi traces from the hotspots in a cafeteria environment [65], and (b) using WiMAX traces from urban deployment (campus, subway, bus) [66]. Again, SpeCWiFi is found to provide good DSA coexistence performance (Fig. 2.10). We found that the cafeteria traces provided a high degree of opportunities (or time-domain white spaces), despite random channel accesses of short and varying durations interspersed in between. However, opportunities in urban WiMAX cases is lower ($U_{s_{max}} < 0.4$). Still, sufficient temporal patterns exist in such channels (which confirms the observations in [3]) that allows for viable DSA coexistence using SpeCWiFi, as seen from CGF values in Fig. 2.10. Even for urban WiMAX case, I_{ps} is lower

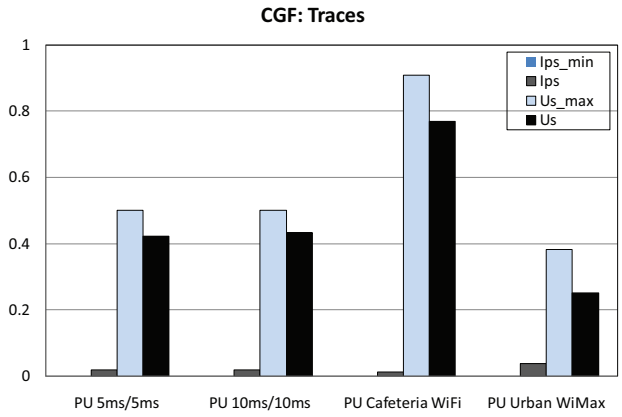


Figure 2.10: Performance with traces. $I_{ps_{min}} = 0$ in all cases.

than 4%, with $U_s \approx 0.26$ or 65% of available opportunities. This result indicates the ability of SpeCWiFi to perform well in realistic deployments.

2.7.3.5 Remarks

Many results have been omitted due to lack of space, but we briefly mention some important observations. Timing analysis shows that with default values of parameters (see Section 2.7.1), about 85% of the time is spent in AM. Also, with 50% channel usage by PUs, TCP stream utilization (U_s) is around 0.32 ($\approx$ 9Mbps), with interference to PUs less than 2%.

In conclusion, the results show that SpeCWiFi achieves efficient and reasonably safe DSA coexistence despite diversity in incumbent characteristics, up to millisecond time-scale in available white spaces.

2.8 Concluding Remarks

This chapter explored the feasibility of achieving general, safe, and efficient time-domain coexistence of SUs on an unlicensed basis along with PUs in a licensed channel. Key shortcomings in current PU–SU coexistence mechanisms, especially while operating in a licensed spectrum region characterized by fast-varying PU access, were iden-

tified. A simple, yet generic formalization of the coexistence problem was presented while introducing the Coexistence Goodness Factor (CGF) as the key coexistence performance metric. The proposed coexistence solution consists of low overhead algorithms for PU pattern recognition based on Approximate Entropy (ApEn), and the corresponding transmission scheduling. A dual-mode MAC operation strategy was introduced to enable its practical incorporation in real systems. An implementation based on 802.11 MAC—called *Spectrum-Conscious WiFi* (SpeCWiFi), was built and evaluated. Our experimental evaluation has shown that SpeCWiFi performs well (SU utilization 96+% with interference to PUs less than 2%, for 50% PU channel-usage), indicating the feasibility of expanding DSA-based coexistence to the relatively untouched licensed spectrum bands. The results also present a basis for developing a business model for unlicensed operation.

CHAPTER III

Analysis of QoS Provisioning in DSA Networks: A Case Study

3.1 Introduction

The DSA protocols proposed thus far aim at improving link layer throughput via opportunistic spectrum utilization. Nevertheless, high throughput at link layer gained from DSA may not translate into better perceived QoS at the application level. This is because DSA is inherently disruptive in nature. DSA causes high degree of wireless resource fluctuations and traffic interruptions due to its fundamental functions, like spectrum sensing and channel-switches (see DSA's functional model in Section 1.1. Current research lacks insights on DSA's impact on application QoS performance.

On the other hand, consumer networking applications are becoming increasingly dominated by different types of traffic that require stricter QoS guarantees (in terms of sustainable data-rates, delay bounds, and so on) as opposed to plain best-effort service. Examples of such applications include multimedia streaming and interactive video, real-time traffic mapping and updates,gaming, and so on.

Consequently, whether and how (in terms of complexity and cost) a DSA protocol and the overall CR system [1] can provide adequate QoS support is a major test for

[1] CR system refers to the complete set of technologies, including MAC-PHY protocols, involved in realizing the CR/SDR function—DSA in this case.

the viability and success of DSA, especially in consumer-oriented wireless systems. In this chapter, we provide an analysis based on a case-study to characterize the key characteristics needed for a CR communication system to meet specific application/service QoS requirements. Specifically, the goal is to provide insights into QoS performance in Dynamic Spectrum Access Networks (DSANs) by investigating the impact of design choices for a personal/portable CR system that must be able to support high quality multimedia streaming over UHF frequency channels.

Based on our study, we present the basic MAC features for supporting applications with demanding QoS requirements in DSA. We propose new techniques specific to DSA, as well as, adapt best practices in conventional MAC design that are especially useful in DSA domain. The suggested QoS-provisioning characteristics are illustrated through the design of *QoS-Provisioned DSA Protocol* (QPDP). QPDP is designed for indoor CR systems operating in UHF channels. QPDP can serve as a model for basic QoS provisioning of any DSA protocol. We also provide the proof-of-concept of our ideas inherent in QPDP via simulation, illustrating full quality HDTV streaming in home environments. The simulation results show the effectiveness of QPDP in QoS provisioning, and reveals the impact of key design components like managing sensing schedule and channel switching mechanism in minimizing negative QoS impacts of DSA.

This chapter is organized as follows. Section 3.2 provides the motivation for this work. Section 3.3 describes the specific platform for our case-study, while Section 3.4 provides the QoS model we consider in this work. Section 3.5 briefly illustrates our proposed QoS-centric cognitive MAC, namely QPDP. Details on QPDP's key QoS provisioning schemes are provided in Sections 3.6 and 3.7, and we discuss the evaluation results in Section 3.8. Section 3.9 concludes the chapter.

3.2 Motivation

This work is motivated by commercial necessities as well as technical challenges. From a commercial point of view, QoS support in CR systems is critical to ensure its success in consumer wireless market. From a technical perspective, the effort is motivated by the lack of awareness into QoS achieved by contemporary DSA and the challenge of designing QoS-provisioning schemes for DSA. As a first step, we intend to provide insights from a case-study of implementing QoS provisioning mechanisms in achieving the QoS goals of a realistic DSA-capable CR system. The specific CR system under consideration is a system for HDTV streaming over UHF bands in home networks.

Providing QoS Provisioning Schemes in DSA: In general, it is difficult to provide reliable QoS in wireless networks due to high degree of variability in wireless conditions. However, it is even more challenging for a DSAN to provide QoS support due to the additional interference from incumbents and the side-effects introduced by the underlying DSA protocol. Disruptions from fundamental functions involved in DSA protocols can make deployment of QoS-sensitive applications infeasible. For example, to maintain viability of DSA (including incumbent protection), the channels must be sensed frequently. This involves quiet periods, which disrupt the ongoing communication traffic and may lead to violation of QoS requirements. It should be noted that DSA, by virtue of its ability to dynamically select and utilize better channels, has a key advantage in providing sufficient resources for sustainable QoS in wireless systems. However, DSA operations must be properly managed in conjunction with suitable safeguard mechanisms in order to provide sufficiently reliable QoS. Our proposed solution QPDP incorporates features to achieve this very goal.

Realizing QoS in a Consumer DSAN: In addition to proposing generic QoS provisioning schemes, this work also aims to investigate their practical application to suit specific deployment environments. For this, we consider an indoor DSAN oper-

ating in UHF (TV) spectrum band. UHF is the first spectrum block to be approved for unlicensed operations by FCC [6]. This ruling allows unlicensed operation in TV bands for both fixed and personal/portable devices with DSA capabilities. However, TV channels have very narrow spectrum-width—only 6MHz. Thus, supporting high bandwidth data traffic in narrow TV bands is a significant challenge in itself. For example, full HDTV streams require around 20Mbps bandwidth with packet error rate (PER) less than 188 bytes/sec. Such QoS-sensitive multimedia traffic is expected to dominate indoor (especially home) wireless networks. Therefore, DSA QoS design must also take a practical and context-oriented view of the specific CR system. Such a system-oriented QoS provisioning approach will show substantial QoS benefits when deployed. This work illustrates this conclusion through the proposed DSA protocol QPDP, which exhibits several QoS provisioning features to accommodate the specific deployment platform considered here.

3.3 CR System Specifications

We consider a secondary device which is equipped with only one CR for simplicity and cost-effectiveness. The CR system's primary task is DSA, and it operates in 512-698 MHz frequency range, or UHF channels 21-51, excluding channel 37. A secondary device can operate on one 6MHz wide UHF channel, at any given time. The CR Systems constitute a home/office wireless network, where high quality multimedia streams are exchanged between the devices, e.g., between central storage unit to the projector.

The incumbents on the UHF band are TV broadcasting services and wireless microphones. The secondary CR systems must meet a set of performance parameters to protect incumbents according to regulatory requirements. It must be able to detect the presence of an incumbent signal stronger than the *Incumbent Detection Threshold* (IDT) within the *Channel Detection Time* (CDT) with a success probability greater

than or equal to the *Probability of Detection* (PD), and with false alarm probability lower than or equal to the maximum *Probability of False Alarm* (PFA).

Furthermore, the *Channel Move Time* (CMT) defines the amount of time the secondary system has to vacate the channel once an incumbent is detected, and the *Channel Closing Transmission Time* (CCTT) limits the amount of transmission time allowed to the secondary system once an incumbent is detected. Actual values for these parameters depend on regulatory directives. Following the guidelines defined in FCC's first-rule-and-order for personal/portable devices [67], we assume CDT = 10s, CMT = 2s, CCTT = 200ms, IDT= -107dBm (for wireless microphones), IDT = -116dBm (for TV broadcasts), PD = 90% and PFA = 10%. Clearly, hardware constraints and DSA regulation requirements, together with application QoS demands, impose significant challenges on overall CR system design, particularly on the underlying DSA MAC-PHY protocol.

3.4 QoS Model

As mentioned earlier, we aim at QoS provisioning for high quality multimedia traffic (e.g., HDTV streaming) in this work, which is a significant and ever growing fraction of overall network traffic. To support such application traffic on DSA networks, our goal is to ensure *better-than-best-effort* QoS [68].

Note that *absolute* (or *guaranteed*) real-time QoS is extremely difficult to provide in wireless communications [69,70], and even more so in DSA (as discussed in Section 3.2). Further, multimedia streaming and interactive applications, despite being highly QoS-sensitive, are flexible and adaptive to short-term network degradations. They can scale their fidelity (e.g., by changing coding schemes on the fly) and are loss-tolerant, up to a certain extent, based on the QoS feedback from the network. Therefore, almost all such applications and services utilize RTP over UDP, rather than TCP. Such applications perform optimally with better-than-best-effort QoS provisioning

which is relatively simpler to provide.

Still, such QoS support must accept diversity of QoS demands, and also ensure adequate long-term resources (e.g., application bandwidth, delay-bounds) for the most stringent QoS requirements expected during operation—HDTV streaming in this case. QPDP design attempts to meet this goal in DSA protocols by providing adaptive better-than-best-effort type of QoS through mechanisms like channel reservation and intelligent sensing schedule

3.5 QPDP Overview

In this section, we provide a brief insight into the proposed DSA protocol, called the *QoS-Provisioned DSA Protocol* (QPDP). The design philosophy of QPDP is not to invent a completely new DSA MAC protocol, but rather to adapt an existing MAC protocol to incorporate DSA-specific features together with QoS support. Such an approach allows us to exploit existing developments and innovations in this area. We choose WiMedia [71] as our base MAC protocol due to its salient features for QoS provisioning, as well as, mobility and coexistence support.

In order to provide adequate QoS support for DSA, we distinguish between fine (low-level) and coarse (high-level) communication aspects at MAC level, and target both for QoS provisioning. For instance, reservation-based medium access provides QoS support at fine granularity (i.e., at packet level). On the other hand, network and spectrum management protects the overall connection stream.

As shown in Fig. 3.1, QPDP is logically divided into *Lower MAC* and *Upper MAC* functions. The Lower MAC is adapted from WiMedia MAC, and is characterized by distributed reservation-based channel access. The Lower MAC is mostly responsible for routine MAC operations, such as superframe synchronization and frame processing over a channel. The Upper MAC, on the other hand, coordinates high level on-demand management of channels and the overall network for incumbent protection and fair

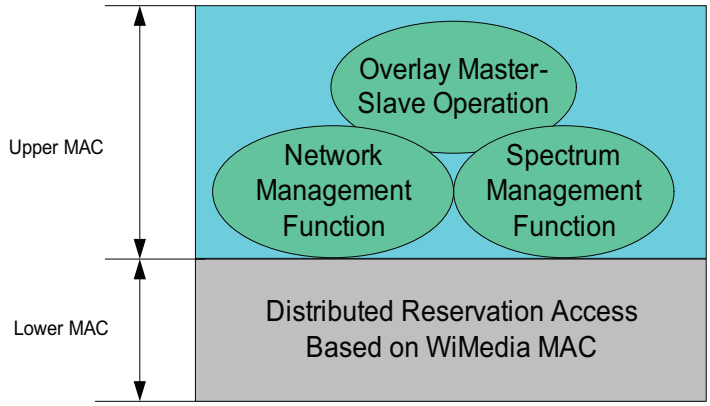


Figure 3.1: *QPDP Overview:* The model of MAC architecture with its key components.

secondary device coexistence.

To make network management consistent and simple, we propose an *Overlay Master-Slave* operation in QPDP. The basic idea is to designate one of the secondary devices as a master—either through user configuration or any other external mechanism. The master device assists network entry, sensing, channel classification and channel switching. A secondary device is designated as master or slave offline, e.g., as an initial configuration parameter.

Note that the Overlay Master-Slave operation over distributed Lower MAC is different from how a purely centralized MAC operates. In a pure centralized MAC, the master acts as the only device performing coordination of beaconing, synchronization, and channel reservation. If the master device fails, the whole network fails, suffering the single-point-of-failure problem. In contrast, the Overlay Master-Slave operation will allow devices to maintain peer-to-peer communications even when the master device is temporarily down, since the Lower MAC is coordinated in a distributed fashion. Such a loosely coupled design allows sufficient time for the master device to recover or to be re-established gracefully.

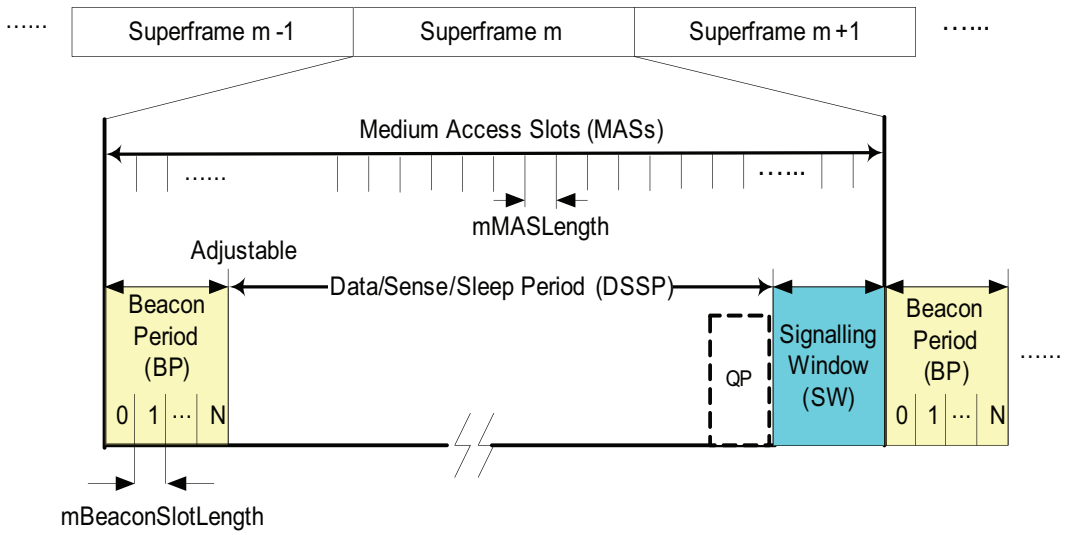


Figure 3.2: *QPDP MAC superframe structure:* The 256 MASs are divided into many contiguous groups which are designated to allow special functions.

3.6 Efficient Control Exchange and Medium Access

We now detail key Lower MAC mechanisms in QPDP w.r.t. QoS support. Although QoS provisioning is illustrated here in the context of QPDP and targeted at the CR system described in Section 3.3, it must be emphasized that the proposed techniques can be considered fundamental to any DSA protocol in order to provide QoS support. Note that many low-level details about DSA itself have been omitted as they are not the primary focus of this work.

3.6.1 Distributed Beaconing based Control and Coordination

For supporting QoS in variable wireless conditions, and particularly with DSA, there must be a low overhead control information propagation mechanism in the network to provide agile adaptation to changing environment. The overall control and coordination must be distributed as conditions (e.g., incumbent presence) may change (or get noticed) in the vicinity of any subset of the devices in the DSAN. Further, it must also feature on-demand control packet transmission (e.g., to inform about arrival

of incumbents), in addition to periodic coordination information exchange. To achieve these goals, QPDP features distributed beaconing and signaling, as described next.

As shown in Fig. 3.2, the MAC structure (adapted from WiMedia MAC) follows a recurring superframe structure, which consists of a *Beacon Period* (BP), *Data/Sense/Sleep Period* (DSSP), and a *Signaling Window* (SW).

Each superframe consists of 256 equal-interval *Medium Access Slots* (MASs) numbered 0-255. A MAS represents a unit of time that can be accessed by either reservation or contention, or utilized for sensing *Quiet Period* (QP). A QP is needed for spectrum sensing. The beginning of a superframe is the BP, and is used to transmit (and receive) special control packets called beacons. During BP, each device in the network transmits a beacon in its beacon slot. The number of beacon slots, i.e., BP length, is adjustable according to the number of devices in the network. Beacons are used for coordination among member devices as well as for negotiating and informing DSA decisions.

Apart from self-identification data, beacons consist of a variable number of *Information Elements* (IEs) corresponding to various aspects of MAC operations. For example, an IE requesting reservation of MASs can be included in a device's beacon. Compared with centralized beaconing, distributed beaconing can effectively avoid the hidden terminal problem, which not only causes collisions but also makes DSA spectrum sensing unreliable. Moreover, distributed beaconing allows the overall system to be more reliable and avoid the single point-of-failure problem.

The remaining MASs (those not included in the BP and SW) in a superframe can be used for data transmission, spectrum sensing activities, or remain un-utilized. This portion of the superframe is collectively called as the DSSP. DSSP MASs can be reserved beforehand by any device for data transmission, or be shared among all nodes through contention-based access. Further, MASs in DSSP can also be reserved as QPs for spectrum sensing.

Additionally, there are few specialized MASs (SW) at the end of superframe for exchanging additional control and management information, such as network entry, sensing report and channel request. Any device may use the SW to send control/management information on demand. In contrast to the BP, the SW is shared by all devices opportunistically, thus improving channel efficiency for signaling. There are energy-efficiency and reliability advantages of using reserved SW, instead of some random available MAS. For example, a device can go to sleep mode during DSSP without missing any control message. Although a device can still use any available MAS in DSSP for exchanging control information, it may require all the intended receivers to remain awake during DSSP, which reduces energy efficiency. Moreover, MASs in DSSP may not be available during "peak data traffic time", which could cause unacceptable delays to critical control messages (such as channel-switch messages) in order to protect incumbents from secondary transmissions.

3.6.2 Distributed Channel Reservation

Reserving the medium for a particular communication stream is a standard technique used in networking to ensure QoS guarantees (e.g., in circuit-switched networks) in both wired and wireless scenarios. In contrast to contention-based access, reservation-based access allows a stream to maintain steady bandwidth resource, as well as, ensure low jitter. Moreover, it can further improve spectrum efficiency associated with DSA, since it avoids the overhead of collisions in contention-based access. Therefore, any DSA protocol should incorporate low overhead channel reservation and release mechanisms for QoS support. Furthermore, the channel reservation/release approach should ensure fairness among contending secondary devices.

In QPDP MAC, channel reservation and release are achieved through the use of beacons. By default, all the MASs in DSSP of a superframe are available for contention-based access by all the devices in the DSA network. However, they can

also be reserved for solitary transmission by any participating device. Each device includes *Reservation Availability IE* in its beacon, indicating the device's view about channel reservation status of the MASs in the upcoming superframe. A special IE called the *Reservation Request IE* is included by a device in order to reserve a specific range of MASs in the superframe. On receiving a beacon containing a reservation request, other devices update their MAS availability map, and their transmission during the reserved MASs is disallowed. Reservation Relinquish IE can be included by a device to request the release of certain reserved MASs by another device. This ensures fairness in the reservation process. In our current design, reservation is secured in a FCFS fashion, when there is a conflict in the process during the same superframe.

3.7 Network and Spectrum Management

In this section, we discuss efficient network and spectrum management features in QPDP, which are based on the Overlay Master-Slave approach discussed earlier. We focus on four key aspects: bootstrapping, spectrum sensing, channel switch, and channel sharing between neighboring (or coexisting) networks.

3.7.1 Network Entry and Association

Automatic device discovery and association is an essential component of providing complete QoS provisioning support for real-world consumer wireless systems. It directly impacts user-perceived QoS, especially in consumer-oriented home/office networks, where users want their devices to be quickly usable and connected. Network entry and association is not straightforward in case of DSA since no default channel is available. [2]

In QPDP, this is accomplished as follows. When the master node powers up,

[2]Control channels may not be available, or unknown to new devices. Further, many DSA protocols do not use control channel due to scalability and regulatory issues.

it automatically performs an initial channel scan from channel 21 to 51 (excluding 37). On each channel, the master senses for at least one superframe duration to look for QPDP beacons. If no QPDP beacon is detected, the master continues to scan for an additional duration to detect incumbents. The master classifies each channel into following categories: idle channel, incumbent occupied channel, secondary device occupied channel, or busy channel (with unknown sources). After the initial scan, the master selects one of the clean (free/only SU-occupied) channels as the operating channel, and other clean channels as backup channels. The master then starts beaconing on the selected channel.

When a slave (i.e., a device not designated by the user as the master device) powers up, it also automatically performs an initial channel scan and classifies channels as described above. In this process, the slave should discover the master device through beacons if the master powered up earlier. In case the master is not found, the slave device can choose to start another round of channel scan until it finds the master. After locating the master device, the slave device starts association with the master device by sending a special join request message in the SW of a superframe. On receiving the request message, the master device can start the authentication process. After authentication, the master device replies indicating whether the new device has been accepted or denied. A new device may be denied either due to authentication failure or traffic congestion. This provides a degree of network admission control. If the new device is accepted, a beacon slot will be allocated to the new device for its beaconing.

As part of the association process, the slave device also reports the channel scan results to the master, especially the channel status of the current operating channel and the backup channels. The master then resolves discrepancies in sensing information, if any; and both the master as well as the associated slave devices can select the best operating channel (in terms of channel quality), as well as, the list of best

available backup channels.

3.7.2 Spectrum Sensing

Spectrum sensing is a unique and necessary function in DSA protocols. It consists of two major components: the MAC sensing schedule (or QP schedule) and PHY sensing algorithm. To maintain steady and high QoS, disruptions due to QPs should be minimized. The selection of a spectrum-sensing algorithm is also critical in ensuring reliable protection for incumbents. [3] Thus, a DSA protocol must design its overall sensing mechanism to balance the QoS-demand while ensuring sufficient sensing performance.

In QPDP, the sensing function is designed as follows. There are two basic types of sensing schemes, fine sensing and fast sensing (or simple energy detection). Fine sensing is mandatory (from FCC) and should be able to detect incumbent signal as low as -116dBm. An example of fine sensing algorithm is the FFT-based algorithm proposed in [72], which is also used in our evaluation of QPDP MAC. Since fine sensing requires large QP duration, it cannot be scheduled frequently, as the overhead and QoS degradation would be significant.

Energy detection (fast sensing) is optional, and can detect incumbents when incumbent signal is higher than -85dBm, at which level secondary devices are subject to severe service interruption. Due to a much shorter detection time, energy detection can be scheduled more often, thus reducing the service interruption when incumbent signal is strong. As suggested in our evaluation results, a combined fine sensing and energy detection scheme performs much better in terms of service recovery.

There are two types of sensing tasks: (1) analyzing utilization and detecting incumbents in the current channel, and, (2) monitoring backup channels. The former, also called in-service monitoring, requires highly reliable sensing, and thus regular QPs

[3] The algorithm depends on the type of incumbent signals to be detected. In UHF band, the typical incumbent signals are TV broadcasts and wireless microphones.

are needed. Further, to ensure effective in-service monitoring, all secondary devices are required to participate in collaborative spectrum sensing. Backup channel scans, on the other hand, can be scheduled less frequently and are accomplished individually or with fewer secondary devices. In QPDP, the master can also schedule backup channel scan for slave secondary devices (according to their traffic schedule), so that service interruption may be minimized.

As mentioned earlier, QP requires all traffic to be suspended, causing interruptions to QoS-sensitive applications. Therefore, in a multimedia CR system, long QPs should be avoided. For this, we propose scheduling of multiple short QPs throughout CDT. Assume that QPs are scheduled at a periodic interval (say QPI) with each sensing lasting for certain minimum duration (say QPD). Note that QPD value depends on the underlying sensing algorithm. Let T be the sensing time (which also depends on the sensing algorithm) needed to detect incumbent at IDT with required PD and PFA. Then, as long as $(QPD/QPI).CDT \geq T$, the regulatory requirement is met. QPDP schedules QPs once every M superframes by putting the QP right before the SW. To ensure that each device follows the same QP schedule, the master and other nodes advertise the QP schedule within their beacons. Further, nodes joining the network can obtain the QP schedule from the master as part of the association procedure.

Whenever a new sensing sample is obtained, the device processes the last M ($M = T/QPD$) sensing samples. Then, a sensing report is sent to the master during the SW period or through beacons. It is possible that there is a discrepancy on sensing results between secondary devices. Therefore, the master device can further consolidate the sensing results by performing data fusion to increase the probability of detection and reduce the rate of false alarms.

In addition to regular QPs, on-demand QPs can always be scheduled whenever some abnormal behavior is observed. For example, if strong interference or sudden

channel quality drop is observed, a device can request to schedule QP for the detection of incumbents or other interfering sources.

3.7.3 Channel Switching

Channel switches constitute another important aspect of DSA. If incumbent signal (above the IDT threshold) is detected on the operating channel, the secondary network must switch to another channel (within the CMT), in order to avoid interfering with the incumbent. Moreover, a network may also decide to move to another channel if a hidden neighbor network is found (network merge, to be discussed later, is another option).

The interruption overhead (and hence QoS degradation) associated with a channel-switch can be significantly reduced if the next channel to be used is already known to the communicating SU devices. Usage of backup channels (BCs) prevents reactive sensing or probing in order to search for a new channel. Further, since BCs are negotiated between devices prior to an actual channel switch, it minimizes the coordination control overhead involved at the beginning of utilizing a new channel. Thus, BCs can play an important role in maintaining the desired QoS level during DSA operations, and should be incorporated by DSA protocols to offset significant QoS degradation due to channel-switches.

In QPDP, the master is responsible for designating backup channels, ordered by certain metric (e.g., channel utilization). The master uses its own spectrum sensing results and spectrum sensing reports collected from other nodes to build the ordered list of BCs. The BC list is selected by the master during the initial channel scan and updated afterward. Regular check of a target BC is needed in order to make sure the channel is clean and available when needed in future. To minimize the overhead for backup channel scans, the master can coordinate such effort by appointing a particular device (including itself) to sense a particular channel within a certain time

window. A designated node would opportunistically use its idle period for backup channel scan so as not to disrupt the ongoing communications.

Once an incumbent is detected, the master broadcasts a channel-switch command in its beacons in the next N superframes, indicating the time to move to the first pre-negotiated backup channel. It is also possible, however, that the incumbent signal power level is so high that no beacons can be received. Therefore, a timeout mechanism is implemented at the nodes to deal with such a situation. For instance, once a slave device cannot receive the master's beacon for N consecutive superframes, it would automatically start a channel-switching procedure, in order to discover the master, starting with the first BC.

After moving to the new channel and re-synchronization with the master device, all devices continue to keep the same channel reservation schedule for beaconing and data transmission, as used in the previous channel. We call this *Channel Imaging*. The benefit of Channel Imaging is to resume transmission as fast as possible by avoiding the time-consuming channel reservation re-negotiation. As a result, transmission suspension (or QoS degradation) due to channel switching process can be reduced.

3.7.4 Channel Sharing

Coexistence of secondary networks (also called self-coexistence) should ensure minimal conflicts so as to provide consistent QoS to ongoing communication streams. Two secondary networks may be closely located geographically, or come into range due to mobility of the constituent nodes. A DSA protocol must be able to identify such a situation and invoke mechanisms to minimize the resultant interference.

In QPDP, a device discovers an alien network by detecting alien beacons which may be received in periods other than BP. Alien beacons contain a different network ID. A slave device then reports the information contained in the alien beacon to its master. It is the responsibility of the master to decide whether to share channel with

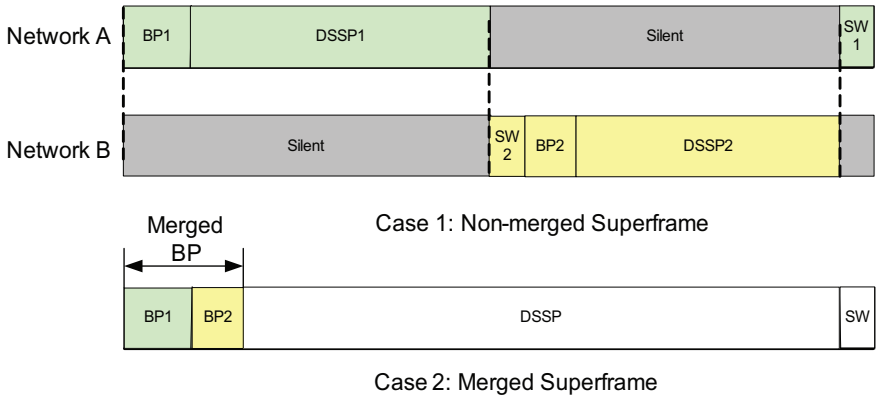


Figure 3.3: *Coexisting of DSANs:* Possible ways of channel sharing between neighboring secondary networks.

the alien network or to switch to another channel. The decision is based on available bandwidth on sharing versus switching to a new channel.

