Spring 2021

Scheduling based optimization in software defined radio and wireless networks

Nathan Daniel Price

Department: Electrical and Computer Engineering

SCHEDULING BASED OPTIMIZATION IN SOFTWARE DEFINED RADIO AND WIRELESS NETWORKS

by

NATHAN DANIEL PRICE

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

2021

Approved by

Dr. Maciej Zawodniok, Advisor
Dr. Kurt Kosbar
Dr. Sahra Sedigh Sarvestani
Dr. Joe Stanley
Dr. Venkata Sriram Siddhardh Nadendla

PUBLICATION DISSERTATION OPTION

This dissertation consists of the following three articles, formatted in the style used by the Missouri University of Science and Technology:

Paper I Transceivers as a Resource : Scheduling Time and Bandwidth in Software-Defined Radio, found on pages 7-37, has been accepted in the IEEE Access Journal.

Paper II Decoupling Hardware and Software Concerns in Aircraft Telemetry SDR Systems, found on pages 38-50, has been accepted by the International Telemetry Conference (ITC).

Paper III Optimizing Transceiver Reuse in a Multi-Radio Software-Defined Radio Platform by Markov Decision Process, found on pages 51-86, is intended for submission to the IEEE Transactions on Vehicular