If the two networks negotiate and decide to merge, the Lower MAC then figures out how to share the channel. One approach is to share a channel as shown in case 1 of Fig. 3.3, i.e., each network alternates the use of channel for certain duration (static contiguous time block). Although this solution is straightforward, QoS provisioning will be a major issue, especially for delay-sensitive applications. For example, the packet delay variation will increase significantly since no transmission is allowed during the silent periods. Moreover, pseudo-static access-time allocation is inefficient in terms of channel sharing between the neighboring networks.

QPDP addresses these issues by allowing two networks to merge into the same superframe, as shown in case 2 of Fig. 3.3. Superframe merge allows two neighboring networks to share channel MASs, thus improving channel efficiency and reducing delay variation. QPDP also merges QPs of the two networks, thus reducing sensing overhead.

Note that two neighboring networks can still choose to function independently. One network can decide to move to another channel or re-start without disrupting the operation of another network. Moreover, network association and device authen-

tication are limited to individual networks only, thus maintaining security features. This network independence is possible due to flexible coupling between Lower MAC and Upper MAC functions, as described in Section 3.5.

3.8 Performance Evaluation

In this section, we analyze the performance of QPDP using simulations. We are particularly interested in two aspects of QPDP w.r.t. QoS provisioning: (1) MAC efficiency in supporting high rate, low delay, and small error rate in a typical indoor environment; (2) MAC robustness in response to incumbent disruptions.

We simulate QPDP in a home network setting using the OPNET Modeler [73]. Note that our platform CR system must be able to support HDTV streaming, which requires extremely high-level of QoS, making it close to the worst-case usage scenario for any consumer wireless network QoS provisioning model. Full 1080p HDTV streams requires high data-rate (close to 20Mbps) [4], small end-to-end delay (less than 100ms), small jitter (less than 50ms), and very low bit error rate (less than 5%). The evaluation will show how QPDP (with its QoS provisioning features) matches up to such stringent QoS requirements, while still providing DSA capability.

The OPNET implementation for simulations involved creation of HDTV generator (source) and receiver (sink) nodes. The key elements of the implementation include the OPNET process models for QPDP. We modeled HDTV transmission characteristics (e.g., packet size, data-rate.) in the generator process models, together with the probabilistic models simulating actual HDTV stream packet generation. We also created the incumbent node process model which mirrors typical incumbent behavior expected in UHF bands.

[4]HDTV streaming bandwidth requirement may vary depending on its type and the compression codec used. Here, we consider the standard H.264 codec with 1080p streams.

Table 3.1: PHY parameters

Parameter	value
Transmission power (dBm)	20
Noise power spectrum density (dBm)	-174
Noise figure (dB)	6
Implementation loss (dB)	6
Communication distance (m)	30
Path loss exponent	3
Multipath fading model	Exp. Rayleigh

Table 3.2: PHY-OFDM parameters

Parameter	value
Signal bandwidth (MHz)	5.40
FFT size	128
Inter-carrier spacing (KHz)	50
No. of data subcarriers	4
Modulation for data payload	64-QAM, 5/6
Modulation for PLCP header, beacon, control frame	16-QAM, 1/2
RS outer coding t	5
Preamble	4 sym
PLCP (PHY+MAC) header	1 sym
Guard interval	$T_{FFT}/16$
Symbol duration (μs)	21.25

3.8.1 Evaluation Setup

The setup consists of a single-house home network (range of ≈40m). The channel is modeled with exponential Rayleigh multipath fading. The path loss factor is set to 3. We assume the transmission distance between the HDTV transmitter and the HDTV receiver is 30m.

As introduced in Section 3.3, the CR system is operating in UHF band (channel 21-51, excluding channel 37). The PHY of the CR system is based on OFDM with a total 128 FFT size. The subcarrier space is 50kHz. The guard interval is set to 1/16 and therefore, the OFDM symbol duration is 21.25us. In other words, it allows the system to mitigate inter symbol interference (ISI) when delay spread is less than

Table 3.3: MAC parameters

Parameter	value
Superframe length (μs)	110,592
No. of MAS	256
MAS length (μs)	432
Max. BP length (MAS)	5
Min. sensing time per CDT (ms)	30
Beacon slot length (μs)	432
Channel switch command repetition count	3

Table 3.4: Sensing schemes used

Abbr.	Description
FS-1	Fine sensing scheme 1. 5ms QP every 3 superframes
FS-2	Fine sensing scheme 2. 10ms QP every 6 superframes
FS-3	Fine sensing scheme 3. 15ms QP every 9 superframes
FS-4	Fine sensing scheme 4. 20ms QP every 12 superframes
FS-5	Fine sensing scheme 5. 25ms QP every 15 superframes
FS-6	Fine sensing scheme 6. 30ms QP every 18 superframes
ED	Energy detection, 1 MAS QP scheduled every superframe.

100ns, typical in home environments.

In the simulation setup, the sender and receiver nodes power up within 1 second of the start of the simulation, unless mentioned otherwise. HDTV sender and receiver are pre-designated as the master and the slave node respectively. They automatically associate with each other to form the DSA network, through the QPDP mechanisms, discussed earlier in Section 3.7. HDTV streaming is started once the network is formed. The rest of simulation parameters can be found in Table 3.1, 3.2, and 3.3.

Also, the list of sensing schemes (used to study the impact of sensing schedule) is presented in Table 3.4. Note that in each of the fine sensing schemes, the long-term average sensing overhead remains the same. Also, energy detection works only when incumbent signal strength is greater than -85dBm.

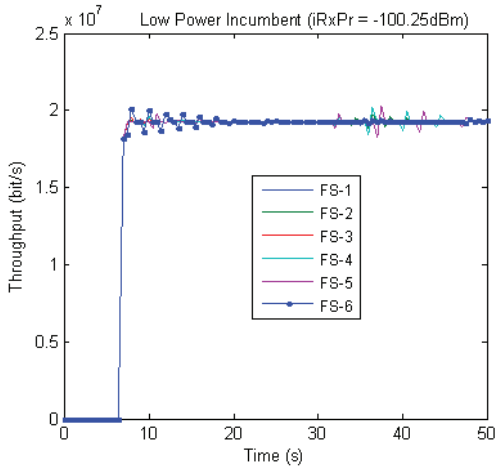


Figure 3.4: Throughput in presence of low power incumbent.

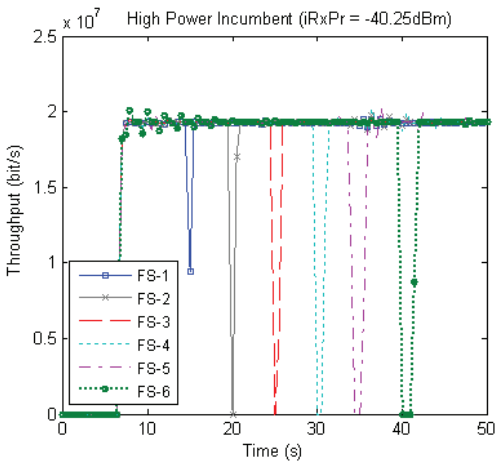


Figure 3.5: Throughput with high power incumbent.

3.8.2 Simulation Results

Fig. 3.4 shows the throughput performance in presence of a low power incumbent. We observe that the proposed sensing mechanism does detect the incumbent by aggregating multiple sensing samples, and the devices switch to a BC. The incumbent signal power received ($iRxPr$) is found to be at -100.25dbm. It can be noticed that the incumbent transmission power is low enough to allow secondary HDTV communication to continue without introducing any perceivable degradation to QoS level, as seen from throughput values. The observation is consistent through various fine sensing schemes, as seen from the graph.

Fig. 3.5 shows the impact of a high power incumbent on throughput performance. Since the incumbent now transmits at high power, it immediately affects the reception of the HDTV stream resulting in a higher percentage of packets received in error. Thus, the required QoS level for throughput cannot be maintained and the throughput drops as seen in the graph.

The graph in Fig. 3.5 also shows the relative degree of impact on application throughput depending on the delay in incumbent detection of various sensing schedules. If the QP is frequent (5ms every 3 superframes), the impact on application QoS

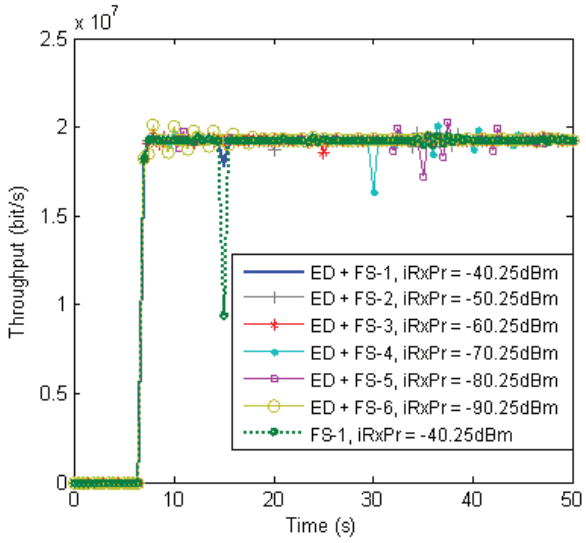


Figure 3.6: Combined fast sensing (energy detection) and fine sensing minimizes sensing interruptions but maintains its effectiveness.

is much less than the case when sensing is less frequent (e.g., 30ms every 18 superframes). For sensing scheme FS-6, service recovery could take up to 18 superframes, i.e., 2s. For sensing scheme FS-1, service recovery time (detection time plus channel switching time plus session resume time) is bounded by 3 superframes, about 0.33s. Note that in both sensing schemes, the average sensing rate (hence average sensing overhead) is the same. Thus, differences in the sensing schedule can influence how quickly QoS-degradation can be detected without increasing the overall overhead. Also, we observe that *Channel Imaging* contributes significantly in minimizing the disruption due to channel-switches.

Fig. 3.6 shows the throughput performance when both energy detection (fast sensing) and fine sensing are incorporated. As received incumbent signal increases from -100.25dBm to -40.25dBm, the throughput does not drop significantly as compared to the case when only fine sensing is applied. Note that the energy detection is scheduled in every superframe (for 1 MAS). Thus, incumbent detection time can be limited to one superframe when incumbent signal is strong enough to cause immediate service disruption to the secondary users.

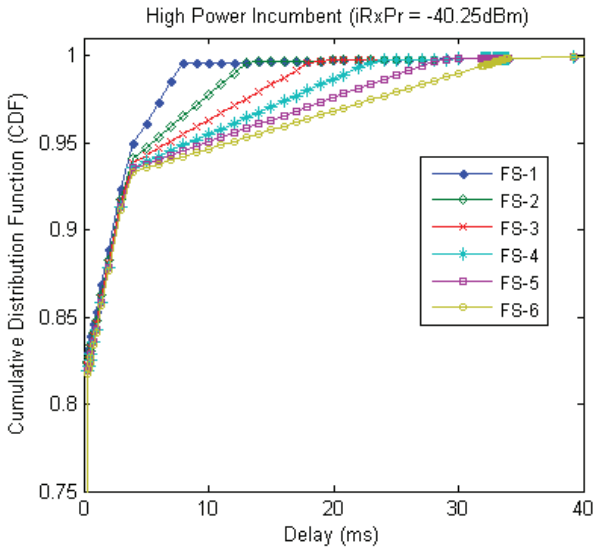


Figure 3.7: QP schedule directly impacts delay performance.

Fig. 3.7 shows the delay performance when different sensing schedules are deployed. As expected, the sensing schemes significantly affect the delay experienced. As shown, more than 5% packets experience significantly higher delay when the sensing QP duration is increased from 5ms (every 3 superframes) to 30ms (every 18 superframes). With low power incumbent, similar delay performance is observed. The primary reason for this is the suspension of transmission during QPs.

In all of the above scenarios, quick incumbent detection and fast channel switches (Fig. 3.8) play a significant role in sustaining the required application QoS levels (e.g., achieving nearly 20Mbps bandwidth) with minimal disruptions. The top part of the figure shows the throughput variation with time. When an incumbent starts transmitting on the same channel (shown in middle part of the figure), a quick channel-switch is initiated for recovery. Use of BCs and Channel Imaging contribute towards fast recovery from the disruption.

The narrow spectrum-width (approx. 6 MHz) of TV channels necessitates designing extremely efficient DSA protocols in order to sustain stringent multimedia QoS. The results highlight the efficiency of QPDP in delivering a high quality HDTV

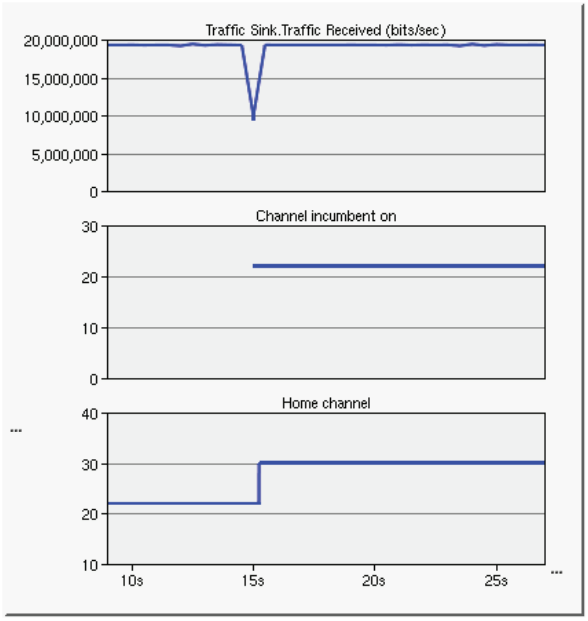


Figure 3.8: Fast incumbent detection and avoidance (by channel-switches) by QPDP minimizes traffic loss and sustains QoS.

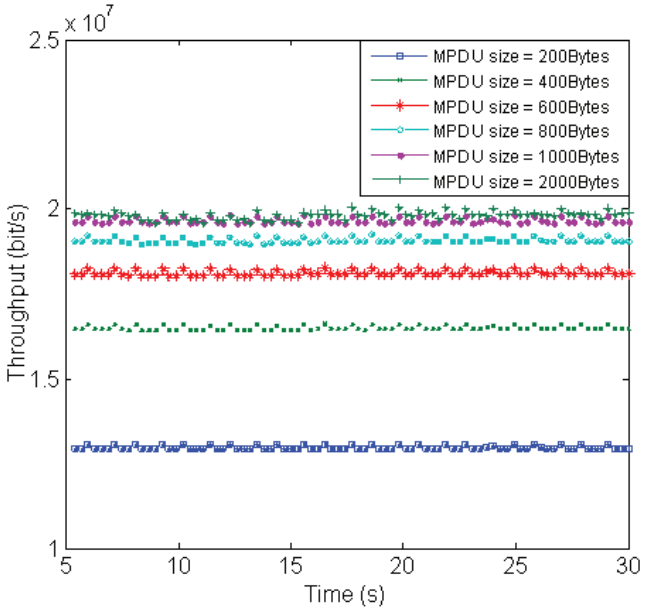


Figure 3.9: Frame aggregation is found to be crucial in achieving high protocol efficiency.

stream consisting of small packets. The techniques of channel reservation and frame aggregation (see Fig. 3.9) play a key role in ensuring protocol efficiency. The spectrum efficiency, calculated by dividing goodput with signal bandwidth, reaches as high as

3.7 bit/s/Hz.

3.9 Concluding Remarks

In this chapter, we have provided a case study of DSA QoS provisioning in the context of designing QoS-Provisioned DSA protocol (QPDP) for enabling DSA-based multimedia streaming in indoor home/office wireless networks. QPDP incorporates both fine-grained and coarse-grained QoS provisioning mechanisms. Fine-grained (packet-level) QoS support is provided primarily via slot-based channel reservation. Coarse-grained QoS support ensures QoS at stream (or connection) level and is provided through intelligent network and spectrum management. QPDP was evaluated by simulating HDTV streaming in a single home network setting through the OPNET Modeler. The simulation results show the effectiveness of QPDP in supporting high level QoS requirements throughout a communication session in the targeted CR system, and reveal the impact of key QoS provisioning designs like sensing schedule in minimizing traffic disruption while ensuring necessary incumbent protection. It is shown that a high level of protocol efficiency, as achieved by QPDP through various intelligent optimizations, is critical to supporting QoS in narrow TV bands in DSA networks.

CHAPTER IV

Application QoS Support in DSA

4.1 Introduction

In Chapter III, we presented basic QoS provisioning mechanisms for DSA protocols. However, the proposed approaches were primarily remedial in nature with just the MAC level perspective—not incorporating application QoS awareness as an integral part of DSA behavior. Further, they were optimized for the specific study platform, namely the home WLAN featuring multimedia streaming. In this chapter, we consider assimilation of application layer QoS in DSA.

Gap Between State-of-art DSA[1] and Application QoS: The main motivation behind this chapter is our investigation which reveals that though conventional DSA leads to gain in spectral resource at the channel level (i.e., the link layer bandwidth) it does not translate into corresponding gain in application performance. DSA's negative side-effects, which are undesirable for network applications, constitute the main cause behind this phenomena. Fundamental functions executed in DSA, like spectrum sensing or channel switches, could cause delays and disruptions to the applications, thereby introducing QoS degradation. Containing the interference to incumbents is the key requirement for opportunistic usage of the licensed

[1]In this chapter, the term "state-of-art DSA" denotes contemporary or existing DSA, and they are used interchangeably throughout the text.

spectrum. Thus, any PU activity adds to the interruptions suffered by SU applications. Further, DSA may also result in link capacity fluctuations due to a reduction in frequency-width or less-efficient MAC-PHY schemes on a new channel. In the worst case, session handovers, terminations and re-establishments may occur, exacerbating application QoS degradation.

Therefore, contrary to the current perception in the DSA research community, we argue that achieving gains in link layer capacity is not good enough at the application level. We make two mutually contradictory observations about contemporary DSA that define the main theme of this chapter.

O1. DSA aims to improve network application performance.

O2. Traditional DSA is agnostic to application needs.

Clearly, O1 is orthogonal to O2. Hence, DSA incurs unwarranted side-effects, directly impacting application QoS. We postulate that there exists a tension between DSA operations and a network application's traffic requirements, which necessitates cooperation between the two entities to effectively achieve DSA's goals.

While there has been some recent work in accounting for QoS in DSA [11, 43, 44], only low level QoS metrics have been considered, e.g., link SNR and BER. Thus, they are unable to accurately capture specific application layer QoS demands which have high-level semantics.

Proposed Approach: In this chapter, we propose an adaptive and application-aware service framework for DSA in order to improve application performance. We call this DSA-enhancement *Context-Aware Spectrum Agility* (CASA), where *context* comprises application hints as well as current spectrum conditions. The high level application context is first processed through *semantic matching* to determine their dependency on low level DSA parameters. CASA exploits the short-term correlation of the recent networking state with near future in making DSA application-aware. The

adaptivity scheme of CASA is built-upon reward-based Reinforcement Learning [74] methods. The main goal of CASA is to minimize the undesirable impacts of DSA, and thus, re-enforce and enhance the merits of DSA as a performance-improving feature in wireless devices. Our evaluation shows that CASA deployment improves application performance significantly during DSA operation, and indirectly, makes DSA resilient under stringent application QoS requirements.

Our solution is inspired by the advantages shown in the areas of power management and wireless network selection [46, 47], when application-awareness is incorporated in corresponding optimization algorithms. These works highlight that application behavior/demands are the most significant aspect of networking context, and their effective integration with networking protocols can be instrumental towards improving application performance.

Organization: The chapter is organized as follows. Section 4.2 details the motivation for our work. Sections 4.3–4.7 describe the CASA framework, while Section 4.8 describes its implementation. Section 4.9 presents CASA evaluation and results. Section 4.10 concludes this chapter.

4.2 Motivation

As seen in Section 1.1, there are various functionally-disjoint components in DSA. Therefore, their overhead and inter-play may have unwanted side-effects on end-user applications in typical networking conditions. This observation motivated us to investigate the behavior of DSA in direct relation to application QoS requirements. Based on our analysis, we confirmed the existence of the following problem with contemporary DSA: *Despite resulting in gain of spectral resources, DSA produces unwanted impact on application QoS performance due to disruptive side-effects produced by its basic functions.* We next elaborate on the main causes for the observed problem, and also present experimental proof confirming the same.

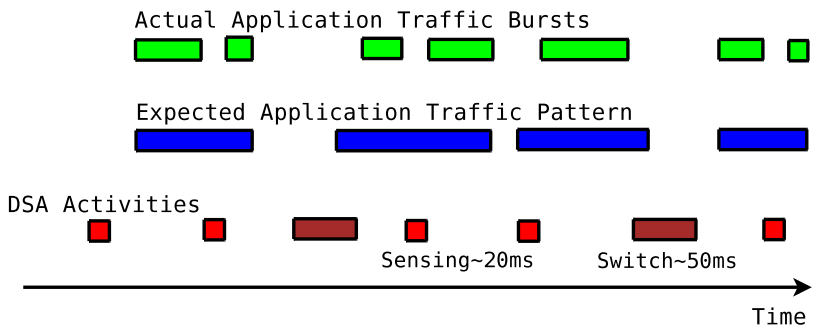


Figure 4.1: Disruptions caused by DSA operations result in a fractured flow of application traffic.

Impact of spectrum sensing: Effective spectrum sensing requires scheduling of *quiet periods* (QPs), during which no data can be transmitted/received. Depending on the channel characteristics and the sensing scheme used, each QP may vary from tens to hundreds of milliseconds. For example, a QP in IEEE 802.22 [21] may last for 25ms. Further, a QP may be scheduled very often, as frequent sensing improves DSA performance. Frequent spectrum sensing is also required to comply with regulatory guidelines on quick incumbent detection. For instance, FCC requires incumbent signal to be detected within 2 seconds in TV bands [6].

Thus, the overhead of sensing (due to QPs) can adversely impact application QoS by introducing extra delay & jitter, and reducing usable bandwidth in the process (see Fig. 4.1). The negative side-effects of sensing are magnified when the channel is narrowband and offers lower capacity (e.g., TV bands are 6MHz wide). Sometimes, due to size/cost/hardware constraints, a separate spectrum-sensor may not be available. In such cases, "out-of-band" sensing will add to "in-band" sensing impact by significantly multiplying the amount of QPs scheduled on the only available wireless interface. External sensing approaches (e.g., sensing database) may mitigate the sensing impact at the cost of additional infrastructure. However, this does not fully eliminate sensing overhead as in-device sensing would still be required for validation purpose and to detect any unexpected incumbent activity, or when external sensing

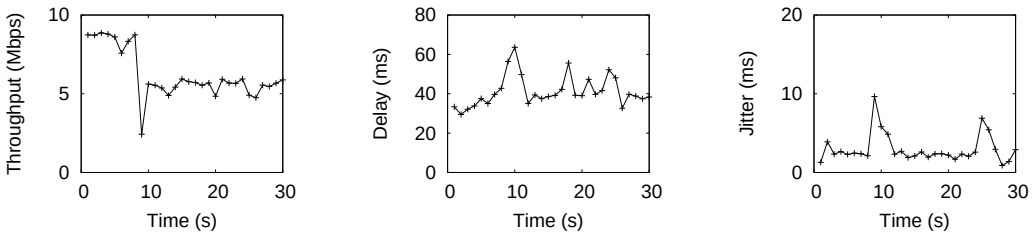
data is unavailable.

Impact of switching channels: Channel-switching is fundamental to DSA. Like spectrum-sensing, it has a similar disruptive side-effect. A channel-switch interrupts application traffic for durations lasting several milliseconds. This includes the time to reset the wireless interface(s), and more importantly, loading MAC-PHY protocols to access the new channel and complete management tasks like association or authentication with the secondary base station.

Channel-switching may not happen as frequently as spectrum-sensing, and hence, its delay/jitter impact is lower. However, it can produce the adverse side-effect of reducing the available bandwidth for application traffic. For example, the new channel may have a lower frequency-width (narrowband channel) than the previous channel, which could result in a sudden decrease in link capacity. Capacity reduction can also be contributed by the usage of a comparatively less-efficient MAC-PHY schemes in the new channel, even though the channel is better in terms of utilization and radio characteristics.

Impact of incumbent protection: Very limited interference to the PUs is of critical importance to DSA (ideally there should be no interference to incumbents). Hence, a SU must stop its transmission or switch channels, whenever it detects PU activity. If incumbent activity is prolonged, then SU application traffic is effectively stopped, resulting in severe application QoS degradation. Although the underlying DSA protocol is designed to take corrective actions when such a situation persists (e.g., by switching to a different channel), an incumbent activity still results in significant disruption to ongoing communication.

Even short-term incumbent activity can adversely affect QoS-sensitive applications if they occur frequently. Thus, average incumbent utilization metric may not be sufficient to determine the quality of a channel with respect to application requirements—information about incumbent access pattern must be taken into ac-



(a) Throughput (b) Delay (c) Jitter

Figure 4.2: Variation of key QoS metrics with time using traditional DSA.

count.

Other unwanted QoS impacts of DSA: Additional delays can also be introduced due to coordination of devices in a multi-SU or adhoc-type networks. Though not the primary focus of this work, many contemporary DSA protocols require periodic listening to a *control* channel for coordination [4], which can be a significant disruption to application traffic in single-interface systems.

Compounding impact of side-effects: The decline of service quality because of the negative side-effects of DSA usually turns out to be more significant and lasts for a much longer duration, because other layers in the protocol stack may perceive incorrect network conditions. For example, TCP may view the sensing-introduced delays as congestion in the network. Similarly, QoS-centric protocols like RTP [75] will experience frequent short-term adjustment periods. Further, application sessions may be completely disrupted, leading to additional overheads of their re-establishment.

Experimental demonstration of the problem: We conducted simple testbed experiments to study the impact of DSA side-effects on application traffic. We use our implementation of PU and SU emulators (details in Appendix B) for this purpose.

The experiment setup consists of three channels with average incumbent utilization of 0.7, 0.5 and 0.3, respectively. Channels 1 and 2 can support the highest 802.11a raw capacity (54Mbps), while channel 3 is constrained to 30% of the maximum possible value (e.g., TV channels may be freer than ISM channels, but have low capacity due to narrow spectrum-width). The only secondary user in this setup communicates

with a fixed host on the Internet (through a DSA-capable access point) and bootstraps on channel 1. We keep the application bandwidth demand at 10Mbps, which is enough to support multiple high-quality QoS streams (e.g., each G.711 audio + H.261 video typically requires 460kbps [76]) along with other types of network traffic.

Fig. 4.2 shows the typical behavior of traditional DSA in a sample experiment run. The impact of DSA on application traffic in terms of three QoS parameters—throughput, end-to-end (roundtrip) delay, and jitter—are plotted. The graphs show their average values over each second through a 30-second period of application session. As seen, the throughput target is not met: in fact, it drops significantly at around 9s. As channel utilization is the contemporary DSA's key decision parameter, the SU switches to channel 3 as it has the lowest utilization. However, it fails to take into account its lower capacity relative to the application bandwidth requirements (which is 10Mbps). This scenario may occur quite often when SUs switch to channels in different spectrum regions. For example, from a highly utilized 2.4GHz channel to less utilized (but lower capacity) 4G uplink cellular channel opened by its operator for unlicensed operation.

Though the delay and jitter are found to be acceptable on average, they are found to fluctuate significantly on a short time-scale. In particular, jitter is as high as 10ms in many cases. Since DSA is supposed to be a performance-enhancing technology, any degradation of services because of introducing DSA would not be attractive, especially for those applications that require certain minimum bandwidth, delay, and jitter guarantees.

4.3 Context-Aware Spectrum Agility (CASA) Overview

We propose a novel wireless networking service architecture, called *Context-Aware Spectrum Agility* (CASA), which augments traditional DSA by making it application-aware in order to address the issues identified in Section 4.2. CASA introduces

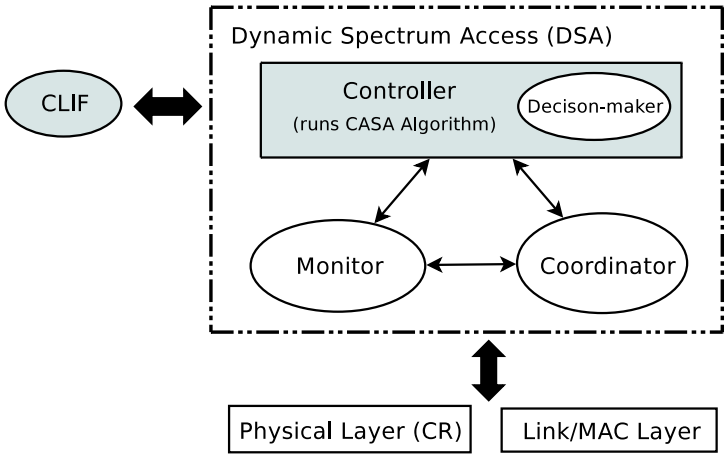


Figure 4.3: CASA's main component is the CASA Algorithm, that augments the "Decision-Maker" to form "Controller".

application-awareness to DSA through the *CASA Algorithm* which takes context information (both application and lower layer contexts) as input, and updates key DSA parameters as necessary. CASA Algorithm executes periodically as part of the *Controller* as shown in Fig. 4.3. This period is referred to as *CASA Epoch* (τ_{casa}). CASA also provides a low overhead interface called *Cross-Layer Interaction Framework* (CLIF) to enable applications to export their QoS context information. Details on various elements involved in CASA are provided in coming sections.

4.4 Context Information

4.4.1 Application Context

The application context comprises conventional application QoS attributes. Specifically, they are thresholds for three important QoS metrics for a network application—*minimum bandwidth*, *maximum delay*, and *maximum jitter*. These are considered on an end-to-end and per-session basis. This set of QoS parameters provides a simple but effective abstraction of application layer QoS demands. Bandwidth, latency, and jitter are chosen because they are directly impacted by DSA side-effects (as seen in Section 4.2).

Since most modern QoS-sensitive network applications already monitor and manage QoS metrics, application QoS hints can be provided without significant modifications or additional overhead. For example, multimedia streaming applications typically use RTP sub-layer [75], which monitors traffic characteristics including QoS information through its RTCP component. Thus, RTP/RTCP can be leveraged to provide the necessary application context.

We formalize the application context as follows. Assume that there are n network applications running on a SU device. For every ongoing communication session, application i exports the required bandwidth (b^i_{req}), end-to-end delay (d^i_{req}), and jitter (j^i_{req}). Here we show the analysis for one communication session per-application in interest of presentation clarity. However, CASA supports any number of sessions for each application.

Similar to the "requirement parameters", the applications also provide the current "observation parameters"—bandwidth (b^i_{obs}), delay (d^i_{obs}), and jitter (j^i_{obs}), for the corresponding communication sessions.

Individual applications contexts are combined to generate the following cumulative application context.

1. $B^{app}_{req} = \sum_{i=1}^{n} b^i_{req}$,
2. $D^{app}_{req} = \min\{d^i_{req}\}, 1 \leq i \leq n$,
3. $J^{app}_{req} = \min\{j^i_{req}\}, 1 \leq i \leq n$.
4. $B^{app}_{obs} = \sum_{i=1}^{n} b^i_{obs}$,
5. $D^{app}_{obs} = \min\{d^i_{obs}\}, 1 \leq i \leq n$,
6. $J^{app}_{obs} = \min\{j^i_{obs}\}, 1 \leq i \leq n$.

Thus, the complete application layer context is the pairs: (B^{app}_{req}, B^{app}_{obs}), (D^{app}_{req}, D^{app}_{obs}), and (J^{app}_{req}, J^{app}_{obs}).

Aggregation of all application QoS hints in thus-defined manner simplifies CASA design. At the same time, this method guarantees that fulfillment of any aggregated requirement parameter corresponds to the fulfillment of the corresponding individual application QoS requirement, with a very high likelihood. Analogous reasoning can be easily applied for "observation" parameters.

In the CASA architecture, applications can update and export their QoS requirements/observation pairs whenever they change. Providing all three of the parameter pairs is optional—an application may also provide a subset of these parameter pairs (or none of them) depending upon its operational needs.

It must be pointed out that there is a subtle but important difference between the QoS abstractions corresponding to delay/jitter (D^{app}_{req}, D^{app}_{obs}, J^{app}_{req}, J^{app}_{obs}), and those corresponding to bandwidth (B^{app}_{req}, B^{app}_{obs}). Delay or jitter is *transit-additive* in nature—individual network segments traversed along the communication path contribute to produce the cumulative end-to-end delay/jitter as seen by applications. Thus, each link or network segment along the communication path must minimize its individual contribution to these metrics for better application QoS. On the other hand, bandwidth is *transit-reductive* in nature—the overall bandwidth experienced by the application is the least bandwidth experienced along the entire communication path. Therefore, bandwidth provided by each network segment must be sufficiently high for acceptable end-to-end bandwidth. This distinction is key to how we derive estimates from raw lower layer context to semantically match application QoS abstractions. Semantic context matching is elaborated in Section 4.5.

4.4.2 Lower-layer context

Basic physical layer data are known natively through the CR/DSA MAC-PHY parameters. They include the the set of channels C (or the *spectrum*)[2] available

[2] "Channel" implies "licensed channel", unless otherwise mentioned.

Table 4.1: List of Symbols

Symbol	Description
λ_{pkt}	Rate of app. packets generation
B^{app}_{req}, D^{app}_{req}, J^{app}_{req}	App. layer bandwidth, e2e delay & jitter reqs.
C	Set of channels in the spectrum
c	A channel, $c \in C$
$u(c)$	Total utilization of c
$u^{pu}(c)$, $u^{su}(c)$	Utilization by PUs and SUs respectively in c
$T^{pu}_{on}(c)$, $T^{pu}_{off}(c)$	Random var. for PU's ON/OFF durations in c
$T^{su}_{on}(c)$, $T^{su}_{off}(c)$	Random var. for SU's ON/OFF durations in c
$t_{sense}(c)$, $r_{sense}(c)$	Duration and rate of sensing in c
$t_{switch}(c1, c2)$	Overall time to switch from channel $c1$ to $c2$
$B(c)$	App. bandwidth estimate for channel c
$D(c)$,$J(c)$	Additional delay and jitter when using c
$e(c)$	Efficiency of MAC-PHY protocols used on c
τ_{casa}	CASA Epoch duration
P_{curr},P_{past}	Current and past push factors
P_{MAX}	Normalization constant for push factor
N	History window (no. of past epochs)

for DSA, and the corresponding access protocols to use in each spectrum region. Consider any channel $c \in C$. The average channel utilization by PUs ($u^{pu}(c)$), and by SUs ($u^{su}(c)$), is known through spectrum sensing (as discussed in Section 1.1). The total utilization seen on channel c is therefore:

$$u(c) = u^{pu}(c) + u^{su}(c) \tag{4.1}$$

Average ON/OFF durations of PUs ($E[T^{pu}_{on}(c)]$, $E[T^{pu}_{off}(c)]$) and SUs ($E[T^{su}_{on}(c)]$, $E[T^{su}_{off}(c)]$) on channel c are also known, as they are used to calculate $u^{pu}(c)$ and $u^{su}(c)$ [39].

The expected spectrum sensing duration ($t_{sense}(c)$) and the rate of sensing schedule ($r_{sense}(c)$) are obtained from the spectrum sensing scheme followed by the DSA protocol. The values of r_{sense} and t_{sense} are chosen from a range determined by the PHY sensing schemes and regulatory policies. Also, the set of their allowed values can be different for different channels in the spectrum.

In contrast to the above mentioned MAC-PHY parameters, some of the required lower layer information may not be directly available. In such cases, low overhead estimation techniques are used, as discussed next.

The physical layer data-rate ($b(c)$) and the application layer efficiency ($e(c)$) for each of the MAC-PHY scheme on channel c is assumed to be known. If the MAC-PHY protocol has a dynamic rate adaptation feature, then recent historical information on data-rate employed will be used to compute the weighted average estimate for $b(c)$. Weights are based on the duration for which the data-rate was used. If no history is available yet, a median of available data-rates is selected as the initial value.

λ_{pkt} is the cumulative rate of packets arriving at link layer transmission queue from all the applications running on the device. λ_{pkt} can be safely assumed to be constant across the short epoch (τ_{casa}) duration. This assumption is especially valid for many QoS-sensitive traffic like Internet multimedia streaming and VoIP [76].

$t_{switch}(c1, c2)$ denotes the average time for the wireless interface to enter the ready state for data transfer in a new channel $c2$ after switching from the current channel $c1$. It includes the time to deactivate/activate new MAC-PHY mechanisms (if needed), as well as, associating with the new secondary service gateway base station. If $t_{switch}(c1, c2)$ values are not available statically beforehand (for any $c1$,$c2$ pair), CASA builds this information progressively from recent historical observations by calculating their average.

4.5 Semantic Alignment of Contexts

In order to adapt DSA to application needs there must be a semantic congruence between application QoS abstractions and lower layer DSA attributes that impact them. Albeit limited to application and MAC-PHY layers in this chapter, we believe that semantic context alignment is a significant step towards realizing fully cognitive networks [77, 78]. Note, however, that the concept of semantic matching introduced

here is at a high level and generic. For example, it is different from predicting the expected share of a channel's airtime, as in [17].

To achieve the semantic matching of application and lower layer contexts, we process the raw MAC-PHY information to provide a reasonable estimate of attainable application layer QoS. We trade some accuracy for low overhead and implementable system design. Thus, we avoid restrictive assumptions and impractical system modeling. Instead, we employ average values and approximate estimates in translating low level context to compare with application level QoS abstractions.

$B(c)$ denotes the average application layer bandwidth expected on channel c, which is estimated as follows.

$$B(c) = b(c).e(c).\{1 - u(c)\}. \tag{4.2}$$

Delay and jitter are *transit-additive* parameters, as mentioned in Section 4.4. Therefore, we calculate the "additional" delay and jitter contributed by the DSA network, rather than their overall end-to-end values. In our system model, only the one-hop wireless link behavior changes due to DSA, while other aspects of the end-to-end traffic flow remain the same.

Since PUs repeat an ON/OFF cycle of channel access, it is sufficient to consider a unit cycle of average duration $T_{cycle}(c) = E[T_{on}^{pu}(c)] + E[T_{off}^{pu}(c)]$ for channel c.

To calculate the additional delay, we take into account the following observations. First, communication is stopped during PUs' ON durations. Second, communication is delayed when the channel is shared with other SUs. In the worst case, a SU can be delayed for the entire duration during which other SUs access the channel. Third, communication is stopped during sensing periods. Finally, during the rest of the time, $T_{free} = T_{cycle} - (E[T_{on}^{pu}(c)] + E[T_{on}^{su}(c)])$, the communication continues without

additional delay due to DSA. Therefore, the average additional delay is given as:

$$\begin{aligned} D(c) = \frac{1}{\lambda_{pkt} T_{cycle}} \Bigg\{ & (\lambda_{pkt} E[T_{on}^{pu}(c)]). \frac{E[T_{on}^{pu}(c)]}{2} + \\ & (\lambda_{pkt} E[T_{on}^{su}(c)]). \frac{E[T_{on}^{su}(c)]}{2} + \\ & \lambda_{pkt}. (t_{sense}(c) r_{sense}(c) T_{free}). \frac{t_{sense}}{2} + 0 \Bigg\} \\ = u^{pu}(c). \frac{E[T_{on}^{pu}(c)]}{2} & + u^{su}(c). \frac{E[T_{on}^{su}(c)]}{2} + \\ & \{1 - u(c)\} r_{sense}(c). \frac{t_{sense}^2}{2}. \end{aligned} \tag{3}$$

In the above equation, $E[T_{on}^{pu}(c)]/2$ and $E[T_{on}^{su}(c)]/2$ are approximate estimates of the average extra delay when incumbents and other secondary devices access the channel. Sensing impact on delay, when the traffic is already stopped due to incumbent access or sharing, is not considered. Note that $\lambda_{pkt} T_{cycle}$ is the average number of application packets generated during the entire cycle for channel c.

To derive the additional jitter, again consider a cycle of duration $T_{cycle}(c)$. Additional jitter is typically introduced due to the difference in delays encountered when the channel is occupied by incumbents and other SUs, as compared to a completely free channel. We use the standard mean packet-to-packet delay variation (MPPDV) metric, which is the basis of jitter calculation in RTCP/RTP as defined in RFC 3550 [75]. The average extra jitter is given by:

$$\begin{aligned} J(c) = \frac{1}{\lambda_{pkt} T_{cycle}} \Bigg\{ & (\lambda_{pkt} E[T_{on}^{pu}(c)]). \frac{1}{\lambda_{pkt}} + \\ & (\lambda_{pkt} E[T_{on}^{su}(c)]). \frac{1}{\lambda_{pkt}} + \\ & \lambda_{pkt}. (t_{sense}(c) r_{sense}(c) T_{free}). \frac{1}{\lambda_{pkt}} + 0 \Bigg\} \\ = \frac{1}{\lambda_{pkt}} \Bigg\{ & u(c) + (1 - u(c)) t_{sense}(c) r_{sense}(c) \Bigg\}. \end{aligned} \tag{4}$$

We observe that bandwidth and additional delay estimates (Eqs. (4.2) and (3)) do not depend on packet-arrival rate λ_{pkt}. All the derivations above are based on uniform application packet arrival (which is valid considering the short epoch duration and QoS-sensitive traffic characteristics), but it can be shown that the semantic matching equations (4.2), (3), and (4) are the same for other distributions like the Poisson packet-arrival process.

As seen from our experiment results in Section 4.9.3.3, the semantic context dependency equations (4.2), (3), and (4) are sufficiently accurate in estimating the higher-layer QoS abstractions from lower layer information.

4.6 CASA Algorithm

CASA Algorithm (see Algorithm 3) executes at the beginning of every CASA Epoch τ_{casa}, after $B(c)$, $D(c)$ and $J(c)$ values are updated using the semantic dependency equations. At an abstract level, CASA Algorithm is a greedy reward-based algorithm and is loosely derived from Reinforcement Learning (RL) techniques, e.g., Q-Learning [74], in which rewards are assigned to actions based on past history of success.

RL techniques have been previously found to be useful in DSA optimization, e.g., for channel selection and spectrum sharing [44, 79]. Though we build upon the well-studied RL approach, CASA Algorithm is not a direct adaptation of any RL algorithm. Further, unlike such prior works in DSA, CASA Algorithm is utilized for optimizing DSA operations in relation to application QoS parameters.

CASA Algorithm strives to move DSA to a state where all application requirements are met by giving higher rewards to channels and DSA parameter combinations that achieve this goal to a higher degree. The abstraction of rewards is captured through the *push factor* value, which is explained below.

The algorithm first initializes the current push factor P_{curr} to 0, and reads in

Algorithm 3 CASA Algorithm

Require: $(B^{app}_{req}, B^{app}_{obs}), (D^{app}_{req}, D^{app}_{obs}), (J^{app}_{req}, J^{app}_{obs})$
Require: C, $\forall c \in C\ B(c), D)c), J(c), t_{sense}(c), r_{sense}(c)$
Ensure: C is sorted in increasing order of $u(c)$

Let $chan \leftarrow$ current channel
Calculate P_{past}
$P_{curr} \leftarrow 0$ {Initialize current push factor}
if $B^{app}_{obs} < B^{app}_{req}$ **then**
  $P_{curr} \leftarrow P_{curr} + 1$
end if
if $D^{app}_{obs} > D^{app}_{req}$ **then**
  $P_{curr} \leftarrow P_{curr} + 1$
  Reduce $r_{sense}(chan), t_{sense}(chan)$, if possible
end if
if $J^{app}_{obs} > J^{app}_{req}$ **then**
  $P_{curr} \leftarrow P_{curr} + 1$
  Reduce $r_{sense}(chan), t_{sense}(chan)$, if possible
end if
if $(P_{curr} + P_{past}) > 0$ **then**
  for each $c \in C, c \neq chan$ **do**
    if $B(c) \geq (1/\gamma_B) B^{app}_{req}$ **then**
      $P_{curr} \leftarrow P_{curr} + 1$
    end if
    if $D(c) \leq \gamma_D D^{app}_{req}$ **then**
      $P_{curr} \leftarrow P_{curr} + 1$
    end if
    if $J(c) \leq \gamma_J J^{app}_{req}$ **then**
      $P_{curr} \leftarrow P_{curr} + 1$
    end if
    if $t_{switch}(chan, c) \leq \gamma_D D^{app}_{req}$ **then**
      $P_{curr} \leftarrow P_{curr} + 1$
    end if
    if $(P_{curr} + P_{past})/(P_{MAX} + 1) > rand[0, 1]$ **then**
      Switch to channel c
      break
    end if
  end for
end if

the current application QoS requirements/observed values provided. The cumulative weight of previous push factors up to the last epoch (a history window of size N), is represented by P_{past}. We use simple exponential function to compute P_{past}. If 0 is

the most recent epoch and $N-1$ the least recent, then $P_{past} = \sum_{i=0}^{N-1} P_i w^i$, where w is the positive non-zero weight unit ($w < 1$). Clearly, the push factor in the most recent epoch has the highest weight and decreases for epochs in less recent epochs. This computation scheme captures the short-term correlation of near future with recent past in terms of the networking state experienced by the SU device. Note that $P_{past} + P_{curr}$ should be bounded above by P_{MAX}, and hence, P_{MAX} must be chosen based on window size N and weight unit w.

After the initialization step, CASA Algorithm performs multiple checks for potential application requirement violations and takes decisions to adapt DSA operations. In the first two decision-making steps, the algorithm adjusts r_{sense} and t_{sense} on the current channel, if feasible, in order to meet the delay and jitter requirements (D_{req}^{app} and J_{req}^{app}). This prevents violation of the these requirements during spectrum sensing.

In addition to adjusting DSA parameters, the push factor P_{curr} is concurrently updated by checking the extent to which the application requirements are satisfied. Any shortfall in meeting a requirement increases the push factor. Based on the combined push factor, $P_{curr} + P_{past}$, if there is a better channel available, CASA goes for a channel-switch in a probabilistic fashion. A random value in $[0, 1]$ is generated, and a channel-switch occurs if the generated value is less than $(P_{curr}+P_{past})/(P_{MAX}+1)$.

Clearly, a higher push factor implies better chances for a channel-switch, and vice versa. Probabilistic channel-switching prevents overcrowding a single channel, which could happen if multiple SU devices switch to a channel that is globally perceived to be the best at the moment. Note that this channel-switching invocation is independent of the deterministic channel-switching that may occur due to other DSA events. On every channel-switch, the device initializes the sensing parameters (r_{sense}, t_{sense}) to its highest possible values, so that sensing is most aggressive.

To allow for flexible operational control in practice, CASA Algorithm incorpo-

rates configuration parameters for its comparison process. For instance, γ_D acts as an administrative control knob that determines allowable limits on additional delay relative to end-to-end delay bound, depending on the deployment environment characteristics. For instance, in realistic deployments, γ_D would typically be less than 0.2. Similarly, γ_J and γ_B are the control knobs for jitter and bandwidth comparisons, respectively.

In summary, N, w, P_{MAX}, γ_D, γ_J, and γ_B constitute the configurable design parameters. CASA Algorithm has very low runtime and space complexity—linearly proportional to size of set C, thus reflecting the design principle of providing low overhead and easily deployed adaptation for DSA.

In some scenarios, there could be additional traffic during an epoch, e.g., due to legacy applications that may not provide any information to CASA. We approximate their impact by again relying on the principle of correlation of future with recent past. The key idea is to monitor the MAC packet queue to calculate the exponential moving average of the actual bandwidth at link layer, and compare it with the expected B_{req}^{app} value. If actual bandwidth is found to be greater, then their ratio is added to γ_B, γ_D, and γ_J.

4.7 Cross-Layer Interaction Framework (CLIF)

For CASA to be effective, an efficient cross-layer mechanism is needed to access the application layer context. This task is accomplished through CLIF (see Fig. 4.4), which exposes the interface for applications to export their hints.

CLIF consists of two components: the *Network Parameters Repository* (NPR), and the *Interface Functions*.

Network Parameters Repository is a central storehouse of parameters exported by the application layer. The link layer can query this module to get the parameter values, and hence, gain additional context information. NPR stores the parameters

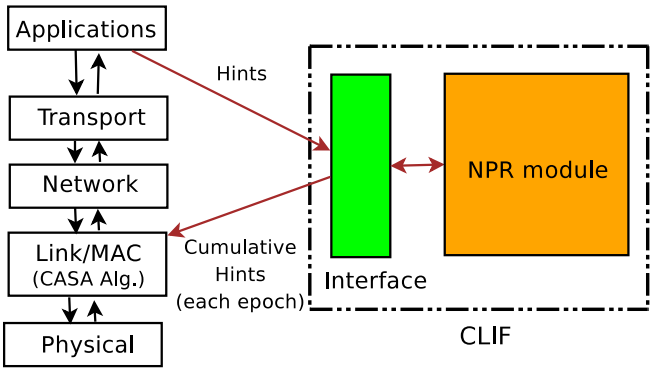


Figure 4.4: Schematic overview of CLIF in relation to the network stack.

```
ExportParameter(IN layer_id,IN param_id,IN param_value)
ImportParameter(IN layer_id,IN param_id,OUT param_value)
ImportParameter(IN param_id,IN op_type,OUT param_value)
UpdateParameter(IN layer_id,IN param_id,IN param_value)
RemoveParameter(IN layer_id,IN layer_id,IN param_id)
GetParameterList(OUT param_ids)
```

Figure 4.5: Important *Interface Functions* provided by CLIF.

in a two dimensional hash table for fast lookups and updates. This module also incorporates a processing component in order to calculate basic aggregation functions, e.g., adding up individual bandwidth requirements to determine B_{req}^{app}.

Interface Functions consists of a well-defined list of functions (Fig. 4.5) for accessing and updating the NPR.

4.8 Implementation

Fig. 4.6 shows our implementation model of CASA. We build CASA into the SU emulator implementation (described in Appendix B). CLIF is implemented as a loadable Linux kernel module. Applications link with a user-level library implementation of the CLIF's Interface Functions. The CLIF implementation also provides an *Application Adaptation Layer* (AAL) to simplify interaction with multiple applications. CLIF has been implemented in the kernel-space rather than user-space, in order to improve performance by reducing user-kernel boundary crossings. This is

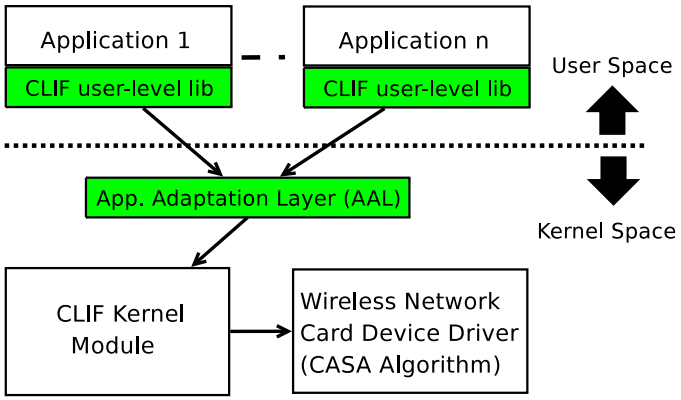


Figure 4.6: Implementation model of CASA.

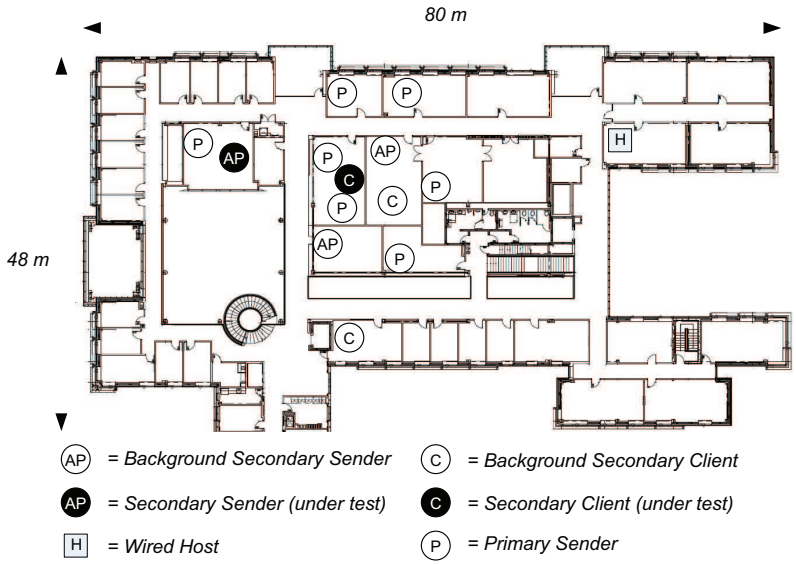


Figure 4.7: The testbed is deployed on 4th floor of the department building. Each primary network is on a different channel. Only important nodes are shown to reduce clutter in the figure.

because the frequency of access to CLIF module by lower layer resident CASA Algorithm is expected to be significantly higher than the rate of application context exports/updates.

Other details on the implementation of PU and SU features can be found in Appendix B.

4.9 Evaluation

4.9.1 Evaluation Metrics

We use the term "state-of-art DSA" to denote a traditional DSA protocol that does not utilize the CASA service architecture. We compared the performance of DSA operating with CASA against state-of-art DSA in terms of the following metrics.

- Average application throughput (i.e., goodput), delay, and jitter during an application session run.
- Bandwidth fulfillment quotient (F_b)—fraction of session time during which the application bandwidth requirement is satisfied.
- Delay fulfillment quotient (F_d)—fraction of session time during which the application delay requirement is satisfied.
- Jitter fulfillment quotient (F_j)—fraction of session time during which the application jitter requirement is satisfied.

Since the ultimate goal of DSA is to improve application performance, our metrics are application centric, not of a lower layer focus.

There are two important overheads introduced by CASA.

- Cross-layer communication delay (CD).
- Possible suboptimal channel selection and switches (CS).

CD occurs due to additional cross-layer communication related to export of application hints.

CS is introduced because CASA adjusts r_{sense} and t_{sense} as well as control channel-switch decisions in response to application requirements. A reduction in these parameters lead to stale information about channels, and may lead to poorer spectrum management decisions in DSA.

4.9.2 Testbed Setup

Our experimental setup consists of AP-client laptop pairs—each pair is a part of 802.11 infrastructure-type first/last-mile access network. In each experiment, at least two secondary networks and one primary network are active. Every secondary network is composed of 3-5 clients paired with an AP. The test AP (i.e., the AP of the secondary network being tested) is connected to the Internet via our University ethernet. At application level, the test client connects to a fixed host in the wired segment through the test AP. This setup is in accordance with our system model (see Section 1.2).

We deploy the fixed host on a different subnet of the university LAN in an attempt to increase the number of hops in the end-to-end communication path. However, the machines are part of the same campus network—resulting in lower end-to-end latency and jitter compared to those typical on the open Internet. Nevertheless, this setup is adequate for testing CASA—the observed trends and insights remain valid for the generic DSA deployments. Fig. 4.7 shows the testbed topology.

We mainly use `iperf` to generate traffic on secondary devices and to record the performance metrics. A custom script issues `iperf` commands and uses the CLIF user-level library to issue application context hints. The traffic requirement mirrors the typical requirement of QoS-sensitive networking applications (e.g., VoIP/video streaming) [76,80]. In particular, the bandwidth requirement is kept at 10Mbps which reflects the traffic demands of many simultaneously running high-quality multimedia communication applications. Also, we use delay requirement as 50ms, and jitter requirement as 2ms, based on typical single network SLA agreements [81]. Further, to compare actual consumer application performance with/without CASA, we use the open-source Ekiga softphone [82] (formerly known as GnomeMeeting) to generate videoconferencing sessions.

The spectrum C consists of seven channels (802.11a channels 36, 38, 40, 42, 44, 46,

and 48). There are seven incumbent networks, one on each channel. They generate different random ON/OFF traffic with the average $T_{ON} + T_{OFF} = 50$ms. Additional secondary devices are also enabled on the channels such that the average spectrum availability varies across the spectrum—70%, 60%, 50%, 40%, 30%, 20%, and 10% utilization on channels 36, 38, 40, 42, 44, 46, and 48 respectively. The raw physical-layer capacity is 54Mbps for channels 36-44, while it is reduced to two-third and one-third of 54Mbps for channels 46 and 48, respectively.

This spectrum setup emulates the expected spectrum environment for DSA networking, where certain licensed channels exhibit higher utilization (e.g., because of lower DSA access-cost/physical capacity ratio) while others have lower utilization (e.g., because of higher DSA access-cost/physical capacity ratio). Note that secondary market business model will play a major role in governing DSA networks, but they are still under active development [10, 60, 83] and beyond the scope of this chapter.

For emulating state-of-art DSA, we use $r_{sense} = 2$ (per second), and $t_{sense} = 0.05$s. We use the following CASA parameter values for all channels (unless otherwise noted):

1. $r_{sense} \in \{4, 2, 1, 0.5, 0\}$ (per second).
2. $t_{sense} \in \{0.1, 0.05, 0.025, 0.0125, 0\}$ (second).
3. $\tau_{casa} = 1$ (second).
4. $w = 0.5$, and $P_{MAX} = 6$.
5. $N = 8$ epochs.
6. $\gamma_J = 0.2$, $\gamma_D = 0.2$, $\gamma_B = 0.5$

DSA parameter values described here are based on their typical values in standard drafts, e.g., 802.22 [21]. Additional discussion on selecting the values for r_{sense} and t_{sense} can be found in [39, 84]. Values for CASA-related configuration parameters

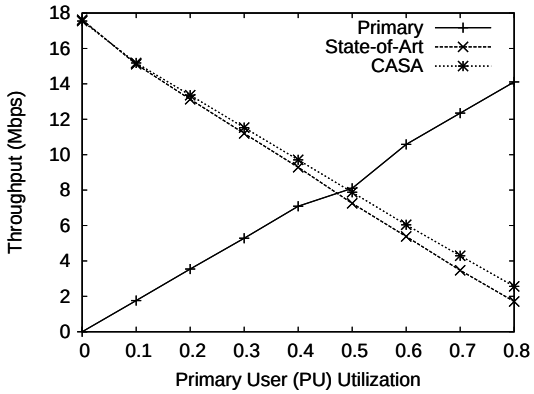


Figure 4.8: Throughput with variation in primary traffic volume.

(e.g., τ_{casa}) have been chosen keeping in view the values of key DSA parameters and through calibration from initial experiments.

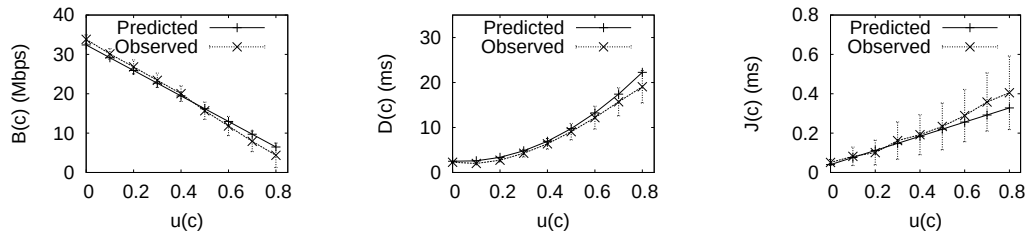
4.9.3 Results and Discussion

4.9.3.1 Overhead Analysis

We characterize the overhead CD through a timing study of CLIF system calls in the Linux kernel. The average delay with CLIF interface function calls is observed to be around 1μs on average. This is insignificant compared to acceptable network delays and traffic-burst time (which are in the order of tens of ms) of applications, especially as there are only a few interface function calls associated with each application communication session. Further, the additional memory space taken by CASA is found to be quite low (around 20kB).

4.9.3.2 Accuracy of DSA implementation

To test that DSA features are properly implemented (e.g., primary traffic must not be interfered) on base 802.11 MAC, we conducted simple microbenchmark experiments. In this experimental setting, we have one primary and one secondary network and the spectrum is limited to one channel (c = channel 36), with PHY capacity set at 24Mbps. Both the primary and secondary transmitters are saturated with UDP



(a) Bandwidth (b) Delay Increment (c) Jitter Increment

Figure 4.9: Analyzing the accuracy of semantic context matching equations.

traffic, i.e., they always have packets to transmit. We allow the primary transmitter to operate at a specified channel utilization. The rest of the channel-time is available for opportunistic secondary utilization. Each experiment run lasts for 2 minutes, and we conducted 100 test runs.

Fig. 4.8 shows the average throughput plotted against primary traffic load. As seen from the graph, the primary traffic gets preferential channel access, as it should, when DSA is deployed. The primary throughput is limited only by its own channel utilization. When more opportunities are available on the channel, the secondary network (both for state-of-art DSA as well as CASA-based DSA) shows greater throughput, as expected. Thus, apart from showing that our implementation is accurate, the result also shows the impact of primary traffic load on secondary communication.

Even in this simple scenario, using CASA results in better performance than state-of-art DSA, especially at higher primary utilizations. The throughput improvement is around 8% when PU utilizatin is 0.4, while the improvement margin increases to 51% when PU utilization is 0.8. This outcome is due to CASA dynamically managing DSA parameters to accommodate application traffic needs without violating DSA's constraints. This improvement can be even better if the spectrum consists of more channels—this is shown in results discussed later.

4.9.3.3 Accuracy of Semantic Matching

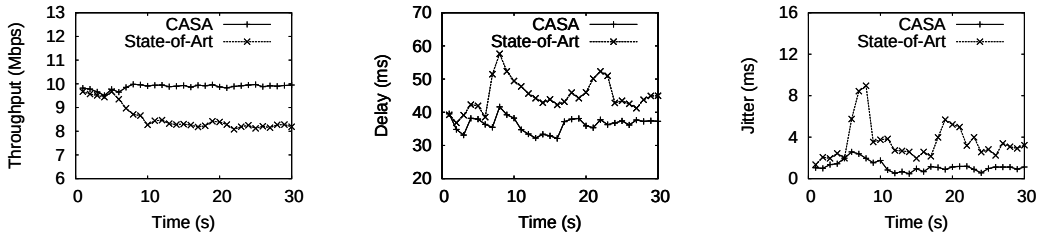
Semantic dependency equations are microbenchmarked in a controlled environment to ascertain their validity in terms of their estimation accuracy. For this purpose, we again use a single channel (c = channel 36) with one secondary network as the test candidate. The PHY layer capacity is set at 54Mbps. Both primary and background secondary transmitters are present on the channel. The total utilization $u(c)$ is changed for each set of experiments, with $u^{pu}(c) : u^{su}(c) = 4 : 1$ in each case.

Also, due to the *transit-reductive* nature of bandwidth parameter (see Section 4.4), saturation-level UDP traffic is used to quantify the maximum bandwidth available for different values of $u(c)$. For the delay/jitter case, we use UDP traffic at 10Mbps, with packet size of 500 bytes ($\lambda_{pkt} = 2500$). Further, $e(c) \approx 0.6$ for 802.11 MAC using UDP. CASA Algorithm is disabled—only semantic matching is performed. The sensing parameters, therefore, are fixed at $r_{sense} = 2s^{-1}$ and $t_{sense} = 0.05s$.

Fig. 4.9 shows that the observed values from experiments closely match the predictions from the semantic dependency equations. For each of bandwidth, delay, and jitter parameters, the average observed values (with 95% confidence interval) are within 8% of their predictions. Thus, the proposed semantic matching is found to be highly accurate. For higher channel utilizations ($\approx> 0.7$), the predicted values diverge slightly from the observed values. This arises primarily due to unaccounted MAC effects that are significantly more prominent at a very high contention level, e.g., unusually large backoff duration. However, the observed difference is very small, and further, very high utilization channels are unlikely to be used for DSA.

4.9.3.4 CASA Performance

To analyze the end-to-end performance of CASA, we use the setup described earlier in Section 4.9.2. Fig. 4.10 shows the temporal variation for throughput, delay and jitter in a 30s period starting from the beginning of the session. Each point on



(a) Throughput (b) Delay (c) Jitter

Figure 4.10: Temporal variation of QoS metrics.

the graphs denotes the average value over the past 1s window. Note that we measure the QoS metrics for several minutes, but skip the full duration in the plots as the long-term behavior is similar.

As seen from the graphs, CASA-based DSA produces significant improvements for each metric. In particular, throughput is found to dip starting at around 8s mark for state-of-art DSA. This corresponds to a channel-switch event to channel 48. State-of-art DSA is found to switch to a channel with lower utilization without considering the expected application QoS performance in the new channel. Thus, the throughput drops as the new channel cannot support the 10Mbps bandwidth. With CASA, while the secondary group still switches channels at approximately the same time, the new channel is selected such that it can support the application's bandwidth demand. Consequently, the throughput requirement is continued to be met throughout.

The end-to-end delay and jitter (Figs. 4.10(b) and 4.10(c)) is also found to be higher (with considerably more variation) for state-of-art DSA. Again, the reason for this is that CASA Algorithm results in a better channel selection, where packets are delayed less and the variability is lower. Adaptation of sensing parameters also contributes towards achieving better delay/jitter performance.

Fig. 4.11 shows the normalized values (together with the 95% confidence interval) for the QoS metrics relative to their requirements, for both state-of-art and CASA-based DSA. The observations are made over a 120s duration. Also, Fig. 4.12 shows the fulfillment quotient of the QoS demands (F_b, F_d, and F_j)over the same duration.

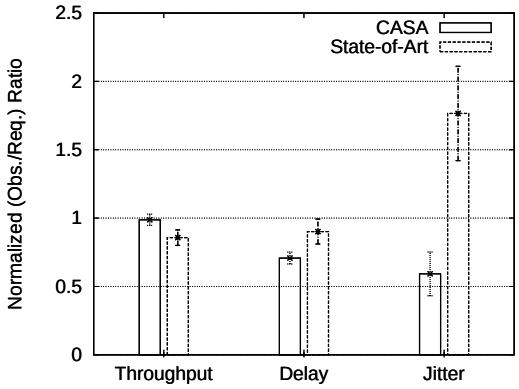


Figure 4.11: Performance comparison on QoS metrics. Observed average values are normalized w.r.t. required values and plotted.

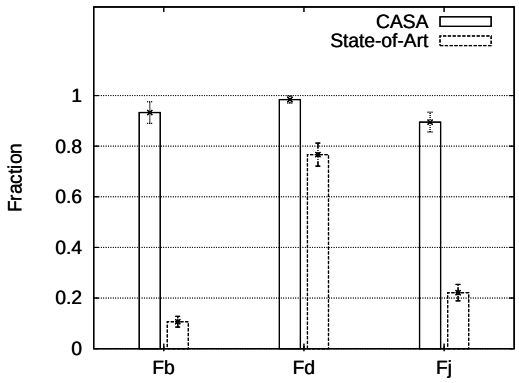


Figure 4.12: CASA is effective in sustaining QoS over the entire communication session.

From the two graphs, CASA-based DSA outperforms state-of-art DSA in fulfilling the requirements, especially in supporting QoS demands through the full duration of the communication session. For example, the throughput for CASA is ≈10Mbps (Fig. 4.11), as required, for more than 90% of the communication session (Fig. 4.12). Similarly, average delay and jitter are 30% and 64%, respectively, lower than the requirement for more than 90% of the session run time. CASA is observed to be especially effective in reducing jitter to a very low level as compared to state-of-art DSA. Thus, even the stringent jitter requirement of 2ms is satisfied for almost the entire session.

State-of-art DSA is also found to be somewhat effective in matching up to the

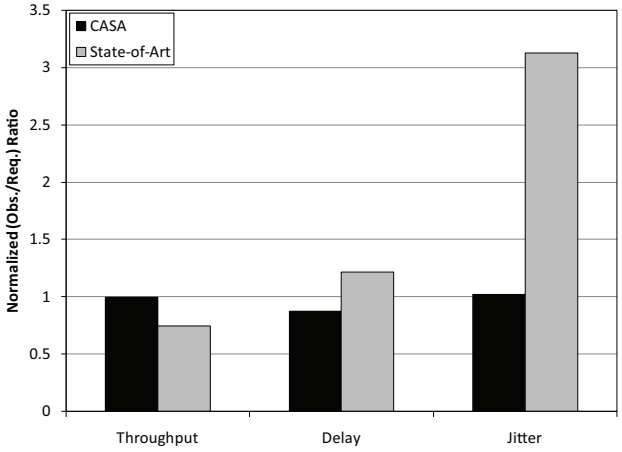


Figure 4.13: Performance comparison with Ekiga VoIP sessions. Observation is normalized w.r.t. to requirements.

delay metric. We attribute this to the topology bias of our testbed. In particular, the network path traversed by the packets is quite small (the end-hosts lie on the same network domain) with very few intermediate hops. Thus, the end-to-end delay typically does not exceed the application mandated requirement despite DSA side-effects. In general, we expect state-of-art DSA to perform significantly worse in the open Internet scenario, where the end-to-end delay is significantly higher. Thus, the benefits of CASA would be more pronounced than that demonstrated in our setup.

To study the performance of consumer-oriented applications, we run 6 videoconferencing sessions on the test secondary network using Ekiga VoIP softphone [82]. Each videoconferencing session requires around 460kbps (G.711 audio with H.261 video codecs used) of application throughput. Fig. 4.13 shows the comparison of performance (on QoS metrics) when CASA is used as compared to state-of-art DSA. With CASA enabled, the VoIP session is able to consistently achieve the required goodput demand—an improvement of 33% over state-of-art DSA. Similarly, the delay and jitter found to reduce by 28% and 67% respectively—thus illustrating CASA's ability to match up to QoS demands in realistic settings.

4.9.3.5 Effects of Design Parameters

Experiments were also conducted to study the impact of CASA's design parameters. In brief, following are the key observations.

Increasing P_{MAX} reduces channel-switches, and hence, applications do not get sufficient bandwidth in situations where a channel is shared by a large number of secondary users. Increasing the history window size N (beyond the current value of 8) does not lead to any significant change in the observed behavior. However, reducing N results in frequent channel-switches, especially when $N \leq 2$, resulting in poorer performance on all metrics. The effect of changing the epoch duration (τ_{casa}) is found to be similar to that of modifying N.

4.9.3.6 Summary of Evaluation Results

As is evident from the evaluation results, the overheads CD and CS (which are implicitly incurred in all the experiments) do not offset the advantages of CASA. The application centric context-awareness introduced by CASA improves application performance significantly. Average delay and jitter are improved by 30% on average. Also, CASA fulfills QoS demands for more than 90% of the duration of a communication session, which is more than triple of state-of-art DSA. Hence, CASA increases the resilience of DSA protocols in supporting application demands to a greater degree, especially in unfavorable environment.

4.10 Concluding Remarks

We argued for, and showed the importance of application-awareness in making DSA intelligent and more robust. We proposed a systems-based optimization mechanism together with the service architecture, called *Context Aware Spectrum Agility* (CASA), that combines application-awareness with channel-state knowledge in DSA.

CASA targets wireless client-type end-devices, and has been designed for easy deployment and implementation. The key component of CASA is the *CASA Algorithm*, which dynamically adapts DSA to accommodate specific application QoS requirements. We derived semantic dependency equations for matching application context with lower layer MAC-PHY information, which play a critical role in the CASA Algorithm. CASA can operate with any DSA protocol without introducing any significant modification or overhead. Our evaluation has shown CASA to increase the resilience of the DSA protocols significantly in supporting stringent application QoS requirements with intelligent, application-aware decisions. CASA illustrates that application-centric adaptivity of DSA parameters is effective in managing DSA side-effects, thereby enhancing the benefits of DSA in consumer wireless networks.

CHAPTER V

End-to-end Connection Management in DSA-based WLANs

5.1 Introduction

While Chapter IV proposed the CASA service architecture for application QoS support in DSA from a device-centric viewpoint, it is not sufficient for end-to-end communication. This is because CASA targets only one part of the network connection, i.e., the wireless device. Thus, it has only "local" scope for QoS management. In this chapter, we investigate into network level QoS support for DSANs which complements CASA in providing a complete QoS solution for DSA.

The primary function served by a majority of 802.11 WLANs (e.g., Wifi hotspots, home/office wireless networks) is to serve as the *first/last-mile access network* to the wired network cloud or the Internet, thus enabling the end devices to avail of networking services (e.g., web-access or VoIP) over the wireless medium. Hence, DSANs are expected to be utilized by consumers in an identical manner. Thus, DSANs must match and exceed the end-to-end performance of traditional WLANs in order to be commercially viable. This objective represents an important step towards effective integration of DSANs with existing networking infrastructure, which is important for the success of DSA.

Currently, there is a lack of "end-to-end" insights into DSA. In this chapter, we investigate issues involved in integrating a DSAN with rest of the networking infrastructure.

Need for connection management in DSANs: As noted earlier in Section 4.2, DSA entails operational constraints in a rapidly-changing spectrum environment. Therefore, it involves a number of functions and events that are disruptive to ongoing network traffic. Examples include spectrum sensing, channel switching, spectrum management & coordination, and incumbent activity. Apart from performance degradation at the link/PHY layers, such disruptive DSA-related phenomena can make long-term adverse impacts on the end-to-end communication. From a networking viewpoint, the transport layer is the first layer (from bottom of the networking stack) with true end-to-end semantics. We observe that the adverse impact of DSA on end-to-end connections is primarily a consequence of its negative side-effects at the transport layer. The main reason for this undesirable reaction is the ignorance of higher layer transport protocols about lower layer DSA semantics, as we discuss next.

TCP streams: Consider the example of incumbent activity on the licensed channel. A TCP connection between a server host in the cloud and a client on the DSAN can experience timeouts when the client cannot send out ACK packets in time, because of an ongoing DSA-induced quiet period on incumbent detection. Consequently, the TCP's congestion control mechanism will be unnecessarily invoked leading to further performance degradation. Other fundamental DSA functions like spectrum sensing and channel-switching also contribute to this negative impact. Such interruptions can be frequent, given (a) the regulatory restrictions imposed on unlicensed operations, (b) DSA service provider requirements, as well as, (c) unforeseen incumbent activity.

Techniques have been proposed in the past to address performance problems aris-

ing due to packet loss in the presence of high bit-error rate experienced on wireless medium, particularly for TCP [85, 86]. In modern WLANs, random wireless errors are not a significant problem, as the quirks of early-era wireless protocols have been addressed through more sophisticated error detection/correction schemes. In the context of DSANs, delays and losses arise due to DSA-related events, and the disruption could last significantly longer. Unlike random wireless errors, knowledge about many of the disruptive DSA events can be deterministically obtained, thus making a proactive approach feasible in masking their side-effects at the transport level.

UDP flows: While there have been few TCP connection management schemes for WLANs in the past (as noted earlier), not much work has been done for UDP-based network connections. UDP is a connection-less protocol, and by design, it does not address packet errors, delays, or losses. Thus, managing UDP flows was not considered of particular importance because applications requiring ordered delivery and reliability should use TCP instead. However, we argue that UDP connection management is now very important as UDP flows carry a significant portion of network traffic. The reason behind this is the tremendous growth in popularity of multimedia-based network services, e.g., video streaming, voice/video conferencing, etc., which typically use UDP to ensure timeliness of delivery. While such applications can tolerate some disruptions/losses, they are QoS-sensitive. End-to-end connections of such popular consumer-oriented applications should, therefore, experience minimal quality degradation when DSA is deployed.

Summary of our approach: In this chapter, we provide a comprehensive solution, called *DSASync*, to address the end-to-end performance issues when integrating a DSAN with the wired Internet cloud. DSASync is a network management framework for regulating TCP and UDP connections traversing the wired–wireless boundary. DSASync includes algorithms based on buffering and traffic-shaping to minimize adverse impacts on ongoing TCP streams as well as UDP flows. There are

two significant advantages of DSASync—it maintains the end-to-end semantics of the existing transport protocols (TCP/UDP), and ensures compatibility by not requiring any changes to their existing implementations.

DSASync exploits built-in control knobs for TCP streams, e.g., receive window size advertisement, to provide a complete TCP connection management solution. Unfortunately, UDP flows, being stateless, do not provide such ready-made hooks. Therefore, it is significantly more challenging to provide complete and non-intrusive connection management for UDP flows. In DSASync, we take a *higher layer* approach to address this problem by utilizing Real-time Transport Protocol (RTP) [87] features to manage the UDP-based connection. Although not all UDP flows use RTP, RTP over UDP is predominantly used for most real-time streaming applications like VoIP, where 100% reliability or in-order delivery is not required, but QoS is still important. In fact, our solution can be generalized for managing any connection that uses RTP, irrespective of the underlying transport protocol.

To the best of our knowledge, this is the first attempt to consider integration issues with deployment of DSANs. DSASync is designed to be compatible, scalable, and practical—a prototype implementation is also developed and evaluated in a testbed as part of this work.

Organization: The chapter is organized as follows. We describe the assumptions and key terms of this chapter in Section 5.2. Section 5.3 details the motivating issues for this work. DSASync details are presented in Section 5.4, with an implementation in Section 5.5. Experimental evaluation of DSASync is presented in Section 5.5. The chapter concludes with Section 5.6.

5.2 Assumption and Notation

As this work is aimed at the edge DSAN (the system model is described in Section 1.2), there is an implicit assumption that all the inbound/outbound network traffic

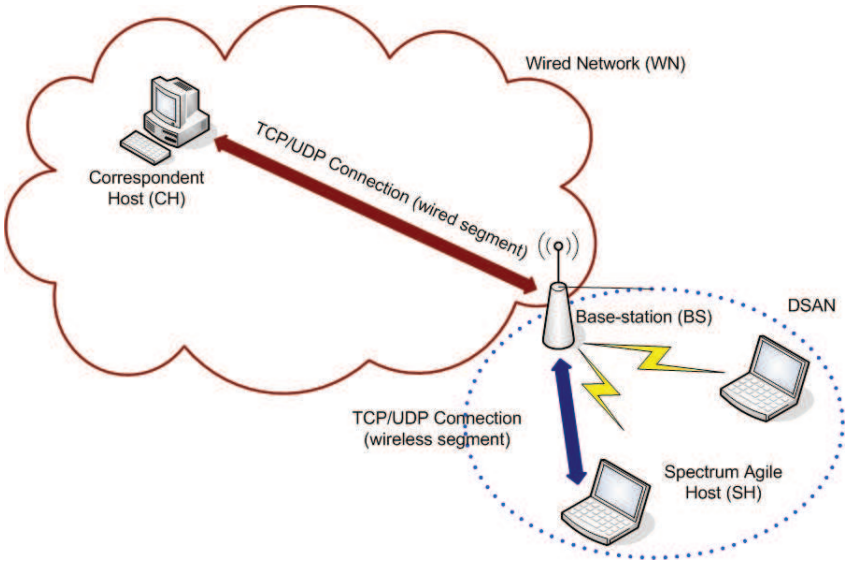


Figure 5.1: Typical end-to-end connection in an edge DSAN.

to/from the DSAN must pass through an access point or the base station (see Fig. 5.1. We utilize this feature to develop the traffic management algorithms. No restriction on DSA MAC-PHY protocol or spectrum sensing is assumed to retain the generality of DSASync.

We will use the following acronyms throughout the chapter.

- *Wired Network* (WN): The network cloud (e.g., the Internet) to which the wireless end-devices communicate to avail of network services.
- *DSA Network* (DSAN): A DSA-based wireless network that is connected to the WN.
- *Spectrum-agile Host* (SH): A DSA-enabled end-device in the DSAN that communicates with a device in the WN.
- *Correspondent Host* (CH): An end-device in the WN that communicates with a SH. The CH is usually a fixed host in the Internet cloud (WN) offering a network service, but may also be another peer wireless device communicating with the SH.

- *Base Station* (BS): The designated device (or access point) that connects the DSAN to the WN. All communications from DSAN to WN and vice-versa must pass through BS.

- *Transmission Freeze Period* (TFP): The duration during which packet transmission is halted by one or more SHs, or by the entire DSAN due to DSA-related events.

5.3 Motivation

As detailed earlier in Section 4.2, DSANs exhibit several characteristics that can adversely impact application QoS, e.g., spectrum sensing, incumbent protection, etc. Along similar lines, their undesirable side-effects also hinder DSAN's effectiveness in functioning as an edge access network. This is because the end-to-end connections to/from DSANs face frequent disruptions due to such DSA-related factors. While CASA provides a device-centric solution to provide QoS support, it does not solve the overall QoS degradation problem in DSANs because CASA does not consider end-to-end connection performance. For effective functioning of DSANs, particularly when they serve as edge access networks, it is important to provide network level support for QoS—which is the objective of this work.

DSA is fundamentally disruptive for ongoing wireless communication, especially on a short-term scale, which cannot be fully eliminated. Therefore, the design principle of DSASync is to carefully manage end-to-end connection flows in order to minimize the impact of the aforementioned disruptive events experienced in the DSAN. Since transport layer forms the basis of end-to-end connections and directly impacts application performance, our solution targets two core transport protocols—TCP and UDP.

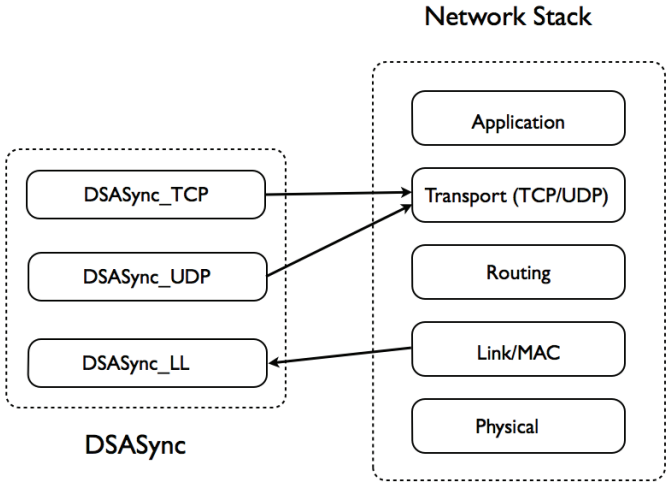


Figure 5.2: Architectural overview of DSASync.

5.4 DSASync

DSASync is logically a link-layer network management protocol (similar to Snoop Agent [85]). However, DSASync manages TCP/UDP connections—Fig. 5.2 shows DSASync's architecture schema. Certain mechanisms of DSASync, e.g., packet buffering during TFPs, are utilized for any end-to-end network connection (whether TCP or UDP) without restrictions. However, unlike TCP, not all UDP flows are fully managed—only those that are part of RTP-based communication are provided the complete benefits of the proposed connection management solution.

To achieve traffic flow management, DSASync sniffs the packets in transit (at the BS), and maintains state information (e.g., last ACK copy, sequence #s, etc) for each ongoing TCP stream it detects. Similarly, it maintains some state information (e.g., copies of the latest *sender report* and *receiver report* packets) for each RTP-based UDP flow identified.

5.4.1 DSASync: Link Layer

The DSASync LL component (`DSASync_LL`) is the "information monitoring" unit of DSASync, which collects and maintains information about DSA parameters required by DSASync. The parameters of interest are as follows.

1. N = total number of wireless nodes associated with the BS in the DSAN.
2. $f_{DSAN}^{sense}(t)$ = the frequency of spectrum sensing by the entire DSAN. This parameter usually corresponds to the cooperative sensing schedule in which all nodes participate.
3. $t_{DSAN}^{sense}(t)$ = the duration of each spectrum sensing event scheduled by the DSAN.
4. $f_i^{sense}(t)$ = the frequency of additional sensing (e.g., out-of-band) sensing performed by node i.
5. $t_i^{sense}(t)$ = the duration of each node-specific sensing event at node i.
6. $f_{DSAN}^{switch}(t)$ = the frequency (rate) of channel switches.
7. $t_{DSAN}^{switch}(t)$ = the delay involved in each channel switch.
8. $g_{PU,ON}(t)$ = the PU's ON time distribution.
9. S_{DSAN} = the Boolean parameter indicating if sensing is currently ongoing in the DSAN.
10. SW_{DSAN} = the Boolean parameter indicating if the DSAN is currently performing a channel-switch.
11. S_i = the Boolean parameter indicating if sensing is currently ongoing at the node i.
12. PU_{ON} = the Boolean parameter indicating if there is currently a PU activity on the current channel.

These parameters are part of any DSA protocol, and are typically available at the link layer. For example, most DSA protocols (specifically—their sensing components) estimate PU ON/OFF distribution in order to enhance DSA performance [39, 88, 89]. Hence, the distribution $g_{PU,ON}(t)$ can be obtained. In the rare scenario where a parameter is not directly available from the DSA protocol, `DSASync_LL` uses history-based estimates for each of them. In practice, the actual function definitions/forms for the parameters are not needed. Rather, the knowledge of their current values is sufficient.

5.4.2 TCP Management

The main task of the TCP management component, `DSASync_TCP`, is to utilize the information collected by `DSASync_LL` in managing both downlink (from CH to SH) and uplink (from SH to CH) TCP traffic. The objective is three-fold: (a) to minimize packet loss, (b) to minimize time-outs and hence, retransmissions, (c) to adjust TCP connection parameters in response to changes in available bandwidth. In the basic design, `DSASync_TCP` executes only at the BS, as the BS has all the necessary information and the incoming/outgoing traffic must pass through it (see Section 5.2 and 5.4.4). `DSASync_TCP` consists of 3 modules—`DSASync_TCP_CH-SH`, `DSASync_TCP_SH-CH`, and `DSASync_TCP_CAP`.

5.4.2.1 DSASync_TCP_CH-SH

This module buffers the downlink (CH-SH) TCP packets for destination wireless nodes during the TFPs. The current state of the destination node (w.r.t. its packet-reception capability) is known from `DSASync_LL`. The buffered packets are then transmitted from the BS to the SH when the transmission can be resumed.

Due to limited buffer space at the BS, it is possible to run out of space before the transmission is resumed. This can happen, for example, if the DSAN is blocked

Algorithm 4 Algorithm TCP_CH-SH-a

Require: B_{free}^{tcp}, B_{low}^{tcp}, $hold$

```
1: p ← incoming CH-SH pkt
2: dest ← destination SH of p
3: src ← source CH of p
4: conn ← p's TCP connection identifier
5: TFP ← SW_DSAN|S_DSAN|S_dest|PU_ON
6: if TFP = 0 then
7:    Add p to transmit queue
8: else
9:    Buffer pkt
10:   if hold = false then
11:      if B_free^tcp < B_low^tcp then
12:         hold = true
13:         for each TCP connection do
14:            Advt. zero rwin to sender CH
15:         end for
16:      else
17:         if SN_new^conn = SN_last^conn + rwin^conn then
18:            Advt. zero rwin src for conn
19:         end if
20:      end if
21:   else
22:      if p=window update request then
23:         Advt. zero rwin to src for conn
24:      end if
25:   end if
26: end if
```

from transmission for a long period due to ongoing incumbent activity. Also, in such a situation the CH may also time out waiting for any ACK from the SH, thus incurring unnecessary retransmission overhead. To prevent this situation, `DSASync_TCP` attempts to proactively pause the sender by exploiting the built-in flow control mechanism of TCP.

Let the allocated space (at the BS) for buffering downlink (CH-SH) TCP packets be B_{alloc}^{tcp}. B_{low}^{tcp} & B_{high}^{tcp} are the configuration parameters for TCP buffer space thresholds, where $B_{alloc}^{tcp} > B_{high}^{tcp} > B_{low}^{tcp}$. B_{free}^{tcp} is the current free buffer space for TCP packets. For the TCP connection $conn$, SN_{last}^{conn} is the latest sequence # acknowl-

Algorithm 5 Algorithm TCP_CH-SH-b

Require: B_{free}^{tcp}, B_{high}^{tcp}, $hold$
1: **if** $hold = true$ **then**
2: **if** $B_{free}^{tcp} > B_{high}^{tcp}$ **then**
3: $hold \leftarrow false$
4: **end if**
5: **end if**
6: **for** $i \leftarrow 1$ to N **do**
7: $TFP \leftarrow SW_{DSAN}|S_{DSAN}|S_i|PU_{ON}$
8: **if** $TFP = 0$ **then**
9: Unbuffer any i's pkt to transmit queue
10: **else**
11: Buffer any i's pkt from transmit queue
12: **end if**
13: **end for**

edged by the SH, SN_{new}^{conn} specifies the sequence # of the latest data packet (coming from CH) buffered at BS, and $rwin^{conn}$ is the latest advertised receive window.

The BS uses the procedure outlined in Algorithms 4 and 5—both executed in parallel—in order to manage CH-SH TCP traffic. Algorithm 4 is executed when a TCP packet is received from the CH and destined for a node in the DSAN, i.e., for each TCP packet about to be added to the outgoing queue at the BS's wireless interface. Algorithm 5 is executed periodically, based on a sufficiently frequent timer interrupt. A separate process updates buffer sizes (when a packet is added/removed) and also overwrites the *rwin* field to 0 in outgoing packets, if *hold* parameter (see Algorithms 4 & 5) is true.

Algorithm 4 exploits the TCP flow control mechanism to avoid buffer overflow (and hence dropped packets) at the BS. The BS advertises a 0-size receive window on behalf of the SH, when the buffer threshold is reached. The same strategy is used to prevent retransmissions (due to timeouts at the sender CH), when the receive window becomes full while still in TFP. The algorithm does not prevent timeouts or retransmissions at the end-points due to non-DSA factors, such as congestion in the network. However, an older packet is replaced with a newly-arrived packet with the

same sequence #, i.e., when a duplicate packet arrives.

It is also possible to manage CH-SH TCP traffic by sending out ACKs to the CH on behalf of the SH, or even splitting TCP connections at the BS. However, DSASync does not take these approaches for two reasons. First, it will violate end-to-end semantics of TCP data flow, e.g., a successful reception of ACK at CH (the source) will no longer imply that the packet has successfully reached SH (the destination). Second, sending ACKs will likely result in receiving more packets during the TFP, which may lead to buffer space getting filled up earlier. Further, the resource overhead will be higher.

5.4.2.2 DSASync_TCP_SH-CH

TCP performance degrades due to irregular behavior in the reception stream. For example, we observe timeouts and retransmissions when a TFP sets in, as CH often does not receive ACKs in time according to its RTT estimate—which is typically quite low as it was based on continuous packet reception during the past non-TFP period. Further, a "start-and-stop" type of data packet reception also contributes to other QoS issues, such as increased application jitter.

To minimize this connection degradation for the uplink (SH-CH) TCP stream, the BS attempts to "smooth" the outgoing flow. The key idea is to spread the uplink packets over the TFPs, so that the CH sees a relatively steady stream of packets despite the disruption at the source SH. Thus, the temporal discontinuities in packet reception are masked. This is accomplished as follows.

Given the information available from `DSASync_LL`, the average fraction of TFPs for node i can be estimated. Consider a time-interval, say $[T - \triangle T, T]$. The total TFP for node i during this $\triangle T$ time window is the sum of delays (on average) due

to sensing, switching and PU activity interruptions.

$$\begin{aligned} TFP_i^{avg} = & E[f_{DSAN}^{sense}(t).t_{DSAN}^{sense}(t).t]_{t=T-\triangle T}^{t=T} \\ & + E[f_i^{sense}(t).t_i^{sense}(t).t]_{t=T-\triangle T}^{t=T} \\ & + E[f_{DSAN}^{switch}(t).t_{DSAN}^{switch}(t).t]_{t=T-\triangle T}^{t=T} \\ & + E[g_{PU,ON}(t).t]_{t=T-\triangle T}^{t=T} \end{aligned}$$

Therefore, the fraction of non-TFP period for node i during $[T - \triangle T, T]$ is given by

$$\alpha_i = 1 - \frac{TFP_i^{avg}}{\triangle T}. \tag{5.1}$$

In practice, historical information on TFP durations during a moving time-window of size $\triangle T$ can be utilized to compute the α_i value for each i.

Let α_{min} be the administrative configuration parameter to limit the extent of traffic shaping. In order to manage uplink TCP traffic the BS executes Algorithm 6. Algorithm 6 outlines the dequeueing process for the outgoing queue at the BS's wired interface. The algorithm modifies the rate of a wireless node src's outgoing TCP packets as:

$$D_{eff,src} = \beta_{src}.D_{src}^{out,tcp} \tag{5.2}$$

where $\beta_{src} = max(\alpha_{src}, \alpha_{min})$, and $D_{src}^{out,tcp}$ is the actual data-rate at which src's outgoing TCP packets are received at the BS. Thus, linear traffic-shaping is applied to the uplink TCP traffic. The α_{min} configuration parameter provides administrative control over (a) excessive delays (and hence very high response-times for applications), and (b) buffer space run-out.

$D_{src}^{out,tcp}$ is easily estimated by monitoring the rate at which node src's TCP data packets enter the BS's outgoing queue (at its wired interface) in the moving time-

Algorithm 6 Algorithm TCP_SH-CH

Require: α_{min}, $D_i^{out,tcp}$, α_i, T_i, $dequeue_i$ $(\forall i \in N)$

while SH-CH queue is non-empty **do**
  $p \leftarrow$ 1st TCP pkt in queue
  $done \leftarrow false$
  $count \leftarrow 1$
  while $done = false$ and $count \leq N$ **do**
    $src \leftarrow$ source of p
    if $dequeue_{src} = false$ **then**
      $\beta_{src} \leftarrow \max(\alpha_{src}, \alpha_{min})$
      $T_{src} \leftarrow$ timestamp of src's last TCP pkt dequeue
      $T_{curr} \leftarrow$ current timestamp
      $elapsed \leftarrow T_{curr} - T_{src}$
      if $\{size(p)/(elapsed) < \beta_{src} D_{src}^{out,tcp}\}$ **then**
        $dequeue_{src} \leftarrow true$
        $done \leftarrow true$
      end if
    end if
    $count \leftarrow count + 1$
    $p \leftarrow$ next TCP pkt in queue
  end while
end while

window $\triangle T$. Similarly, T_{src} and $dequeue_{src}$ are local variables (used in Algorithm 6) that are updated by monitoring the dequeue events. Note that the packets enter the queue in the temporal order they are received, as before. The local variables *done* and *count* ensure that the number of iterations is bounded while searching for a packet that can be dequeued.

In practice, the node-specific sensing duration will not vary significantly for different nodes. This is because the sensing technology across devices is expected to be similar, and the spectrum environment is also similar across the single-hop edge DSAN. It may also turn out to be the least dominating fraction in the α_i calculation (equation (5.1))—$E[f_i^{sense}(t).t_i^{sense}(t).t]_{t=T-\triangle T}^{t=T}$ can be very small compared to other terms like incumbent activity and collective sensing duration. Thus, each α_i, $\forall i \in N$ can be closely approximated by the average of α_i values, say α_{dsan}. Further, since the packets from nodes are queued on a first-come-first-serve basis, the individual data

Algorithm 7 Algorithm TCP_SH-CH-OPT

Require: α_{dsan}, $D_{dsan}^{out,tcp}$, T_{dsan}, $dequeue_{dsan}$

while SH-CH queue is non-empty **do**
  if $dequeue_{dsan} = true$ **then**
    Wait for dequeue to complete
  end if
  $p \leftarrow$ 1st TCP pkt in queue
  $\beta_{dsan} \leftarrow \max(\alpha_{dsan}, \alpha_{min})$
  $T_{dsan} \leftarrow$ timestamp of last TCP pkt dequeue
  $T_{curr} \leftarrow$ current timestamp
  $elapsed \leftarrow T_{curr} - T_{dsan}$
  if $\{size(p)/(elapsed) < \beta_{dsan} D_{dsan}^{out,tcp}\}$ **then**
    $dequeue_{dsan} \leftarrow true$
  else
    Wait $\{(size(p)/\beta_{dsan} D_{dsan}^{out,tcp}) - (elapsed)$ interval
  end if
end while

rates can now also be replaced by the overall incoming data-rate $D_{dsan}^{out,tcp}$.

Hence, Algorithm 6 can be further optimized to yield Algorithm 7. The revised algorithm has a lower implementation and run-time overhead, because it has to maintain fewer state variables. However, the most significant gain is due to reverting back to traditional queue semantics (which has an $O(1)$ dequeueing process, albeit at the "traffic-shaped" rate) for the SH-CH queue in Algorithm 7.

5.4.2.3 DSASync_TCP_CAP

TCP's flow and congestion control mechanism allows its adaptation to gradual capacity changes in the network. Thus, small capacity fluctuations, typically encountered on the same channel, do not warrant any special handling. However, during channel-switches—where substantial and sudden capacity decrease may occur—this adaptation can be prolonged. When there is a significant loss of capacity, there can be substantial packet losses and retransmissions in the process.[1]

[1]When the channel capacity increases, the TCP performance gradually improves by itself. Therefore, DSASync does not take any action in this case.

Algorithm 8 Algorithm TCP_CAP

Require: C, e_{tcp}, $D_i^{in,tcp}$ $(\forall i \in N)$

1: **for** $i \leftarrow 1$ to N **do**
2: **if** $D_i^{in,tcp} > e_{tcp}C$ **then**
3: Send 3 duplicate ACKs to i's CHs
4: **end if**
5: **end for**

For the downlink (CH-SH) traffic, the procedure outlined in Algorithm 8 is executed when a channel-switch event is indicated (through `DSASync_LL component`). In this algorithm, C is the raw physical-layer bandwidth on the new channel, while e_{tcp} is the data transfer efficiency for TCP with the DSA MAC-PHY protocol to be used in the new channel. For example, various studies have shown that $e_{tcp} \approx 0.5$ over 802.11. $D_i^{in,tcp}$ denotes the downlink TCP date-rate for node i. Note that $D_i^{in,tcp}$ is calculated by a sliding time-window based historical averaging of TCP packets received for node i at the BS.

Algorithm 8 triggers TCP's fast retransmit/recovery by sending at least 3 duplicate ACKs to the CH, if the current downlink data-rate for a node cannot be sustained on the new channel. The objective is to prevent slow recovery where *cwnd* is reduced to 1, instead of half of the current value as in fast recovery. Thus, the sender will automatically reduce its sending data-rate with less severe impact than would otherwise occur. Note that we avoided the TCP window scale option to manage such capacity changes, as they are optional—many network routers and firewalls do not implement this feature. In contrast, fast retransmit/recovery feature is a part of most TCP implementations (e.g., TCP Reno) that are commonly used in modern operating systems.

DSASync does not take any action for SH-CH uplink traffic when capacity decreases on a channel-switch, as the uplink data-rate from the source SH is automatically curtailed (due to change in raw network capacity) to reflect the change.

5.4.3 UDP Management

UDP traffic is managed along similar lines as TCP streams, through the `DSASync_UDP` component. However, there are several fundamental differences between them. Although certain TCP connection management techniques like packet buffering is applicable to UDP flows, UDP connections cannot be managed intrinsically because it is stateless and does not provide built-in connection management knobs (unlike TCP). We do not wish to modify the UDP protocol itself (or introduce a new one) to add the required control hooks, as it goes against DSASync's key design principles of compatibility and easy deployment.

However, we observe that UDP is being increasingly used for QoS-sensitive traffic in the Internet today. In particular, applications like multimedia streaming and VoIP, which favor timeliness rather than reliability, use UDP. This motivated us to provide a complete UDP flow management for DSANs, rather than just settle for the buffering mechanism.

Note that most QoS-sensitive UDP-based network applications rely on Real-time Transport Protocol (RTP) [87]. RTP is an application-layer component that consists of two components: (a) RTP Data Transfer Protocol is responsible for application-level framing and delivery, (b) RTP Control Protocol (RTCP) provides QoS feedback of the data stream. Clearly, for UDP flows that are part of RTP-based communication, we can utilize the higher layer RTP information to manage the connections to a significant extent. In this section, we present techniques that exploit RTCP's features to manage the underlying end-to-end UDP traffic.

`DSASync_UDP`, manages UDP connections by using a combination of buffering at the BS (like that for TCP), and opportunistic modification/generation of RTCP packets —*Receiver Report* (RR) and *Sender Report* (SR) (for those UDP flows that are based on RTP). Like `DSASync_TCP`, `DSASync_UDP` sniffs packets in transit to identify active RTP sessions and maintains their meta-data. This passive RTP connection

identification has some limitations, which we discuss in Section 5.4.5. To simplify the algorithms and implementation, the buffer space for UDP traffic is kept separate from that of TCP streams (although it is not strictly necessary).

Note that unlike TCP, RTCP cannot directly influence the ongoing UDP-based connections. It is a passive feedback mechanism and it is up to the applications themselves to take any action in the event of RTCP feedback. Thus, the connection management for RTP-based UDP flows cannot be as responsive as that for TCP streams. The objective here is to (a) minimize the packet losses due to DSA interruptions, and (b) ensure that RTCP feedback due to DSA-related disruptions is available in a timely manner to the applications. Thus, any adverse impact on such QoS-sensitive UDP flows (which traditionally use RTP) can be potentially reduced.

`DSASync_UDP` consists of 3 modules — `DSASync_UDP_CH-SH`, `DSASync_UDP_SH-CH`, `DSASync_UDP_CAP`.

5.4.3.1 DSASync_UDP_CH-SH

`DSASync_UDP_CH-SH` module manages the downlink UDP traffic by using the procedure outlined in Algorithm 9. Like its TCP counterpart, Algorithm 9 is executed when a new UDP packet is received from the CH and destined for a node in the DSAN. Like `DSASync_TCP_CH-SH`, `DSASync_UDP_CH-SH` buffers (and later transmits) the incoming UDP packets during the TFPs based on information available from `DSASync_LL`. However, there is a key difference in buffering semantics between `DSASync_TCP_CH-SH` and `DSASync_UDP_CH-SH`. As stated earlier, applications using UDP traffic emphasize timeliness and are somewhat loss-tolerant. Thus, Algorithm 9 favors new packets over the older (and already buffered) packets when the buffer space gets full. Therefore, the oldest packets are purged from the buffer to create space for the newly-arrived packets.[2]

[2]Note that we do not have any protocol option, like *receive window* in TCP, to immediately quench the flow from CH. Further, there is no retransmission in UDP.

Algorithm 9 Algorithm UDP_CH-SH-a

Require: B_{free}^{udp}

```
1: p ← incoming CH-SH UDP pkt
2: dest ← destination SH of p
3: src ← source CH of p
4: conn ← p's RTP connection id, if RTP-based flow
5: ctime ← current time
6: rr_time ← timestamp of last RR for conn
7: rr_int ← avg. RR transmit interval for conn
8: TFP ← SW_DSAN|S_DSAN|S_dest|PU_ON
9: if TFP = 0 then
10:    Add p to transmit queue to DSAN
11: else
12:    if B_free^udp = 0 then
13:       Flush oldest packets from buffer, to accommodate p
14:       Update any SRs, RRs in the outgoing SH-CH queue
15:       if conn is valid then
16:          if ctime − rr_time ≥ rr_int then
17:             if No RR for conn exist in SH-CH outgoing queue then
18:                Generate a new RR for conn
19:             end if
20:          end if
21:       end if
22:    else
23:       Buffer p
24:    end if
25: end if
```

Algorithm 10 Algorithm UDP_CH-SH-b

```
1: for i ← 1 to N do
2:    TFP ← SW_DSAN|S_DSAN|S_i|PU_ON
3:    if TFP = 0 then
4:       Unbuffer any to-i pkt to transmit queue
5:    else
6:       Buffer any to-i pkt from transmit queue
7:    end if
8: end for
```

Apart from buffering (as seen from Algorithm 9), `DSASync_UDP_CH-SH` provides quick feedback to the sender CH about ongoing QoS degradation on a RTP session, whenever feasible,[3] to limit the packet losses when the buffer space fills. This feedback

[3]There are limits on the amount of RTCP feedback (RR/SR packets), typically limited to 5% of

is provided by updating an existing RR in the outgoing queue towards CH, or in its absence, generating a completely new RR. To avoid unnecessary overhead during packet reception, only the RR for the session corresponding to the newly-received packet is generated. Also, a separate process periodically updates all the uplink RRs to accurately reflect the loss, delay, and jitter encountered in the DSAN.

Algorithm 10 is the companion algorithm for buffer freeze/unfreeze, and is executed in parallel with Algorithm 9. Algorithm 10 is similar to `DSASync_TCP`'s Algorithm 5. However, it doe not maintain additional state variables (like *hold*) because of the different buffering strategy for UDP packets, as described earlier.

Remarks on updating SRs/RRs: In updating RTCP feedback packets, we follow a set of rules to prevent violation of RTP semantics and its end-to-end principle.

1. The SRs and RRs originating from the remote CH are not modified, because they represent the state at the opposite end of the connection. They constitute important feedback information for the application at SH.

2. The "sender info" part of any SR generated by the SH remains unchanged throughout the process (see [87] for RTCP SR/RR packet format).

3. Only the *packet lost fraction*, *number of packets lost*, and *jitter* fields are updated, when needed, for the QoS report blocks contained in SR/RR packets sent from the SH. For example, when downlink (CH-SH) packets are lost at BS due to insufficient buffer space, any SRs/RRs in the uplink (SH-CH) outgoing queue are updated to provide quick feedback to CH.

Note that the procedure to update the individual parameters in SR/RR packets is based on the formulas described in [87]. The additional delay and loss due to TFPs are incorporated in the calculation process to update values of loss fraction and jitter.

the session bandwidth [87].

Algorithm 11 Algorithm UDP_CAP

Require: C, e_{udp}, $D_i^{in,udp}$ $(\forall i \in N)$

1: **for** $i \leftarrow 1$ to N **do**
2: **if** $D_i^{in,udp} > e_{udp}C$ **then**
3: Update loss parameters for i's RRs/SRs in outgoing (CH-SH) queue
4: **end if**
5: **end for**

5.4.3.2 DSASync_UDP_SH-CH

The strategy for the uplink RTP-based UDP flows is identical to that for uplink TCP streams. The key idea is to apply linear traffic-shaping to UDP flows in order to mask the DSA-induced interruptions. The goal is to improve the QoS metrics, especially jitter, for the uplink flow. This is particularly useful when the SH is the main sender of the end-to-end connection.

The traffic-shaping algorithm employed is same as that for TCP (described in Section 5.4.2.1, Algorithm 6) and is based on Eqs. (5.1) and (5.2). The optimized version of the algorithm is also identical (see Algorithm 7). We leave out the pseudocode in the interest of space.

5.4.3.3 DSASync_UDP_CAP

Algorithm 11 is executed for UDP flows (that are RTP-based) on a channel-switch, along similar lines as Algorithm 8 for TCP. The difference lies in the capacity change feedback mechanism. While it is possible to exploit fast retransmit/recovery mechanism for TCP, we rely on RTCP's feedback mechanism for UDP. When the current UDP data-rate for a node is found to be definitely unsustainable on the new channel (line 2 of Algorithm 11), then its outgoing SRs and RRs are proactively updated by the predicted increase in loss due to reduced bandwidth, i.e., by a factor of $e_{udp}C/D_i^{in,udp}$. Here e_{udp} is the data transfer efficiency for UDP over the DSA MAC-PHY protocol being used in the new channel. The goal is to provide quick notification to the sender CH about imminent loss, so that it can take corrective

action. Other optimizations for this algorithm are discussed in Section 5.4.4. As for TCP, DSASync provides capacity change management only for the downlink traffic.

5.4.4 Possible Extensions

Several additional optimizations and enhancements can be built upon the basic DSASync platform presented above in Sections 5.4.1–5.4.3. These enhancements may require domain-specific knowledge about the target DSANs or their expected traffic features based on historical trends. We discuss below two such possible extensions.

5.4.4.1 Per-node DSASync

DSASync agents on wireless client nodes in the DSAN can further help in minimizing losses and improving other QoS metrics. This approach essentially amounts to a distributed architecture of DSASync. Per-node DSASync agents can be implemented using a similar buffer management strategy as the DSASync buffer on the BS. Local DSASync agents will help uplink (SH-CH) traffic bandwidth in particular, as the outgoing packets won't be as easily lost or dropped at the SH itself (during TFPs). However, wireless client nodes can have significantly low resource availability (e.g., a basic smartphone) and this feature may not be feasible or be very limited in usefulness. Further, it can be difficult to ensure that all client wireless nodes implement or use this feature because of their mobility and heterogeneity. For example, different kinds of wireless devices may utilize a DSAN for short periods and then move away. Thus, we consider this as an optional extension of the standard DSASync design.

5.4.4.2 QoS feedback optimization

Depending on the traffic characteristics of the DSAN, the QoS feedback policy of DSASync can be optimized. For example, if there is a substantial presence of non-TCP (or non-UDP) type of traffic through a DSAN, then their data-rate must be

taken into account in determining if the channel capacity is sufficient in Algorithms 8 and 11 (line 2). This is useful and more accurate because the channel capacity is shared between TCP (or UDP) and other types of traffic. Along similar reasoning, if the amount of both TCP and UDP traffic are seen to be similar (and in majority) for a particular DSAN, then their cumulative data-rate (i.e., $D_i^{in,tcp} + D_i^{in,udp}$) should be used for comparison with new channel capacity in Algorithms 8 and 11. In a scenario where highly reactive QoS feedback is required, the total incoming data-rate can be used for comparison with the new channel capacity. Further, all CHs should be notified to scale down, irrespective of their individual incoming data-rate.

5.4.5 Limitations

While DSASync's architecture as a non-intrusive network management entity has numerous important benefits like compatibility and ease of deployment, it also leads to certain limitations. We discuss two key limitations with the current design of DSASync.

5.4.5.1 Identifying RTP-based UDP flows

Proactive connection management for the QoS-sensitive RTP-based UDP flows is an important feature of DSASync. However, in practice, DSASync's passive connection identification, by sniffing packets-in-transit, may not be able to identify all the RTP-based UDP flows. Though it is trivial to check for a UDP or TCP packet using the *protocol* or *next header* field of IP header, no such standard mechanism exists for identifying RTP header which is part of application layer payload. RTP packets don't have pre-assigned specific port number and don't have standard signature, which further limits the ability to identify RTP sessions. In our implementation, we use a method similar to that of packet sniffing tool *Ethereal*, which uses packets seen earlier (e.g., SIP or RTSP packets) during the setup of connection to identify the

RTP sessions. We improve this approach by looking for specific port ranges which are typically used by applications for RTP session setup and subsequent communication.

Though the aforementioned approach works well for identifying most RTP-based UDP connections, it can be seen that the identification process is not foolproof and cannot capture all RTP-based UDP flows. Appropriate configuration of DSASync based on domain-specific knowledge of the DSAN can be used to improve the efficiency of this aspect. However, use of encryption can also limit the connection identification process, as discussed next.

5.4.5.2 Connections with encrypted traffic

DSASync, in its current form, cannot be utilized for traffic that is based on encryption below transport layer, e.g., IPSec—which encrypts IP payload including transport/application headers. Since the encryption is end-to-end, DSASync, as a third entity, cannot sniff or classify the packet in transit. Thus, it is unable to recognize and manage such connections comprehensively, though buffering strategy can still be used.

Most applications, however, utilize encryption at higher layers (e.g., TLS/SSL), which has no impact on DSASync's connection identification process, TCP management, and general UDP management schemes. But RTP-based connection management strategy for UDP may not be feasible because application payload is encrypted and hence, RTP headers cannot be identified.

5.5 Implementation

We evaluate DSASync by implementing it as a Linux kernel module. DSASync kernel module is designed to operate with our SU emulator implementation (see Appendix B).

The implementation of DSASync is simplified, as it has been developed with

practical deployment as its key design goal. Since the DSAN is a single homogeneous wireless cell (see Sections 1.2 and 5.2) with nodes operating on the same DSA MAC-PHY protocol, both `DSASync_LL`, as well as, `DSASync_TCP` and `DSASync_UDP` need to execute only at the BS. There are two contributing factors. First, as mentioned earlier, the required parameters are easily accessible from the link-layer module. Second, the BS, in its role as the "manager" of the DSAN, has full knowledge about the network state (including the required parameters of other nodes).[4]

The key challenge faced during the implementation is sniffing of RTP connections transiting through the BS, which is required by `DSASync_UDP`. As discussed in Section 5.4.5, we rely on detection of session setup packets (e.g., SIP/SDP packets) as well as typical port numbers used by VoIP applications to identify RTP-based UDP connections.

sectionEvaluation

5.5.1 Testbed Setup

A testbed is built according to our system model (see Figs. 1.2 and 5.1), and consists of a WLAN cell with 6 client laptops (the SHs), each equipped with an Atheros-based Linksys WPC 55AG wireless card. Another laptop acts as the AP (the BS) which interfaces with the wired LAN of our University. The CH is deployed on the wired segment of the University LAN. Thought both SH and CH are part of the same local network, resulting in lower end-to-end latencies than what is typically experienced on the Internet, this setup is adequate for testing DSASync. An additional laptop, acting as the incumbent transmitter (based on PU emulator implementation described in Appendix B), is placed in the vicinity of the DSAN. The incumbent produces ON/OFF patterns of random durations according to an expo-

[4]In the case where all the required information is unavailable at BS, the `DSASync_LL` component may need to be deployed at the wireless nodes. Control packets can then be used to transmit information to the BS.

nential distribution.[5] The average of ON/OFF duration for the distribution is varied to change the incumbent channel utilization.

We use 802.11a (channel 36) for wireless communication, and *iperf* (version 2.0.4) [90], a commonly used open source network testing tool, is used to generate TCP/UDP traffic for microbenchmark experiments. For macrobenchmarks involving RTP-based UDP connections, we use the open source VoIP application *ekiga* (version 3.2.6) [82]. Ekiga is a feature-rich softphone and supports multiple signaling protocols (like SIP, H.323) and commonly used audio/video codecs. We instrumented the ekiga source code which allows us to exercise fine-grained control over connection parameters, as well as, observe key events and produce statistical information about its ongoing connections. Tcpdump is also used to monitor the traffic and verify statistics.

The default PHY data-rate is set at 24Mbps, while the buffer capacity at the AP is kept at 500MB each for both TCP and UDP. The default average incumbent channel utilization is 20%, and the average sensing overhead for each SU is 5% of the runtime. The initial TCP send and receive window size is 256KB and each experiment run lasts 20s. Other default values are: $\alpha_{min} = 0.5$, $B^{tcp}_{high} = 500$MB, and $B^{tcp}_{low} = 400$MB. Saturation level traffic is used for both TCP and UDP, unless otherwise noted.

5.5.2 Performance Metrics

Application-layer goodput is the fundamental performance metric used to evaluate DSASync. End-to-end delay and jitter are other metrics used for analysis. For each of the experiments, we compare the performance metrics for two cases: (a) DSA operating with DSASync ("DSASync"), (b) DSA operating without DSASync ("Regular").

[5]Results were statistically similar when other types of probability distributions, like uniform or log-normal distributions were used.

5.5.3 Results and Discussion

5.5.3.1 Overhead characterization

To analyze DSASync's run-time overhead, we compare the goodput achieved using unmodified 802.11a with the scenario where DSASync agent is active at the BS. On the basis of 100 experimental runs, the extra overhead with DSASync is found to result in an average of 1.9% reduction in goodput compared to the best case, i.e.,the goodput when there is zero DSA overhead. The overhead on end-to-end delay is found to be very minor (≈1.1ms). However, we observe that gains from using DSASync when DSA is employed (which are discussed next in Section 5.5.3.2) far outweighs its overhead impact. Thus, DSASync must be activated only when the edge DSAN is actively using DSA.

5.5.3.2 Microbenchmarks

To establish the basic performance trends with DSASync, we first evaluate it using a single wireless client in the WLAN cell. Fig. 5.3 shows the average TCP goodput variation in the time-window of 0-20s. UDP traffic is found exhibit a similar pattern, though the absolute values for goodput is higher because of greater efficiency of UDP resulting from its connectionless nature (no retransmissions, congestion backoff, etc).

It is seen that employment of DSASync results in better goodput as compared to regular DSA, especially in the downlink (CH-SH) direction. For this scenario, the average TCP goodput improvement is 74% over regular DSA (see Fig. 5.4). This is a result of DSASync's ability to effectively mask the TFPs (which is 25% of the total runtime) by buffering the incoming packets at the BS and proactively signaling the sender to cease transmission, when necessary (see Algorithms 4 and 5). Thus, unnecessary reduction in the send window at CH is avoided and there is negligible packet loss. Consequently, there is very little retransmission overhead (0.018Mbps), con-

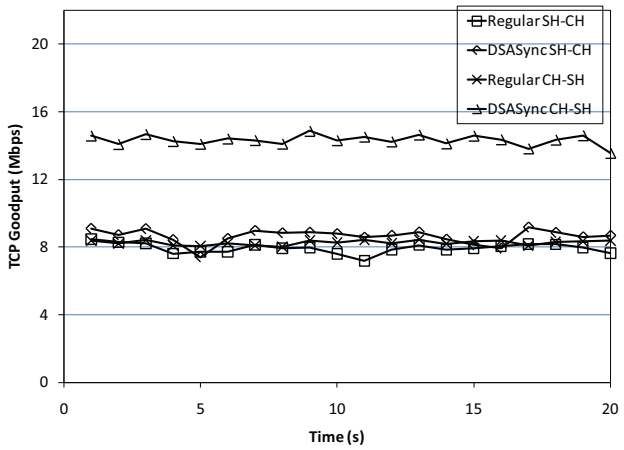


Figure 5.3: Average goodput for TCP, each over last 1s-period, during 0-20s intervals.

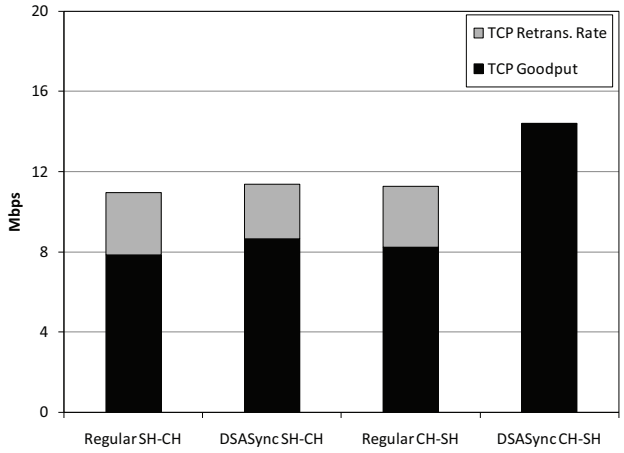


Figure 5.4: Average TCP goodput and retransmission rate.

tributing to a much improved goodput. Through a packet-level analysis in tcpdump, we notice that the downlink (CH-SH) data stream also benefits from the traffic shaping in the uplink (SH-CH) direction. This is because the ACKs are sent to the CH at a lower but steady rate, even during TFPs, which allows CH to continue sending the data packets by advancing its send window.

On the other hand, in absence of DSASync, packets get dropped at the BS during the TFPs. This results in reduction of send window (the sender perceives losses as congestion) and significant retransmission overhead (3.1Mbps). Thus, the goodput is much lower.

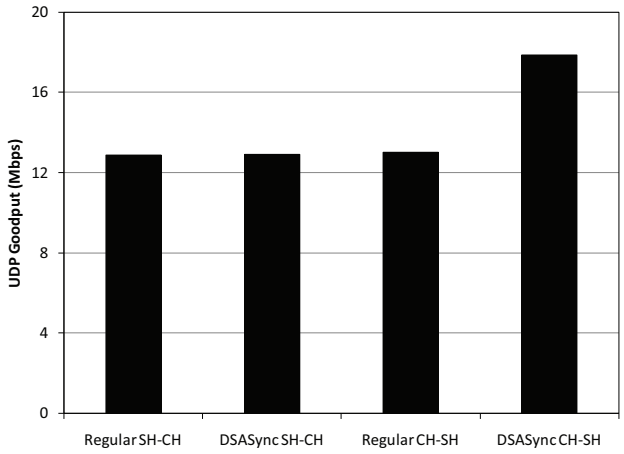


Figure 5.5: Average UDP goodput.

As seen in Fig. 5.4, the gains associated with DSASync for uplink data stream is lower as compared to the downlink direction. Here, the goodput improves by 10% on average. This is because during the TFPs, the data packets originating from the SH side are essentially lost at the SH itself. Thus, the packets don't even reach the BS during interruptions. However, there is still some improvement because the BS shapes the uplink traffic (see Algorithm 7), and also buffers the inbound ACKs for SH.

Similar observations are made for UDP connections, where the average goodput comparison is shown in Fig. 5.5. Again, the improvement is much higher in the downlink direction (about 38%) as compared to the uplink direction due to reasons mentioned above. Since there is no extra burden of retransmissions (even if packets are lost) in UDP, the absolute percentage improvement is lower. Newer packets are continued to be transmitted and contribute to UDP goodput. Note that we are not using RTP-based UDP flows for these microbenchmark experiments to eliminate the application-dependent behavior in these results. Thus, only generic and always-guaranteed UDP management benefits are visible here. Depending on how the higher layer application chooses to respond to RTCP control messages, the advantages of DSASync can be greater, as we evaluate in Section 5.5.3.3.

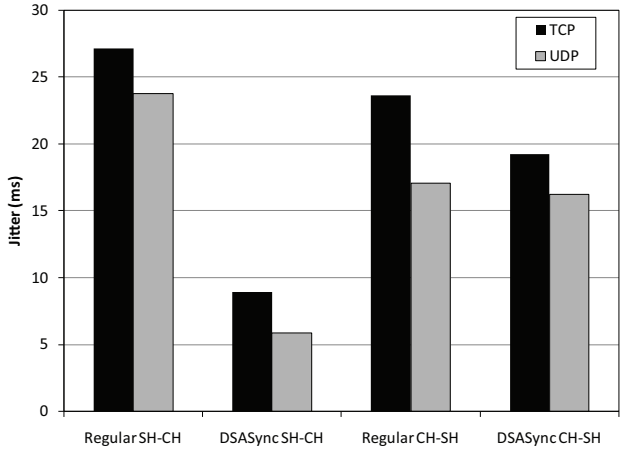


Figure 5.6: Average end-to-end jitter at the receiver.

An interesting trend is seen with the delay variation, which is shown in Fig. 5.6. Variation in delay (at the receiver's end) is a direct indicator of the level of jitter at the application level, which is an important QoS metric. Deployment of DSASync produces a significant reduction in the average jitter at the receiving CH for the uplink traffic, for both TCP and UDP. This is, again, as a result of managing the uplink traffic at the BS. Note that the jitter for downlink data stream remains high, despite substantial improvement in corresponding goodput. This is expected, because the SH cannot receive any data packet during TFPs, even though the BS buffers them for it. Also, note that jitter performance for RTP-based UDP connections (not used in this microbenchmark experiment) can be even better, depending on the application-specific behavior.

These observations suggest that deploying a local DSASync agent at each WLAN node would help in reducing the CH-SH delay variation while also improving SH-CH goodput. Our preliminary results with per-node DSASync agent indicates the validity of above conclusion. However, there are also some drawbacks associated with distributed DSASync model, as discussed earlier in Section 5.4.4.2. We treat this as an optional extension. Note that a DSASync agent at the BS will still be required in the distributed DSASync architecture.

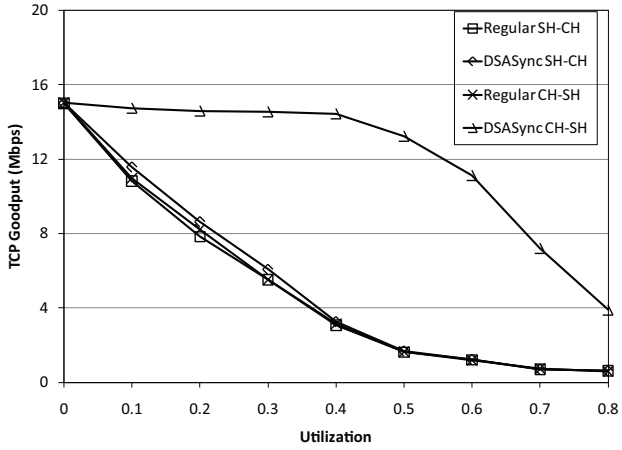


Figure 5.7: TCP goodput with varying amount of DSA disruptions.

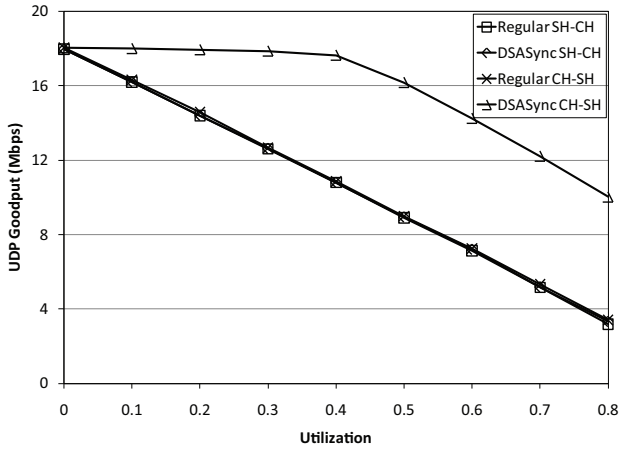


Figure 5.8: UDP goodput with varying amount of DSA disruptions.

Figs. 5.7 and 5.8 show the goodput variation with changes in the magnitude of DSA-related interruptions. The DSA impact is represented by "utilization", which includes sensing overhead and incumbent activity. As expected, the goodput decreases when the DSA behavior becomes more aggressive. However, we note that with DSASync CH-SH goodput improvement is even better at higher utilizations. The goodput drops significantly only when the utilization factor is greater than 0.5. Note that DSA is not suitable for channels that exhibit very high incumbent utilization. Thus, a good DSA MAC protocol would not select such channels anyway (or would switch away from such channels). DSASync leads to marginally better perfor-

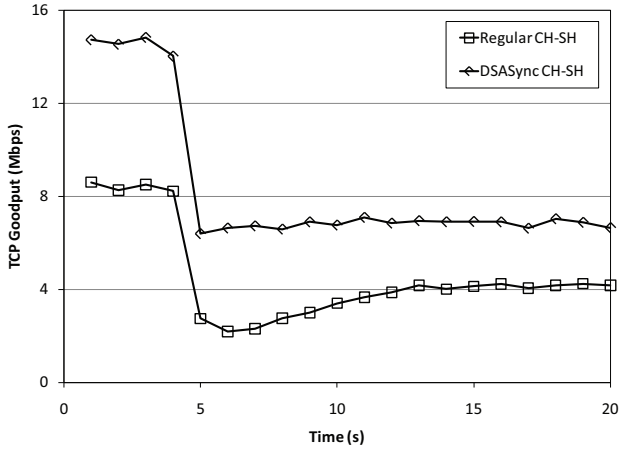


Figure 5.9: Effect of PHY capacity change on TCP connection.

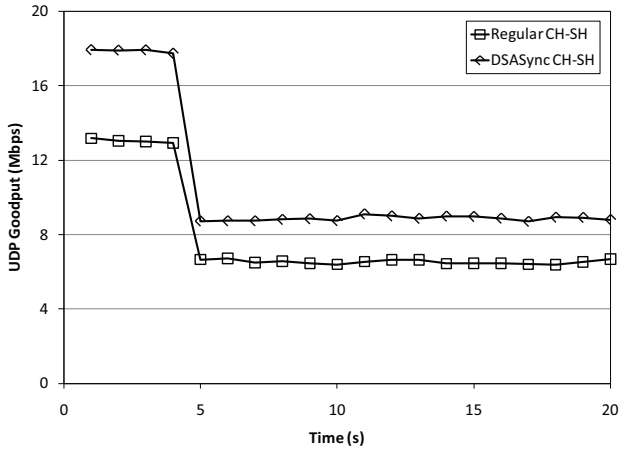


Figure 5.10: Effect of PHY capacity change on UDP flow.

mance ($\approx 10\%$) for the uplink data stream. However, the performance drops quickly with increase in utilization, which again highlights the usefulness of a local DSASync agent at each WLAN node. Our experiments also reveal that larger buffer space at the BS improves the resilience provided by DSASync, especially for UDP flows.

Fig. 5.9 shows the effect of reducing the network capacity, which can occur when the DSAN changes channels. Here the PHY-layer capacity is reduced to 12Mbps from 24Mbps at 5s. As seen in the plot, without DSASync, the CH-SH goodput reduces by almost 70% (the capacity reduction is 50%) and takes some time (6-7s) to recover. However, with DSASync there is no perceptible extra reduction in throughput beyond

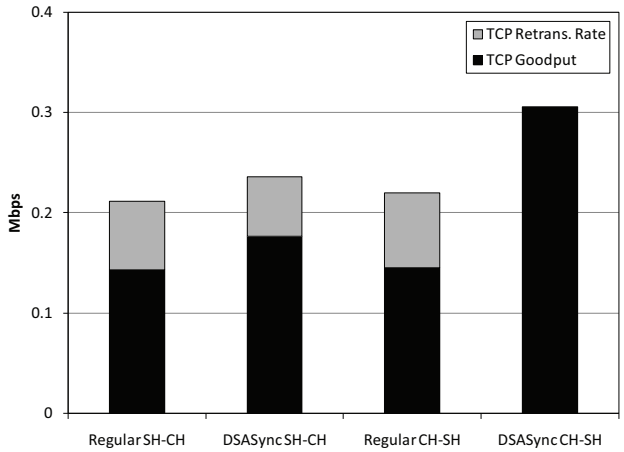


Figure 5.11: Average goodput across multiple TCP connections.

the expected decrease. This is attributed to proactive sender notification through Algorithm 8.

On the other hand, there is no appreciable difference in behavior for UDP connections between *Regular* and *DSASync* case, as seen from Fig. 5.10. This is because the UDP goodput metric is decreased by equal amount in both cases (note that here losses and retransmissions don't matter), although, the buffering mechanism and traffic shaping contribute to higher goodput values when DSASync is used. Again, the RTCP-based proactive feedback mechanism is not active in this microbenchmark experiment. If RTP-based UDP connections are present, such feedback (Algorithm 11) may possibly lead to change in UDP connection behavior depending on application's reaction, e.g., change to a low-bandwidth codec, etc. We observe this phenomena with ekiga during our macrobenchmark experiments discussed next.

5.5.3.3 Macrobenchmarks

To check the scalability of DSASync, 4 TCP and 4 UDP connections are started on each of the 6 clients—thus, there are 48 parallel ongoing connections. Fig. 5.11 shows the average performance experienced by TCP connections in terms of goodput and retransmission rate. Fig. 5.12 shows the performance for UDP connections. The

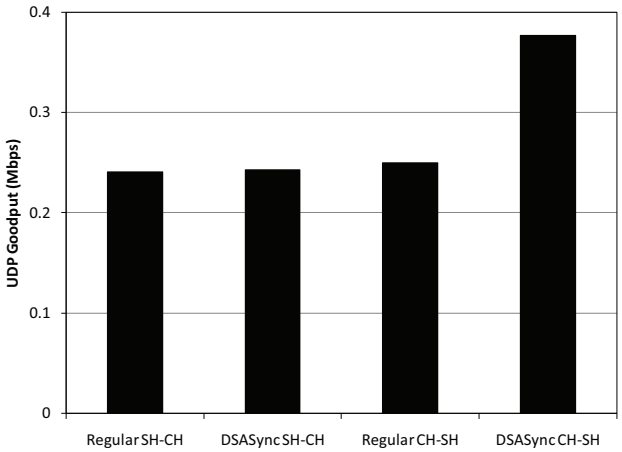


Figure 5.12: Average goodput across multiple UDP connections.

trends are similar to those noted in Figs. 5.4 and 5.5. DSASync is found to perform even better in a larger-scale situation, especially in the downlink CH-SH direction where goodput improves by about 102% for TCP and 51% for UDP. Similar results, as those noted for microbenchmarks, are observed for other corresponding experiments.

To see the benefits of the DSASync in action, especially for the QoS-sensitive RTP-based UDP traffic, we use ekiga [82] softphone to generate videoconferencing sessions using G.711 (audio) and H.261 (video) codecs with the call speed at 384kbps. This requires an actual link bandwidth of around 460kbps each way with low jitter for optimum performance. We randomly create between 5-15 sessions in each experiment run, which are distributed among 6 wireless nodes in the DSAN, with each node communicating with a fixed host in the university network.

Fig. 5.13 shows the average goodput achieved for the communication sessions, while Fig. 5.14 shows the average jitter encountered. Deploying DSASync enables better overall bandwidth and significantly lower jitter for the videoconferencing session, which confirms our end-user experience during the active session. Both audio and video quality were found to be perceptibly better when DSASync was active. However, the advantage is skewed towards the downlink direction, and DSASync can achieve very close to the required bandwidth (460kbps) despite DSA disruptions.

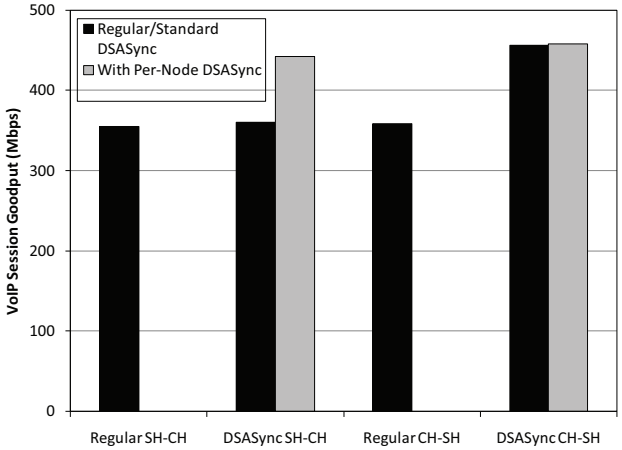


Figure 5.13: Average connection goodput for ekiga VoIP sessions.

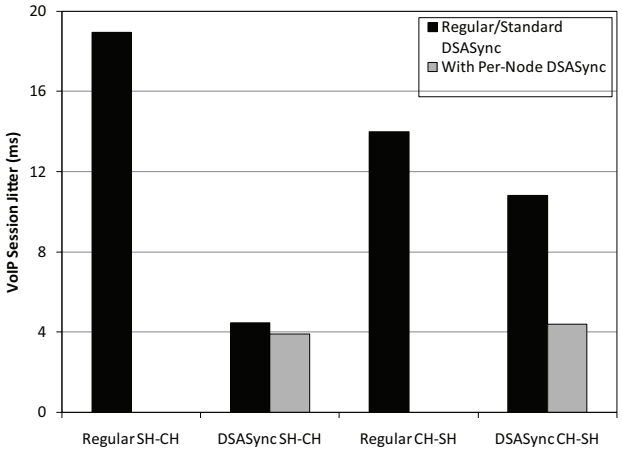


Figure 5.14: Average jitter experience by ekiga VoIP sessions.

The graphs also shows the benefits of deploying a per-node DSASync agent. For this purpose, we implemented and deployed a local DSASync agent at each of the wireless node to manage the ingress/egress traffic at the node itself. The goodput for uplink traffic is found to improve, while also reducing jitter substantially for downlink traffic. Thus, BS based central DSASync agent together with local DSASync agents seem to be a complete solution to manage DSA-related disruptions on end-to-end connections.

5.5.3.4 Remarks

Many other results have been omitted in interest of brevity and space. But we mention some important observations below.

The results obtained from the experiment runs are found to be quite consistent. The standard deviation of the performance metrics is low, and varies between 0.8%-2.3% of the average value. Goodput was found to show lower variation than delay and jitter parameters.

Our experiments indicate that the scalability of DSASync depends on the amount of resources, especially buffer-space, at the BS. Modern base stations or access points are increasingly getting powerful and memory is getting cheaper, so it is not a significant constraint. Large buffers will still be very important for supporting large DSANs, and would constitute a part of network planning. To optimize this aspect, the buffer size at the BS can be established based on the expected amount of DSA overhead as well as traffic characteristics. A simple dynamic buffer allocation scheme can also be applied for this task.

Further, as seen from the results, there is a good case for deploying local per-node DSASync agents, in addition to the proxy-type DSASync agent at the BS. However, we consider it to be an optional extension of DSASync because it is difficult to ensure that all the wireless clients of a DSAN (like a wireless hotspot) implement this feature. Further, limited resources on many mobile devices (e.g., cellphones) may prevent its deployment. For consumer-oriented WLANs, the majority traffic is inbound (downlink) due to dominance of downloads. Large enterprise Wifi vendors report that downlink traffic constitutes 80% of total wireless traffic [91]. The observation from WLAN traffic traces of campuses and offices are similar [92, 93]. Thus, standard DSASync platform will still be sufficient for most deployment scenarios.

In conclusion, we argue that with the trend of increasing computing power and memory availability at low cost, the extra space/computation overhead associated in

running DSASync is insignificant, especially considering the impressive performance gains achieved in this process. More importantly, DSASync accomplishes this with 100% compatibility to existing protocols. Thus, DSASync promises to be an effective network management tool to improve end-to-end connection performance in edge DSANs.

5.6 Concluding Remarks

We identified the important end-to-end communication performance issues when an edge WLAN features DSA capability. In this context, we studied the impact of DSA-related disruptions on TCP/UDP connections to/from the wired cloud. To address the identified problems, we have proposed a novel network management framework called *DSASync*. DSASync primarily comprises an agent on the wired–wireless interface node (e.g., the base station) of the WLAN, which executes algorithms based on buffering and traffic-shaping to minimize the adverse effect on ongoing connections. DSASync features compatibility and ease of deployment as its chief design goals. Consequently, DSASync requires no changes to the TCP/UDP protocols or their existing implementations, and maintains the end-to-end semantics. We evaluated DSASync in a testbed based on our prototype implementation for Linux kernel. The testbed consists of an edge DSA-based WLAN interfaced with our University's wired network. The evaluation results indicate that DSASync makes a significant improvement of performance for end-to-end connections, e.g., the downlink goodput increases by 74% for TCP and 38% for UDP in a single connection environment, with even greater gains (102% for TCP and 51% for UDP) when multiple connections are active in the DSAN. Other QoS metrics are also found to improve significantly, e.g., jitter is reduced by more than 75% for VoIP sessions. Further, DSASync shows resilience in maintaining good end-to-end connection performance with increase in DSA-related disruptions.

CHAPTER VI

Energy Cost Analysis and Management of Software Defined Radios in DSA

6.1 Introduction

A critical requirement for DSA is highly tunable antenna together with a dynamically reconfigurable radio. Software Defined Radio (SDR) [24, 49] is, therefore, considered to be the platform that makes DSA feasible. In contrast to traditional radios (e.g., ASIC or analog circuitry), SDRs consist of minimal hardware (basic radio frontend), with the actual signal and protocol processing performed through software instructions executed on a *General Purpose Processor* (GPP). The basic idea of a SDR is to move software as close to antenna as possible. There are several advantages associated with SDRs, including reconfigurability, updatability and low cost.

We make the following two observations that motivate this work.

O1. *Signal processing in commodity software for GPP is a highly computation-intensive process.* High volume of computation invariably leads to higher energy consumption.

O2. *DSA—the foremost upcoming technology to utilize SDRs, features operations that add to its computational requirement.*

Though SDRs require extensive processing (as noted in O1), the computing power of modern multi-core processors with sufficient resources can sustain such high throughput computations. The necessary computing power is increasingly becoming affordable and small enough to be used in mobile computing devices. Recent research [50,51] have shown that with proper OS scheduling, process priority assignment, and memory reservation, modern MAC-PHY protocols like IEEE 802.11 can run as commodity software.

Unlike progress on the performance front, there hasn't been much research on the energy overhead associated with SDRs. It is known that under heavy computation loads, such as software-based signal/protocol processing, energy consumption increases for GPP. However, the amount of energy overhead has not been quantified. Further, the relative energy impact of different MAC-PHY elements on the SDR platform is unknown, which can be crucial information for reducing energy consumption. Note that due to the very nature of GPP, it is difficult to optimize it for energy consumption. The GPP must be able to support a broad range of instruction types to support general purpose computation.

To elaborate on our observation O2, DSA involves functions like channel switching and spectrum sensing. Changing of the channel (and the associated DSA MAC-PHY) can lead to additional computation overhead. This is especially the case when spectrum-width of the channel increases. Considerable computation overhead can be involved in sensing, especially for feature detection. To the best of our knowledge, no DSA protocol proposed thus far uses energy consumption as an important design parameter

Therefore, motivated with aforementioned observations, we establish the energy profile of using SDRs. We quantify the increase in energy consumption for fundamental wireless communication elements using a SDR, and also investigate the main factors that contribute to this overhead.

Based on SDR's energy profile, we propose the *Dynamic Energy Management in DSA* (DEMD), with the objective of minimizing its energy footprint. DEMD features adaptive PHY chain design to balance the application traffic-demand with minimal energy overhead. Additionally, DEMD ensures that energy-intensive DSA functions are invoked only when needed.

Organization: Rest of the chapter is organized as follows. The testbed setup and methodology for energy measurements are described in Section 6.3. The energy overhead analysis of SDRs is presented in Section 6.4. The need for energy awareness in wireless PHY design for SDRs is discussed in Section 6.5. DEMD and its evaluation are described in Sections 6.6 and 6.7, respectively. The chapter concludes with Section 6.8.

6.2 Definitions

We will use the following terms throughout this chapter.

- *Software Defined Radio* (SDR): A wireless communication radio consisting of (1) a software component (executed on host machine) that describes waveform processing, and (2) a hardware component to send and receive signal.

- *Wireless PHY Chain* (WPC): The signal processing blocks that process the incoming/outgoing waveform in a specific order. For SDRs, WPC is implemented in software.

- *Software Radio Frontend* (SRF): The radio control board (RCB) together with the radio frontend (RF) component of a SDR. SRF provides the necessary hardware capability to communicate on a wireless channel. E.g., USRP2 RCB [27] fitted with a daughterboard is the SRF in our testbed.

- *System Under Test* (SUT): A mobile/wireless device equipped with a SDR.

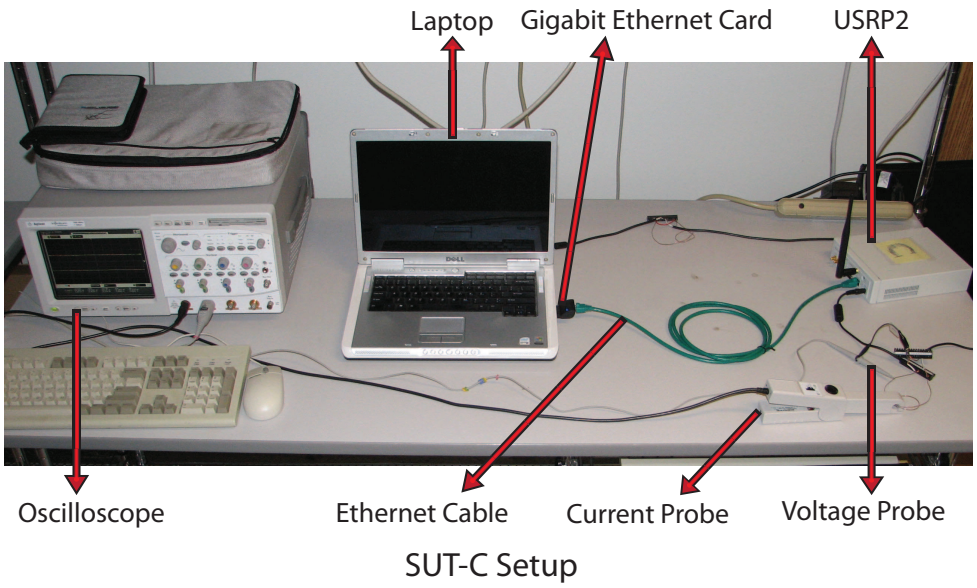


Figure 6.1: The testbed setup showing one SUT along with measurement apparatus.

- *Base Power* (BP): The average power consumed by the SUT (under specified standard configuration), when there is no radio processing load.

6.3 Testbed Setup

We built an energy measurement testbed to study the energy usage characteristics associated with the SDRs. The testbed consists of multiple SUTs. Each SUT comprises a laptop together with USRP2 SRF [27]. Fig. 6.1 shows a SUT setup in our testbed. We next provide details on the standard configuration used for the main components of a SUT.

6.3.1 USRP2

Universal Software Radio Peripheral (USRP) is an experimental SRF, that is widely used in software radio research and also in some production systems [94]. Besides being a basic RF (antenna + oscillator + ADC/DAC), it also provides additional features like downconversion of sampled signals and buffering. We use the second generation of USRP, called USRP2 [27], which features gigabit ethernet con-

Table 6.1: USRP2 Components and Configuration

Component	Detail
Motherboard	FPGA (Xilinix XC3S2000-5FGG456)
Daughterboard	XCVR2450
Antenna	VERT2450
Frequency Band	UNII-1 (Channels 44,48)
Transmit Power	50mW
Power Adapter	6V, 3.5A
Ethernet Cable	CAT 6

nection to the host PC for high bandwidth transfer of radio samples (25MHz @16bits maximum).

The actual analog RF is provided through the *daughterboards*. We use XCVR2450 daughterboards with the USRP2 that has a frequency range of 2.4-2.5GHz and the 4.9-5.9GHz. All the functions of USRP2 + XCVR2450 are controllable from software, which is GNU Radio in our case. Further details on our USRP2 configuration are given in Table 6.3.1.

6.3.2 GNU Radio

GNU Radio [95] provides the toolkit for building software radios. It provides excellent interfacing with the USRP2 SRF. It provides a library of signaling processing blocks which can be utilized to define and process a "waveform". The waveform processing is accomplished by creating a flow graph, where nodes represent the signal processing block and the data (signal samples) flow along its edges. Each block can have input and output ports, and is implemented in C++. The radio flow graph is constructed in Python.

6.3.3 Laptop

Dell Inspiron 6400 featuring dual-core Intel Centrino Duo processor is the host PC component of our SDR platform. Since, it is a general purpose machine, we

Table 6.2: Laptop Components and Configuration

Component	**Detail**
Processor	Intel Centrino Duo T2400 @1.83GHz
Bus/Cache	667MHz FSB, 2MB Cache
Graphics	Integrated Intel MA 950
Memory	DDR2 1GB SDRAM
Hard Drive	80GB @5400rpm
Optical Drive	8x DVD/CD Burner (*Off*)
Battery	6-cell Li-ion (*Removed*)
Power Adapter	Dell PA-12 (19.5V, 3.34A)
Display	15.4" LCD WXGA (*Off*)
Wireless Card	Built-in Centrino 3945 (*Off*)
Ethernet Card	Belkin Gigabit ExpressCard
OS	Ubuntu 9.10 (kernel 2.6.31-17-generic)

controlled its configuration to accurately quantify the energy consumption overhead due to SDR-related processing, e.g., the LCD screen and the in-built wireless card is turned off during experiments. More details on our laptop configuration are given in Table 6.3.3.

6.3.4 Energy Measurement Methodology

Our primary tool for energy measurement is the High precision Agilent Infiniium Oscilloscope 54815A [96]. We use a sampling rate of 1kSa/s to accurately monitor the voltage and current in the circuit being probed. The oscilloscope readings for a measurement session are analyzed for statistical data like average and standard deviation. Out of its 4 channels, two are utilized—one for voltage measurement (V) (using HP 1160A probe) and the other for current measurement (I) (using HP 1146A probe). The power consumption of the circuit is given by $P = V \times I$. Note that this formula does not have A.C. component as the voltage is constant D.C. (the output from the adapters of the laptop and the USRP2 SRF). Thus, the power consumption depends directly on the amount of current drawn by the SUT. The probes together with the oscilloscope have high resolution (1μV) with error margin $< \pm 4\%$.

6.4 Energy Impact of SDR

We performed several experiments to observe the energy overhead involved in running fundamental wireless PHY algorithms on the SDR platform. The experiments were conducted for 3 SUT units and repeated 20 times in an experiment run. We conducted 6 experiment runs for every test scenario, which provided us with a total of 360 data points for each scenario. In rest of the chapter, we use the term "power" and "energy" interchangeably, as power (which is easier to measure) represents the time-average of energy consumed.

6.4.1 BP Determination

First, we determine the average BP consumed by SUTs. Each SUT is powered on but no communication is done using the SDR. The laptop also has only basic operating system components active, with no applications running. The active system processes include the kernel processes together with gnome display process (Xorg). The total CPU utilization in this setup is found to be less than 4%. The average power consumption of the laptop is found to be 19.03W while that of the USRP2 board is 9.11W. Thus, the BP of the SUT turns out to be 28.14±0.41W (stddev.), as shown in Fig. 6.2. Overall, the laptop's power consumption (19.03W) is the dominant factor in the BP, as clearly visible in Fig. 6.2.

6.4.2 Signal Acquire and Transmit

Next, we analyze the energy cost of acquiring and transmitting waveforms in a SDR—no actual signal processing is done. In digital communication, the raw signal is captured by the RF, and converted to digital samples through the ADCs for further processing. In SDRs, this is accomplished through the SRF—the USRP2 RCB + daughterboard in our SUT. The digital samples are then passed on to the host machine. The transmission of signals involves the reverse procedure. Digital samples

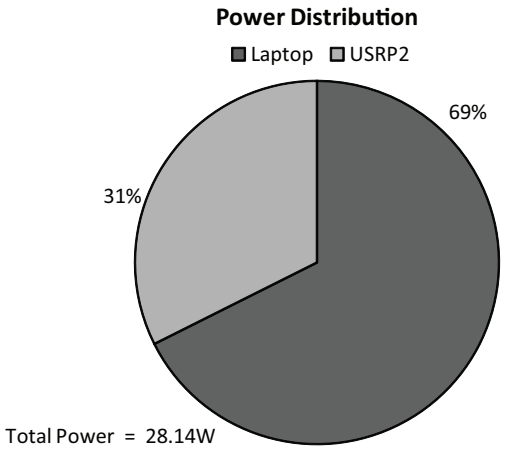


Figure 6.2: BP consumption.

created at the host are sent to the SRF, which is then converted to analog form using the DACs and transmitted over air.

The energy overhead in this process is due to the number of samples generated and transferred to/from host machine. In turn, the number of digital samples also affects the processing and energy cost of signal processing activities. The number of samples depends on the sampling rate of the ADCs/DACs, which is usually fixed. However, the amount of generated samples can be controlled by appropriately *decimating* or *interpolating* the sampling rate. If the frequency-width (or more accurately, the Intermediate Frequency (IF) range) is less than half of the decimated sampling rate, there is no significant loss of information (Nyquist Sampling Theorem). Most RFs (including the USRP2) provide this feature on-board.

Using appropriate decimation/interpolation factors, we measured the increase in power consumed (over BP) when digital samples are acquired or transmitted. Figs. 6.3 and Fig. 6.4 show the average percentage increase in power from BP level (stddev.$<$1.6% for all cases). It is seen that frequency-width has a significant impact on energy overhead. There is a steady increase in energy cost with increase in channel-width.[1] For the highest case (12.5MHz wide channel), the average power con-

[1]The channel-width values shown in the graph are due to decimation factor being a power of 2.

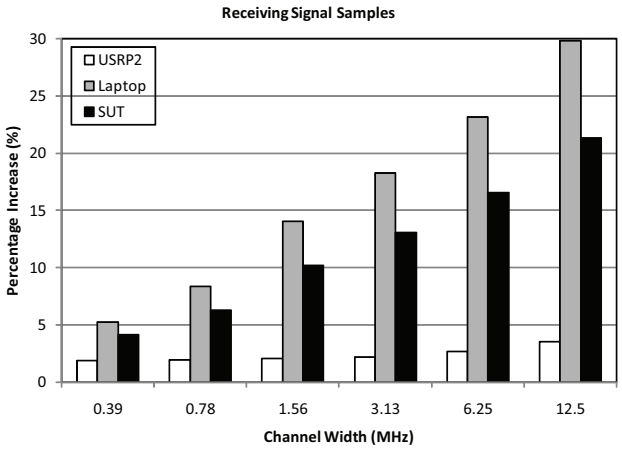


Figure 6.3: Receiving signal samples.

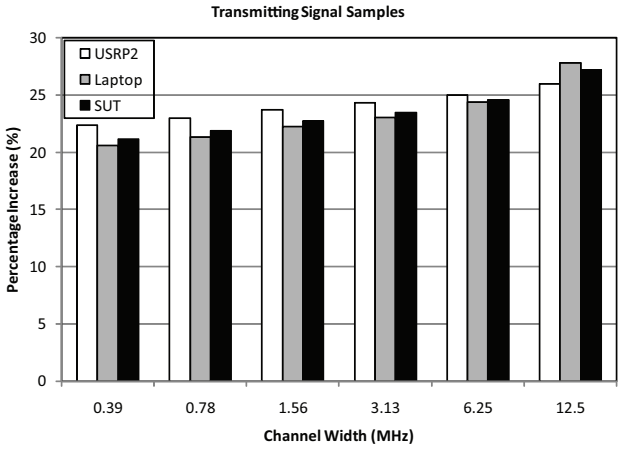


Figure 6.4: Transmitting signal samples.

sumption for receiving samples is 34.14W—an increase of 21.3% over BP. Similarly for transmitting samples in this case, the power is found to be 35.79W—an increase of 27.2% from BP.

For reception, the power consumption increase is mostly at the laptop (about 29%) due mainly to the reception of the sample stream (via PCI-express interface for the gigabit ethernet). On the other hand, for transmit case, the power consumption increase is also noticeable at the USRP2 (about 25.9%). This is expected, as the SRF consumes more energy to transmit on-air than to receive.

6.4.3 Core Signal Analysis

For useful digital communication to happen, the received digital samples must be analyzed to extract channel characteristics. Here, we evaluate the energy overhead involved in two basic signal processing tasks: (a) detecting the signal/noise strength, and (b) checking the frequency components of the signal. Objective (a) is achieved typically by computing the magnitude of the incoming stream of digital signal samples, while objective (b) is accomplished through FFT calculation.

Using GNU Radio, we wrote a block to calculate the average squared magnitude of the incoming digital samples (simple carrier sense). Fig. 6.5 shows the average percentage increase in energy consumption over BP. Again, we notice that the increase in frequency width leads to increase in energy overhead. For 12.5MHz wide channels, the power consumption is 34.5W—an increase of 22.6% over BP. Also, we observe that the percentage increase in power usage here is only slightly more for receiving raw digital samples (shown in Fig. 6.3). This shows that the computation involved (average magnitude squared) is not a significant energy overhead factor.

To analyze frequency components in the signal, we ran the FFT calculation of received samples for different frequency-widths, as shown in Fig. 6.6. GNU Radio comes with built-in FFT block for optimized computation. Still, FFT leads to significantly high power consumption, especially for wide channels. The average power consumption of the SUT was found to be 44.41W—an increase of 57.9% over BP for 12.5MHz channels. Most of this increase is contributed by the laptop, which consumes 84% more power than base scenario, as evident from Fig. 6.6.

The processing overhead involved in FFT computation depends on its *bin size* parameter. Fig. 6.7 shows that with increase in bin size, the energy consumption is higher.

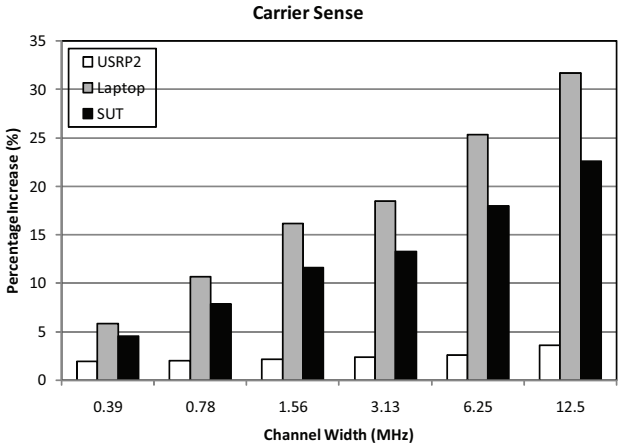


Figure 6.5: Carrier sensing.

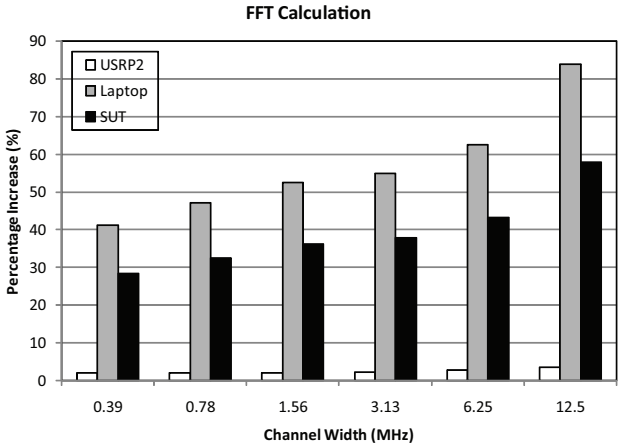


Figure 6.6: FFT Computation.

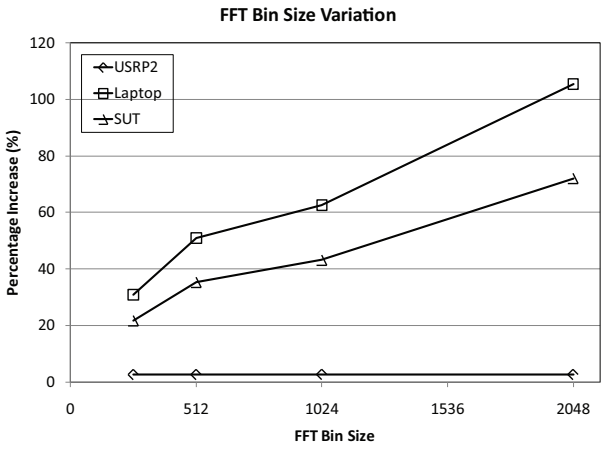


Figure 6.7: Changing the FFT bin size.

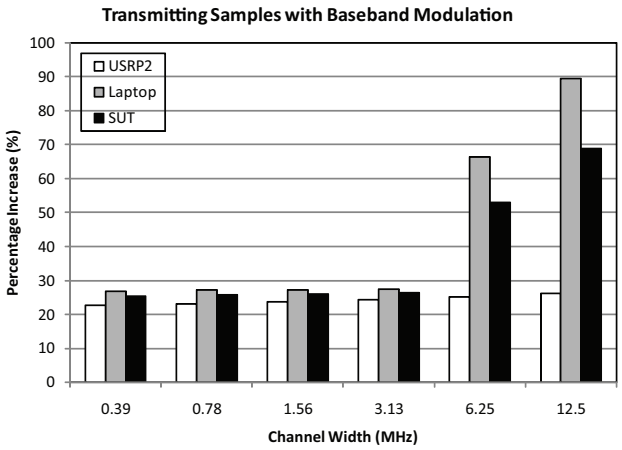


Figure 6.8: Analog modulation.

6.4.4 Analog Signal Modulation

Modulation is a fundamental part of wireless communication, through which the carrier signal is modulated in accordance with the samples to be transmitted. We conducted experiments to quantify the energy consumption when simple analog modulation schemes are executed. Fig. 6.8 shows the increase in power consumption involved in transmitting samples after modulating a sine wave carrier. The overall power increase for 12.5MHz bands is 68.9%. For this case, the increase in laptop's power usage is 89.5%.

6.4.5 Frame Send/Receive

We next measured the energy involved in sending and receiving frames. The receive PHY chain comprises a combination of the basic signal processing activities analyzed in previous subsections—receiving digital samples, carrier sensing, frame detection, and demodulation. The transmit PHY chain comprises the analogous signal processing blocks—modulation, and transmission of digital samples. However, instead of analog modulation (discussed in Section 6.4.4), digital modulation algorithms are used, as typical of modern wireless communication. E.g., DBPSK and DQPSK are used in 802.11b and ZigBee PHY, while GMSK is used for GSM. We use a slightly

modified versions of the built-in GNU Radio library applications `benchmark_tx.py` and `benchmark_rx.py` to conduct this test.

Fig. 6.9 shows the variation of energy consumption for receiving packets with commonly used modulation schemes. We observe that the overall increase in power usage ranges between 76-94% depending on the modulation scheme. D8PSK shows the greatest energy overhead while GMSK shows the least. Almost all of the increase in power consumption is at the host machine—the power consumption of the laptop is more than doubled (an increase of roughly 123%). This clearly shows that software-based digital signal processing on the GPP for general purpose wireless communication produces unsustainable increase in energy consumption.

The energy overhead of transmitting packet is shown in Fig. 6.10. The power consumption increase ranges between 47-54%, depending on the modulation scheme. The USRP2 SRF's contribution is higher (about 26% increase in its own power usage) for the transmit case in comparison to the receive scenario, as expected. However, contrary to our initial expectations, we see that the power consumption for transmitting frames (about 42W) is lower than that for receiving them (about 51W). This is attributed to two main factors: (a) higher computing power is required for demodulation, (b) the receiver chain has to continuously accept and process samples because a packet may arrive at any time. Both of these factors result in a higher GPP utilization while receiving (as also noted in [50]), which leads to its higher energy consumption shown in Fig. 6.10.

6.4.6 Other Experiments

We conducted a subset of previous tests with certain other modern GPP architectures (e.g., Pentium M T4200, Core 2 Duo P8600 mobile processor). We found that despite slightly different BPs (due to different laptop characteristics), the percentage increase in energy consumption is very similar to that used for our SUT

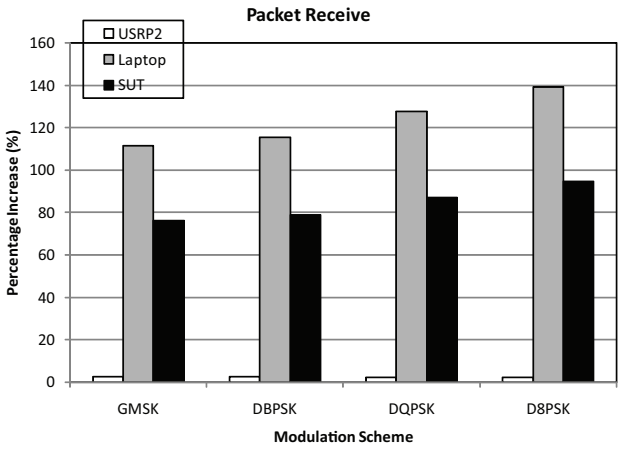


Figure 6.9: Receiving packet, channel-width = 6.25MHz, bitrate = 0.2Mbps.

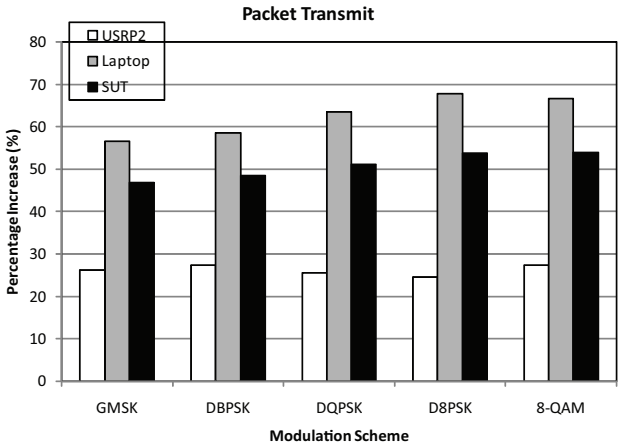


Figure 6.10: Transmitting packet, channel-width = 6.25MHz, bitrate = 0.2Mbps.

(which consists of Core Duo T2400 processor based laptop). Next, we discuss the key observations and trends from the experiment results.

6.4.7 Discussion

Based on the previous experiment results, it is clear that SDRs consume significant amount of energy. We claim that the energy usage increase for modern wireless PHY chains based on SDRs is going to be greater than what we report here. There are three main reasons for this claim.

1) Though GNU Radio + USRP2 SRF provides a flexible and powerful SDR

platform, it still consists of functions that are implemented in hardware to reduce computational load on the GPP. For example, the Digital Down Converter (DDC) and the Digital Up Converter (DUC) are implemented using Verilog and run on USRP2 RCB itself. Thus, it is reasonable to expect that for more *ideal* SDRs, the energy consumption is going to be higher than those presented here.

2) In our experiments, we tested a core subset of the fundamental wireless PHY algorithms. However, we left out certain signal processing blocks used in modern wireless PHYs that require greater computational throughput than that available in our SUT, e.g., Viterbi encoding/decoding scheme (used in 802.11a/g PHY). Thus, for such wireless PHYs such higher computation cost will translate into further increase of energy consumption.

3) Again, our test scenarios were limited to supporting communication upto .5Mbps and with channel-width upto 12.5MHz. Many wireless PHYs support much higher bitrates and channel-widths. For example, IEEE 802.11a wireless PHY operates on 20MHz wide channel and supports 54Mbps bitrate. While the newer GPP architectures can now support such expensive radio processing (as shown in [50]), the energy consumption is likely to be considerably greater than those obtained here for our simpler test cases. Also, in a complete wireless system, MAC layer protocol processing must also be executed on the GPP—which will further add to the energy cost of using SDRs.

In view of the aforementioned points, we conclude that the energy consumption overhead for SDR-based wireless communication, as shown in our experiments (Figs. 6.3–6.10), is towards an optimistic lower bound.

6.5 Case for Energy Aware DSA

Based on the energy profile of SDR usage, as discussed in Section 6.4, it is clear that the energy cost of software-based wireless communication is a major bottleneck

in its deployment on realistic systems. In a basic packet receive and send the energy consumption increases by around 80%, and is easily the most energy consuming process on the system. With more sophisticated schemes, higher bitrate and wider channels, the signal processing on GPP architectures more than doubles its energy consumption.

Despite sufficient computing power to support software signal processing (as shown in [50, 51]), the underlying energy overhead will adversely impact upcoming wireless technologies that fundamentally rely on SDRs (most prominent of which is DSA). Thus, we argue that next-generation wireless technologies, should be aware of their underlying execution platform. If the platform is a SDR, they must optimize their functions accordingly. They must account for energy constraint as an important factor in their design. This is especially significant in view of an increasing push towards energy efficiency in electronics and communication devices.

In rest of this chapter, we specifically focus on DSA. The question we intend to answer is: *How to make DSA energy-aware, such that its energy usage footprint is acceptable?* Energy consumption has not been a part of research efforts in DSA. Our goal is to provide design features in DSA that can make it energy-efficient, in view of the energy profile of the underlying SDR platform.

6.6 Dynamic Energy Management in DSA (DEMD)

Dynamic Energy Management in DSA or DEMD is our proposed solution to reduce the energy footprint of DSA. DEMD achieves its objective by incorporating several *energy control knobs* that impact the energy consumption for the underlying SDRs. The key assumption here is that the DSA protocol[2] executes on a SDR platform to ensure dynamic unlicensed operations. However, this is not an arbitrary assumption. As discussed earlier, SDRs are the enabling platform for DSA.

[2]No specific DSA protocol is assumed for this work.

6.6.1 Energy Control Knobs

The following control knobs are based on the energy-usage profile and trends observed for SDRs, which were presented in Section 6.4. These also serve as a general guideline for features that should be exploited when building any energy-aware protocol over the SDR platform.

Knob 1: Wireless PHY Chain. The primary benefit of DSA is the ability to dynamically update the wireless PHY chain. However, different PHY algorithms have widely varying computational/energy overhead. For example, Viterbi algorithm is highly computation-intensive as compared to other channel encoding/decoding algorithms. Thus, a DSA protocol must be aware of the potential overhead of different wireless PHY chains, and must choose a low overhead chain.

Knob 2: Bit-rate. Again, as discussed in Section 6.4.7, higher bitrates on a SDR result in higher energy consumption. Thus, the DSA protocol should attempt to utilize lower bitrates at the physical layer. For instance, even if the DSA MAC-PHY can support high bitrates like 54Mbps, most of the modern communication applications (e.g., VoIP) can be easily supported with lower PHY bitrates like 11Mbps. This guideline, in most cases, amounts to downgrading the modulation scheme in the wireless PHY chain.

Knob 3: Frequency-Width. Wider channels contribute significantly to the processing and energy load increase on SDRs (Section 6.4). Thus, a DSA protocol must try to utilize narrower spectrum-width channels or spectrum regions for unlicensed operations. For instance, TV channels are 6MHz wide as opposed to 802.11 channels that are 20+MHz wide. Even if the particular DSA protocol features multiple physical channels use at one time or dynamic increase of spectrum-width (as proposed in [17]), the actual frequency-width for communication must be chosen conservatively.

Clearly, the proposed DEMD energy control knobs present an important tradeoff

for DSA protocol—*performance vs. energy*. Exercising these knobs for reducing energy consumption can lower wireless link/network capacity and performance, which may not be acceptable. Therefore, these energy control knobs must be exercised intelligently to ensure that lowering energy consumption does not incur unacceptable wireless communication performance.

6.6.2 DEMD Algorithm

In order to dynamically adapt the energy consumption for DSA, we develop the *DEMD Algorithm*, which is a greedy control algorithm. The intuition behind DEMD Algorithm (Algorithm 12) is not to be performance-greedy even if the opportunity exists. Because, increase in performance for SDR platform comes with rapid increase in energy depletion. It evaluates the suitability of applying the proposed energy control knobs in managing the energy usage of the device in context of the traffic demand.

For brevity, we describe the DEMD Algorithm here for infrastructure wireless networks (see Section 1.2). However, it can be extended to other networking models (including standalone wireless devices) with minimal changes.

Algorithm 12 executes periodically (with interval ΔT) on the *master* node (e.g., the access point in a Wifi cell) of the network, and takes the following input parameters.

1. **Outstanding Traffic Load (OTL):** OTL is the average amount of wireless network traffic waiting to be serviced, i.e., transmitted on air. E.g., OTL can be represented by the average packet queue size at the master node.

2. **Bit-Rate (BR):** BR is the bit-rate used for transmitting packets at the master node.

3. **Channel-Width (CW):** CW is the frequency-width of the transmission chan-

nel used in the the network.

4. **Wireless PHY Chain (WPC):** WPC represents the transmit wireless PHY chain at the master node. A WPC is defined by the set of algorithms, or *WPC elements* used at each block in the chain. E.g., $WPC_s = \{$interpolation factor= k, modulation algorithm= MOD_l, encoding algorithm= ENC_m, error-correction scheme= $ERR_n\}$ describes the WPC s.

It can be safely assumed that the possible values (with min/max bounds) for BR, CW, and the WPC elements are known. Also, a total ordering of the WPCs on the energy consumption metric is known. Let the function $E(i)$ denote the energy rank of the WPC i. Note that since all the possible PHY algorithms to be used in MAC-PHY protocols are known through their standard specifications, the energy ranking of possible WPCs can be calculated once and stored.

As seen from the pseudocode in Algorithm 12, the DEMD Algorithm tries to ensure that the outstanding traffic load on the network is within a specified range $[L_{low}, L_{high}]$. If the current traffic load (OTL_{curr}) is found to be higher than the threshold L_{high}, then the current PHY chain WPC_{curr} is updated to the next better state (in terms of communication performance), which will result in the minimum possible energy increase. This is an application of the Knob 1. Only when the PHY chain is at its maximum, we resort to Knob 2, where the bit-rate BR_{curr} is successively increased. As a last resort, if the BR_{curr} has reached its maximum value, the more drastic step of frequency-width increase (Knob 3) is taken.

The algorithm takes analogous steps for the opposite scenario, i.e., when OTL_{curr} decreases beyond threshold L_{low}. ΔT, L_{high}, and L_{low}, are the configurable parameters for the DEMD Algorithm.

As noted earlier, any change in PHY chain that results in a change in the PHY bit-rate is considered part of Knob 2, and excluded from Knob 1 (including the computation of sets $U^{WPC}, X^{WPC}, Y^{WPC}$). This essentially implies that the modulation

Algorithm 12 DEMD Algorithm

Require: OTL_{curr}, BR_{curr}, CW_{curr}, WPC_{curr}
Require: L_{high}, L_{low}
1: $U^{WPC} \leftarrow$ Universal set of possible WPCs
2: $X^{WPC} \leftarrow \{x : x \in U^{WPC}, E(x) > E(WPC_{curr})\}$
3: $Y^{WPC} \leftarrow \{y : y \in U^{WPC}, E(y) < E(WPC_{curr})\}$
4: **if** $OTL_{curr} > L_{high}$ **then**
5: **if** $X^{WPC} \neq \Phi$ **then**
6: Update WPC with lowest rank element in X^{WPC}
7: Update X^{WPC}, Y^{WPC}
8: **else if** $BR_{curr} < BR_{max}$ **then**
9: Increase BR_{curr} to next value
10: **else if** $CW_{curr} < CW_{max}$ **then**
11: Increase CW_{curr} to next value
12: **end if**
13: **end if**
14: **if** $OTL_{curr} < L_{low}$ **then**
15: **if** $Y^{WPC} \neq \Phi$ **then**
16: Update WPC with largest rank element in Y^{WPC}
17: Update X^{WPC}, Y^{WPC}
18: **else if** $BR_{curr} > BR_{min}$ **then**
19: Lower BR_{curr} to next value
20: **else if** $CW_{curr} > CW_{max}$ **then**
21: Lower CW_{curr} to next value
22: **end if**
23: **end if**

scheme manipulation is excluded from Knob 1.

6.6.3 Discussion

First of all, we emphasize that the DEMD Algorithm must not override the critical actions of the specific DSA MAC-PHY protocol which should always take first priority. It should be implemented as an "add-on" optimization, rather than as a mandatory enforcement. E.g., if a channel-width change is infeasible due to licensed user activity, the DEMD Algorithm's Knob 3 cannot be enforced. Similar is the scenario where channel bitrate or error-correction algorithm (in the WPC) cannot be changed due to noise characteristics of the channel.

Secondly, the reason for the specific order of selecting control knobs (as seen in the Algorithm 12) is to minimize the energy increment as well as potential disruption to any ongoing communication. The energy profile of SDRs (Section 6.4) is instrumental in coming up with the Knob 1 $\rightarrow$ Knob 2 $\rightarrow$ Knob 3 ordering. Knob 1 constitutes simple changes at the PHY level which is seamless to ongoing communication. Knob 2, though still a simple update at the PHY chain, increases energy consumption to a much greater extent. Knob 3 not only results in a greater jump in energy consumption (like Knob 2) but may also cause disruptions to the ongoing data exchange. This is because exercising Knob 3 results in a change of the communication channel itself. As such, it may trigger network-wide changes, such as channel allocation scheme to be updated in the entire network or cause other DSA activities (e.g., spectrum sensing) to take place immediately. Thus, the DEMD Algorithm exercises Knob 3 as a last option.

Note that the DEMD Algorithm considers only the transmit PHY chain rather than the receive PHY chain, because the transmission characteristics of a packet determines the actual receive PHY chain processing. The receiver processes the incoming packet using algorithms for signal blocks that are complementary to those used for transmission by the sender. The transmit PHY information is typically contained in the PHY header of the packet, which is always transmitted using a mutually known scheme.

In summary, it is interesting to note that while radio processing as commodity software is the root cause of SDR's energy overhead, the same feature provides the capability and flexibility to adaptively manage and control its energy consumption through DEMD.

6.7 Evaluation

6.7.1 Implementation and Testbed

We implemented the DEMD Algorithm in GNU Radio as a dummy signal processing block (no real signal processing is done by the block), called `gr_demd`. It is the first node in the GNU Radio flowgraph representing the basic transmit WPC. Other nodes in the flowgraph are similar to the packet transmit flowgraph used in Section 6.4.5. However, the modulation node in the flowgraph can be dynamically updated with different types of modulation blocks, depending on the control of `gr_demd`. Further, the code to dynamically update the channel-width for USRP2 SRF, whenever needed, is also incorporated in `gr_demd`.

A separate python script acts as the traffic source, and continuously generates 500 byte packets at a specified data-rate by invoking `iperf` with appropriate parameters. The packets are then fed to the transmit WPC.

The testbed consists of 3 wireless nodes (laptop + USRP2 SRF), forming an infrastructure DSA WLAN. One of the nodes is the master node (the access point), and is also the transmitter, while the other two nodes are passive receivers. To coordinate the channel-width changes, `gr_demd` makes use of wired ethernet connection between the laptops. Thus, the ethernet connection between nodes emulates the control channel for this DSA WLAN.

As noted in Section 6.4.7, the limited computational throughput of our testbed systems restricts the usage of high bitrates, large channel-widths, and very intensive PHY algorithms. Consequently, we vary the traffic demand only upto 1Mbps to prevent overflow errors. For the same reason, we limit channel-widths upto 3.13MHz and use ethernet-based control channel emulation. Further, the the transmit WPC is simplified, and only the modulation scheme can be dynamically updated in our transmit WPC. Still, this WPC is quite generic and a powerful PHY chain for digital

communication. As described in Section 6.4.5, it similar to the PHY chain used in 802.11b (encoding, carrier sense, modulation), minus the error correction schemes and synchronization.

Our evaluation objective is to compare the energy consumption overhead for "DEMD" vs "no DEMD", over identical SDRs. We are interested in the percentage change rather than absolute values. Thus, the results obtained is generic and representative of SDRs in general, especially since GPP computation (which is the main causes of energy overhead) is the common feature across all the SDR platforms.

The values of the testbed parameters are: $CW \in \{0.2\ (= CW_{min}),\ 0.39,\ 0.78,\ 1.56,\ 3.13\ (= CW_{max})\}$MHz, $\Delta T = 0.5$s, $L_{high} = 32$kbps, $L_{low} = 1kbps$, $BR \in \{1$ (DBPSK), 2 (DQPSK), 3 (D8PSK), 4 (16QAM)$\}$bit/s/Hz. [3]

We use two evaluation metrics: *Traffic-Demand Fulfillment Ratio* (TFR), and *Power Consumption Overhead* (PCO). T is calculated as the ratio of the data throughput achieved at the receiver over the traffic-demand at the sender.

6.7.2 Results and Discussion

We first test DEMD with the control Knob 3 turned off in the DEMD Algorithm. In other words, the channel-width cannot be changed dynamically, but other control knobs (Knob 1 and Knob 2) are active. This emulates the conventional wireless communication (including non-DSA), where the channel is fixed, but the PHY bitrate can be changed adaptively (by changing modulation scheme) to improve performance, e.g., 802.11b/g. The fixed channel-width is kept at 0.78MHz, and the initial modulation is DBPSK.

Fig. 6.11 compares the TFR and PCO metrics when DEMD is active to the scenario where it is absent. As seen from the graph, DEMD lowers the energy overhead significantly—average reduction of 22.14 percentage points is observed in power con-

[3] We provide PHY bitrate in units of spectral efficiency, as the channel-width can change.

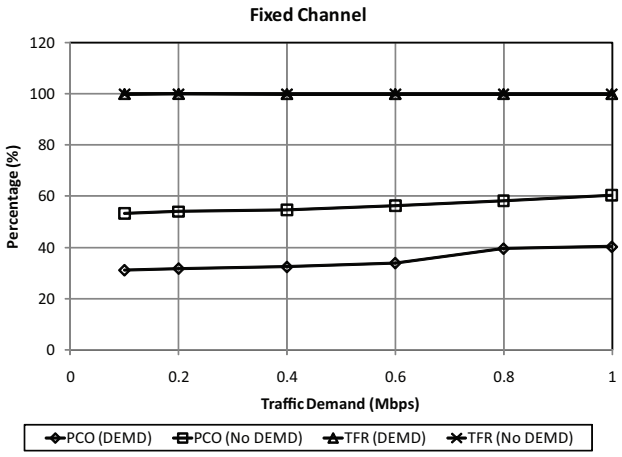


Figure 6.11: Energy and performance comparison when channel is fixed (channel-width = 0.78MHz).

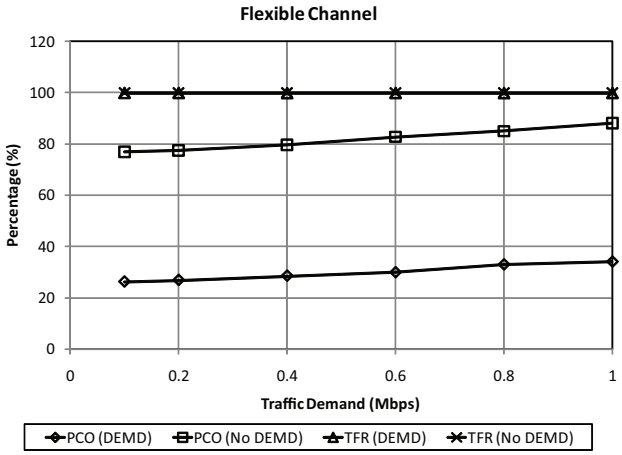


Figure 6.12: Energy and performance comparison when channel can adaptively change.

sumption increase. At the same time, the traffic demand is effectively met—TFR is close to 100% for all traffic loads. With the DEMD Algorithm, the transmitter is found to transmit with DBPSK as long as traffic demand can be met on the channel (upto 0.6Mbps requirement). It switches to DQPSK when the traffic-demand is 0.8Mbps or greater. Thus, with adaptively adjusting the performance on a need-basis, the energy overhead is lower with DEMD. On the other hand, when DEMD is absent, the transmitter immediately selects D8PSK to transmit for every traffic load, as the channel conditions are good (SNR is high). Consequently, the energy overhead is

much higher.

Next, we test the scenario where the channel-width can be adaptively changed depending on the availability. This setting reflects DSA behavior, where the available licensed spectrum segment is dynamically acquired for unlicensed operation [17, 54]. This also captures the situation where channel-width may change due to switching to a different spectrum region, e.g., switching from a 2.4GHz channel to a TV band channel. For this experiment, the initial channel-width is 0.78MHz.

Fig. 6.12 shows the comparison results. Once again, "DEMD" out-performs "No DEMD" case on energy front, while still being able to satisfy the traffic demand. In fact, the reduction in energy overhead is greater—average of 52.75%, as compared to 22.14% for the prior scenario (where channel-width could not be changed). We observe that with DEMD, the channel-width is soon reduced to 0.39MHz (Algorithm 12, lines 20-21). However, it increases to 3.13MHz when DEMD is absent, as the default DSA behavior is to acquire available spectrum in the licensed band order to improve performance. Channel-width increase is the most significant factor in energy consumption, as discussed in Section 6.4.7. Consequently, there is a greater difference in the energy overhead between "DEMD" and "No DEMD" cases, as seen in Fig. 6.12. With DEMD, the channel-width actually doesn't increase from 0.39MHz , as it is not required, even for 1Mbps traffic-load. We notice that just the modulation scheme in the transmit WPC changes to D8PSK to support the load.

For both "DEMD" and "No DEMD", there is a gradual increase in energy overhead as traffic-demand increases. This is expected because of the increase in volume of packet processing. The standard deviation for PCO and TFR readings is less than 2.9% and 0.13% respectively, in Figs. 6.11 and 6.12.

Our evaluation shows that DEMD is successful in balancing the performance vs. energy tradeoff on the SDR platform by adaptively managing the WPC. Though aimed at DSA, the approach followed in DEMD Algorithm is suited to non-DSA

scenarios as well. While the configuration parameters (bitrate, traffic load, channel-width) are kept low to not exceed the compute power of the testbed machines, we expect the results to scale and hold for higher numbers because the fundamental characteristics of the SDR platform will be similar.

Modern wireless NICs (ASIC-based) consume between 1-5W depending on their transmit power and mode of operation. In absolute measure, with DEMD enabled, we estimate our SDR's power consumption (with the USRP2 SRF) to range between 4-12W. Clearly, the energy usage is much more acceptable and closer to conventional radios. Note that without DEMD, the power consumption of the SDR ranges between 10-24W.

In conclusion, the results demonstrate that there must be a rethink in the wireless PHY design for SDRs. Performance alone as the primary design criteria (as is the case for conventional hardware/ASIC wireless radios), is not sufficient for SDRs as the energy consumption will be unacceptably high. In particular, next-generation technologies based on SDRs, like DSA, should incorporate energy consumption as a core element of their design.

6.8 Concluding Remarks

In this chapter, we investigated the energy consumption overhead of SDRs. We develop a basic energy profile for SDRs through testbed experiments and identify the relative impact of different factors that contribute to its high energy consumption. Our experimental analysis shows that the energy overhead incurred on SDRs is unsustainable for wireless communication. We conclude that there is a disconnect between conventional energy-agnostic wireless protocol design (especially the PHY design) and the SDR platform. This is a major concern for next-generation wireless technologies, such as DSA, that rely on the capabilities of SDRs.

We proposed a energy-aware solution for DSA, called Dynamic Energy Manage-

ment for DSA (DEMD), to address the energy consumption overhead problem of the underlying SDR platform. DEMD exploits the empirically obtained energy profile of SDRs to intelligently adapt DSA's energy consumption, while still ensuring good communication performance. The key component of DEMD is the DEMD Algorithm, which introduces adaptive PHY design for optimizing energy usage. The DEMD Algorithm exploits the capability of SDRs to dynamically update the PHY chain at runtime. Our preliminary evaluation shows that DEMD is effective in reducing the energy overhead associated with DSA when operating over the SDR platform.

CHAPTER VII

Conclusion

7.1 Research Contributions

The primary research contribution of this dissertation is to advance the scope and effectiveness of DSA technology, thus enabling its deployment in mainstream wireless systems and networks. The thesis explored and discovered the coexistence, QoS, and energy related problems associated with contemporary DSA. It presented practical, low-overhead, and system-oriented solutions to address the identified issues, together with their extensive evaluation and analysis. First, it described a novel approach to harness licensed spectrum in time-domain on an unlicensed basis (SpeCWiFi), thus improving DSA coexistence. Second, it characterized the impact of DSA on application QoS by analyzing its unwanted side-effects, and presented useful QoS provisioning features in context of DSA (QPDP). Third, it proposed an application-aware DSA optimization approach (CASA) for device-centric QoS management. Fourth, it developed end-to-end connection management scheme (DSASync), that provides network-level support for QoS in DSA networks. Finally, the thesis analyzed the energy consumption involved in DSA, and proposed an optimization strategy (DEMD) to lower its energy-usage overhead.

7.2 Future Directions

This research work can be extended further as follows.

- **Scalability analysis of proposed methods:** Our evaluation of the proposed solutions in this thesis involved experiments over a testbed featuring 5-10 wireless nodes. Although we utilized traces from large WLAN deployments in many cases, this research will greatly benefit from evaluation on a large-scale DSA testbed. Most importantly, such a study would provide valuable insights into the scalability of the methods proposed in this thesis. Additionally, experiments with different topologies as well as different traffic distributions will also contribute to understanding their behavior in diverse networking environments.

- **Domain-specific optimization:** To achieve the goal of generic applicability and deployment, the approaches presented in this thesis did not make any restrictive assumptions. However, many of the proposed techniques, especially CASA and DSASync, can benefit significantly from platform/domain specific knowledge. For example, information about traffic characteristics on the DSAN can be utilized to proactively allocate buffer spaces in the DSASync framework for better performance. Further, the reliability of connection identification can also be improved, particularly for UDP flows. Approaches to provide prioritized service based on important criteria (like traffic classes) can also be explored.

- **Fault-tolerance and security aspects:** This thesis presented approaches that enhanced DSA along coexistence, QoS, and energy dimensions. These methods exploit several functions provided by the underlying DSA protocol and the SDR platform. However, fundamental DSA functions (like spectrum sensing) may not be always reliable, and can even fail. This may adversely impact the proposed techniques and their implementation (e.g., SpeCWiFi) that rely on such DSA functions. One can argue that the poor performance of

DSA enhancing schemes is expected and is a smaller problem when DSA itself is unreliable. However, studies on the impact of poor (or even faulty) DSA operation (e.g., due to sensing errors) on the proposed methods will contribute towards understanding their fault-tolerance properties. Along similar lines, the potential security threats and their impacts on the proposed schemes can be investigated, which were not considered in this thesis.

APPENDIX A

An Introduction to Approximate Entropy

A discrete time-series s is defined as the following.

$$s = [s_1, s_2, \ldots, s_i, s_{i+1}, \ldots], s_i \in R$$

Let the series s be bounded on the number of elements (say N). Consider the binary series s consisting of N elements or bits. ApEn is defined for each length L of consecutive bit vectors that can be constructed from s. For each vector i of length L, its correlation sum $C_i^L(r)$ encapsulates the (normalized) number of vectors (of size L) in s which are "similar" to i within resolution r.

$$C_i^L(r) = \frac{\text{Num. vectors of length } L \text{ similar to } i}{N-L+1}$$

The notion of "similarity" of two vectors is defined using the Hamming distance—the maximum corresponding-element difference of the two vectors. They are considered similar when the Hamming distance between them is less than, or equal to the resolution r.

Given the correlation sums for all vectors of size L within resolution r, the mean size L logarithmic correlation sum $\Phi^L(r)$ of the series s is defined as:

$$\Phi^L(r) = \frac{1}{N-L+1} \sum_{i=1}^{N-L+1} \log C_i^L(r).$$

Finally, approximate entropy of s is defined as:

$$ApEn(L, r, N)(s) = \begin{cases} \Phi^L(r) - \Phi^{L+1}(r) & \text{, if } L \geq 1 \\ -\Phi^1(r) & \text{, if } L = 0. \end{cases} \tag{A.1}$$

ApEn indicates the degree of regularity present in the sensing information s. As can be seen from Eq. (A.1), $ApEn \leq 1$. Large values of ApEn (e.g., > 0.9) denote high irregularity in s, while small values of ApEn (e.g., < 0.1) point to the presence of a regular pattern in s. In this context, ApEn predicts the probability of occurrence of any pattern in s.

For a binary time-series (where $s_i \in \{0, 1\}, \forall i$), the possible values for r are either 0 or 1. Assigning $r = 0$ ensures the strictest comparison of vectors in s for more accurate pattern detection.

APPENDIX B

Primary and Secondary User Emulation

Testbed evaluation for many of the solutions proposed in this thesis (like SpeCWiFi, CASA, and DSASync) requires two categories of wireless devices—*Primary User* (PU), and, *Secondary User* (SU). However, lack of transmission license in licensed channels and unavailability of inexpensive testing equipment for unlicensed bands forced us to emulate PU behavior with off-the-shelf 802.11 cards. Also, at the time of writing this thesis, there is no available implementation of any DSA protocol. This is mainly because DSA is an upcoming field of research with no consensus yet on a standard DSA protocol. Therefore, we emulated SU behavior as well, by implementing the abstract functional model of DSA (see Section 1.1) in 802.11 protocol. Our implementation serves as a model for conducting testbed-based studies of DSA with cheap, off-the-shelf 802.11-based devices in unlicensed spectrum.

We use open-source 802.11 device driver MadWifi [57] (version 0.9.4) on Linux platform (kernel 2.6.24) as the platform for building PU and SU prototypes. Atheros 802.11 cards are used in both PUs and SUs. Each SU or PU is a separate device, usually a laptop, in our experiments.

PU Emulator: We use click-1.6.9 modular router [97] together with MadWifi to emulate PU. Click is configured to generate a burst of packet streams at very short

intervals (every ≈150μs) to emulate a PU's ON period, while there is no transmission during OFF periods. The ON/OFF pattern and durations can be configured to conform to different distributions. To reduce variation in specified ON/OFF intervals, many MadWifi driver settings are controlled. However, due to software delays, about 9% PU ON/OFF periods show an error of more than 5%. Though not a 100% perfect emulation of ON/OFF periods, it proves to be a test in disguise for the adaptability and tolerance of our proposed solutions being tested.

The key challenge in emulating PUs using 802.11-based cards, is to prevent carrier sense and backoff during the PU's ON time. This is accomplished by setting appropriate MadWifi configuration parameters (e.g., *TXOP backoff* is disabled), and by ensuring transmission range asymmetry between PUs and SUs. We typically operate the PU transmitter at high power (18dBm), and it is placed far away ($\approx 25ft$) from SU nodes. SU nodes typically operate at 5dBm, and the PU receiver is kept within the range ($\approx 5ft$) of SU to analyze the resultant interference.

SU Emulator: The SU emulator implementation consists of 1800+ lines of code. It is fully contained within the MadWifi driver module *ath_pci* and the associated modified 802.11 state modules: *wlan_scan_ap*, *wlan*, and *wlan_scan_sta*. To compensate for hardware behavior that conforms to the 802.11 standard, certain approximations were utilized as described next.

To emulate the sensing QPI, various hardware queues' congestion window parameters are updated, depending on the QPI delay needed ($cwmin = cwmax = k$). Although this does not ensure exact delay every time (random *cwnd* could not be disabled as it is performed in hardware), it ensures that there is an average delay proportional to k. Also, for stopping/restarting transmissions, the kernel-provided functions[1] were used, together with the MadWifi-provided function to drain transmission queues.

[1] *netif_stop_queue* and *netif_wake_queue*

We found the *jiffies*-based kernel timer interrupt system to be of insufficient resolution and precision for our experiments. So, High resolution timers *hrtimers* were utilized to provide sufficient resolution with high precision interrupts (in order of microseconds). A high-resolution clock source is also needed (e.g., *acpi_pm* is available in all modern laptops) to actually provide fine-grained timer resolution.

Further, lack of floating-point in the kernel necessitated implementation of fixed-point arithmetic for calculating values like channel utilization and ApEn.

BIBLIOGRAPHY

[1] P. Kolodzy. *Spectrum Policy Task Force: Findings and Recommendations*, March 2003.

[2] Mark A. McHenry and Karl Steadman. *Spectrum Occupancy Measurements*. Shared Spectrum Company, 2004. `http://www.sharedspectrum.com/measurements`.

[3] D. Chen et al. Mining Spectrum Usage Data: a Large-scale Spectrum Measurement Study. In *ACM MOBICOM*, pages 13–24, September 2009.

[4] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty. NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer Networks Journal (Elsevier)*, 50:2127–2159, September 2006.

[5] FCC. *FCC Adopts Rules for Unlicensed Use of Television White Spaces*, November 2008. FCC News 202/418-0500.

[6] FCC. *Unlicensed Operation in the TV Broadcast Bands: Second Memorandum Opinion and Order*, September 2010. ET Docket No. FCC 10-174.

[7] Agam Shah. *Dell to offer 'white space' connectivity in laptops*. IT World, November 2008. `http://www.itworld.com/mobile-amp-wireless/57281/dell-offer-white-space%-connectivity-laptops`.

[8] Neeraj Srivastava and Sharon Hanson. *Expanding wireless communication with white spaces*. Dell Inc., October 2008. `http://www.dell.com/downloads/global/vectors/White_spaces.pdf`.

[9] Julie Vort. *Microsoft, Dell, Spectrum Bridge launch first public white spaces*. Network World, October 2009. `http://www.networkworld.com/community/node/46577`.

[10] Spectrum Bridge. *SpecEx: The Online Marketplace for Spectrum*. `http://www.specex.com/`.

[11] C. Zao, T. Jin, C. Chigan, and Z. Tian. QoS-aware Distributed Spectrum Sharing for Heterogeneous Wireless Cognitive Networks. *Computer Networks*, 52(4):864–878, 2008.

[12] S. Hamouda and B. Hamdaoui. Dynamic Spectrum Access in Heterogeneous Networks: HSDPA and WiMAX. In *IEEE IWCMC*, pages 1253–1257, June 2009.

[13] *GNU Radio*. `http://www.gnu.org/software/gnuradio/`.

[14] Ettus Research LLC. *Universal Software Radio Peripheral*. `http://www.ettus.com`.

[15] R. E. Ramos and K. Madani. A novel generic distributed intelligent reconfigurable mobile network architecture. *IEEE VTC*, pages 1927–1931, May 2002.

[16] J. Mitola and G. Q. Maguire. Cognitive Radio: Making Software Radios More Personal. In *IEEE Personal Communications*, pages 13–18, August 1999.

[17] P. Bahl, R. Chandra, T. Moscibroda, R. Murty, and M. Welsh. White Space Networking with Wi-Fi like Connectivity. In *ACM SIGCOMM*, August 2009.

[18] B. Hamdaoui and K. G. Shin. OS-MAC: An Efficient MAC Protocol for Spectrum-Agile Wireless Networks. *IEEE TMC*, 7(7), July 2008.

[19] C. Cordeiro and K. Challapali. C-MAC: A Cognitive MAC Protocol for Multi-Channel Wireless Network. In *IEEE DySPAN*, pages 147–157, April 2007.

[20] Q. Zhao, L. Tong, and A. Swami. A Cross-layer Approach to Cognitive MAC for Spectrum Agility. In *IEEE Asilomar Conference on Signals, Systems and Computers*, November 2005.

[21] *IEEE 802 LAN/MAN Standards Committee 802.22 WG on WRANs*. `http://www.ieee802.org/22/`.

[22] *ECMA 392: MAC and PHY for Operation in TV White Space*, December 2009. `http://www.ecma-international.org/publications/standards/ecma-392.htm`.

[23] S. Deb, V. Srinivasan, and R. Maheshwari. Dynamic Spectrum Access in DTV Whitespaces: Design Rules, Architecture and Algorithms. In *15th ACM MOBICOM*, pages 1–12, September 2009.

[24] *SDR Forum*. `http://www.sdrforum.org`.

[25] Y. N. Papantonopoulous. *High-speed ADC technology paves the way for software defined radio.*, August 2008. `http://www.rfdesignline.com/showArticle.jhtml?articleID=201202962`.

[26] Jean Kumagai. Radio revolutionaries. *IEEE Spectrum*, 4(1):28–32, January 2007.

[27] *USRP2*. `http://www.ettus.com/products`.

[28] L. Cao and H. Zheng. SPARTA: Stable and Efficient Spectrum Access in Next Generation Dynamic Spectrum Access Networks. In *IEEE INFOCOM*, April 2008.

[29] L. Yang, L. Cao, H. Zheng, and E. Belding. Traffic Aware Dynamic Spectrum Access. In *WICON*, November 2008.

[30] L. Berlemann et al. Unlicensed Operation of IEEE 802.16: Coexistence with 802.11(A) in Shared Frequency Bands. In *IEEE PIMRC*, pages 1–5, September 2006.

[31] R. Francisco, L. Huang, G. Dolmans, and H. Groot. Coexistence of ZigBee Wireless Sensor Networks and Bluetooth Inside a Vehicle. In *20th IEEE PIMRC*, pages 2700–2704, 2009.

[32] M. L. Huang and S. Park. A WLAN and ZigBee Coexistence Mechanism for Wearable Health Monitoring System. In *IEEE ISCIT*, pages 555–559, 2009.

[33] *IEEE Std 802.11 - 2007*, March 2007. LAN/MAN Committee of IEEE Computer Society, `http://standards.ieee.org/getieee802/download/802.11-2007.pdf`.

[34] *802.11n: Next-Generation Wireless LAN Technology*, April 2007. Broadcom white paper, `http://www.broadcom.com/collateral/wp/802_11n-WP100-R.pdf`.

[35] S. Geirhofer, L. Tong, and B. M. Sadler. Dynamic Spectrum Access in the Time Domain: Modeling and Exploiting White Space. *IEEE Communications Magazine*, 45(5), May 2007.

[36] S. Geirhofer, L. Tong, and B. M. Sadler. Cognitive Medium Access: A Protocol for Enhancing Coexistence in WLAN Bands. In *IEEE GLOBECOM*, pages 14–25, November 2007.

[37] S. H. Ahmad, M. Liu, T. Javidi, Q. Zhao, and B. Krishnamachari. Optimality of Myopic Sensing in Multi-Channel Opportunistic Access. *IEEE Transaction on Information Theory*, 55(9):4040–4050, September 2009.

[38] N. B. Chang and M. Liu. Optimal Channel Probing and Transmission Scheduling for Opportunistic Spectrum Access. *IEEE Transaction on Networking*, 17(6):1805–1818, December 2009.

[39] H. Kim and K. G. Shin. Efficient Discovery of Spectrum Opportunities with MAC-Layer Sensing in Cognitive Radio Networks. *IEEE TMC*, 7(5), May 2008.

[40] S. M. Pincus. Approximate Entropy as a Measure of System Complexity. *Proceedings of the National Academy of Sciences (PNAS)*, 88:2297–2301, March 1991.

[41] M. Wellens, J. Riihijarvi, and P. Mahonen. Evaluation of Spectrum Occupancy using Approximate and Multiscale Entropy Metrics. In *IEEE Workshop on Networking Technologies for Software Defined Radio Networks (SDR)*, June 2008.

[42] J. Choi, M. Jain, K. Srinivasan, P. Lewis, and S. Katti. Achieving Single Channel, Full Duplex Wireless Communication. In *16th ACM MOBICOM*, pages 1–12, September 2010.

[43] H. N. Pham, J. Xiang, Y. Zhang, and T. Skeie. QoS-Aware Channel Selection in Cognitive Radio Networks: A Game-Theoretic Approach. In *IEEE GLOBECOM*, pages 1–7, December 2008.

[44] K. L. A. Yau, P. Komisarczuk, and P. D. Teal. Context-Awareness and Intelligence in Distributed Cognitive Radio Networks: A Reinforcement Learning Approach. In *IEEE Communications Theory Workshop (AusCTW)*, pages 35–42, February 2010.

[45] Aggregation Aware Spectrum Assignment in Cognitive Ad-hoc Networks. D. Chen and Q. Zhang and W. Jia. In *IEEE CROWNCOM*, May 2008.

[46] M. Anand, E. Nightingale, and J. Flinn. Ghosts in the Machine: Interfaces for Better Power Management. In *2nd MOBISYS*, 2004.

[47] B. Higgins et al. Intentional networking: Opportunistic exploitation of mobile network diversity. In *16th ACM MOBICOM*, September 2010.

[48] S. Haykin. Cognitive Radio: Brain-empowered wireless communications. *IEEE JSAC*, 23(2):201–220, February 2005.

[49] Vanu Inc. Where Software meets the Spectrum. http://www.vanu.com.

[50] K. Tan et al. SORA: High Performance Software Radio Using General Purpose Multi-Core processors. In *NSDI*, April 2009.

[51] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste. Enabling MAC Protocol Implementations on Software-Defined Radios. In *NSDI*, April 2009.

[52] Y. Lin, H. Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner. SODA: A Low-power Architecture for Software Radio. In *33rd ISCA*, pages 89–101, June 2006.

[53] M. Woh, Y. Lin, S. Seo, S. Mahlke, T. Mudge, C. Chakrabarti, R. Bruce, D. Kershaw, A. Reid, M. Wilder, and K. Flautner. From SODA to Scotch: The Evolution of a Wireless Baseband Processor. In *41st MICRO*, pages 152–163, November 2008.

[54] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl. A Case for Adapting Channel Width in Wireless Network. In *ACM SIGCOMM*, August 2008.

[55] C. Chou, H. Kim, K. G. Shin, and S. Shankar N. What and how much to gain by spectral agility? *IEEE JSAC*, 25(3), April 2007.

[56] *IEEE P802.16h/D8*, November 2008. Part16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems.

[57] *The MadWifi Project*. `http://madwifi-project.org`.

[58] H. Kim and K. G. Shin. In-band Spectrum Sensing in Cognitive Radio Networks: Energy Detection or Feature Detection? In *14th ACM MOBICOM*, pages 14–25, September 2008.

[59] R. Gummadi, H. Balakrishnan, and S. Seshan. Metronome: Coordinating Spectrum Sharing in Heterogeneous Wireless Networks. In *1st COMSNETS*, pages 1–10, January 2009.

[60] K. E. Nolan et al. Value Creation and Migration in Adaptive Cognitive and Radio Systems. In *Cognitive Radio, Software Defined Radio, and Adaptive Wireless Systems*, chapter 5, pages 145–159. Springer, 2007.

[61] S. M. Pincus and B. H. Singer. Randomness and Degree of Irregularity. *Proceedings of the National Academy of Sciences (PNAS)*, 93:2083–2088, March 1996.

[62] D. Cysarz, H. Bettermann, and P. Van Leeuwen. Entropies of Short Binary Sequences in Heart Period Dynamics. *AJP - Heart and Circulatory Physiology*, 278(6):H2163–H2172, June 2000.

[63] Spectrum Bridge Inc. *First White Spaces Network*, October 2009. `http://spectrumbridge.com/web/images/pdfs/PR/claudville-whitespaceproje%ct-pressrelease.pdf`.

[64] R. Chandra, R. Mahajan, V. Padmanabhan, and M. Zhang. *CRAWDAD data set microsoft/osdi2006*, May 2007. `http://crawdad.cs.dartmouth.edu/microsoft/osdi2006`.

[65] Caleb Phillips and Suresh Singh. *CRAWDAD data set pdx/vwave*, Sept. 2007. `http://crawdad.cs.dartmouth.edu/pdx/vwave`.

[66] Xiofei Wang. *CRAWDAD trace set snu/wow_via_wimax*, Oct. 2009. `http://crawdad.cs.dartmouth.edu/snu/wow_via_wimax`.

[67] FCC. *Facilitating Opportunities for Flexible, Efficient and Reliable Spectrum Use Employing Cognitive Radio Technologies*, 2003. ET Docket No. 03-108.

[68] B. Ishibashi, N. Bouabdallah, and R. Boutaba. QoS Performance Analysis of Cognitive Radio-based Virtual Wireless Networks. In *IEEE INFOCOM*, pages 336–340, 2008.

[69] D. Wu. QoS Provisioning in Wireless Networks. *Wireless Communications and Mobile Computing*, 5(8):957–969, 2005.

[70] J. Kim and A. Jamalipour. Traffic Management and QoS Provisioning in Future Wireless IP Networks. *IEEE Personal Communications*, 8(5):46–55, October 2001.

[71] WiMedia Alliance. *WiMedia*. `http://www.wimedia.org/`.

[72] C. Cordeiro, M. Ghosh, D. Cavalcanti, and K. Challapali. Spectrum Sensing for Dynamic Spectrum Access of TV Bands. In *IEEE CrownCom*, August 2007.

[73] OPNET Technologies, Inc. *OPNET Modeler Wireless Suite*. `http://www.opnet.com/solutions/network_rd/modeler_wireless.html`.

[74] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1st edition, 1998.

[75] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, Internet Engineering Task Force, July 2003. `http://www.rfc-editor.org/rfc/rfc3550.txt`.

[76] Tim Szigeti and Christina Hattingh. *End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs*. Cisco Press, 1st edition, November 2004.

[77] Qusay Mahmoud. *Cognitive Networks: Towards Self-Aware Networks*. Wiley InterScience, 1st edition, September 2007.

[78] R. W. Thomas, L. A. DaSilva, and A. B. MacKenzie. Cognitive Networks. In *1st IEEE DySPAN*, pages 352–360, November 2005.

[79] K. L. A. Yau, P. Komisarczuk, and P. D. Teal. A Context-Aware and Intelligent Dynamic Channel Selection Scheme for Cognitive Radio Networks. In *IEEE CROWNCOM*, June 2009.

[80] Xipeng Xiao. *Technical, Commercial and Regulatory Challenges of QoS: An Internet Service Model Perspective*. Morgan Kaufmann, 1st edition, October 2008.

[81] Voip-Info.org. *VOIP QoS Requirements*, September 2009. `http://www.voip-info.org/wiki/view/QoS`.

[82] *Ekiga*. `http://ekiga.org`.

[83] H. Kim and K. G. Shin. Understanding Wi-Fi 2.0: From the Economical Perspective of Wireless Service Providers. *IEEE WCM*, 17(4), August 2010.

[84] Chun-Ting Chou, Hyoil Kim, K. G. Shin, and Sai Shankar N. What and How Much to Gain by Spectral Agility? *IEEE Journal on Selected Areas in Communications (JSAC)*, 25(3), April 2007.

[85] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving tcp/ip performance over wireless networks. In *1st ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pages 2–11, November 1995.

[86] A. Bakre and B. R. Badrinath. I-tcp: Indirect tcp for mobile hosts. In *15th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 136–143, May 1995.

[87] H. Schulzrine, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. *Internet RFC 3550*, July 2003. `http://www.ietf.org/rfc/rfc3550.text`.

[88] M. Gandetto and C. Regazzoni. Spectrum sensing: A distributed approach for cognitive terminals. *IEEE JSAC*, 25(3):546–557, 2007.

[89] N. B. Chang and M. Liu. Optimal Channel Probing and Transmission Scheduling for Opportunistic Spectrum Access. In *13th ACM MOBICOM*, pages 27–38, September 2007.

[90] *Iperf*. `http://sourceforge.net/projects/iperf`.

[91] *Advanced RF Management for Wireless Grids*. `http://www.arubanetworks.com/pdf/rf-for-grids.pdf`.

[92] Y. Cheng, J. Bellardo, P. Benko, A. C. Snoeren, G. M. Voelker, and S. Savage. Jigsaw: Solving the Puzzle of Enterprise 802.11 Analysis. In *ACM SIGCOMM*, pages 39–50, September 2006.

[93] J. Eriksson, S. Agarwal, P. Bahl, and J. Padhye. Feaibility Study of Mesh Networks for All-wireless Offices. In *MOBISYS*, pages 69–82, June 2006.

[94] *Universal Software Radio Peripheral*. `http://en.wikipedia.org/wiki/Universal_Software_Radio_Peripheral`.

[95] *GNU Radio*. `http://gnuradio.org/redmine/wiki/gnuradio`.

[96] *Agilent 54815A Infiniium Oscilloscope*. `http://www.home.agilent.com/agilent/product.jspx?nid=-536902798.5368808%87.00&cc=US&lc=eng`.

[97] *Click Modular Router*. `http://read.cs.ucla.edu/click/`.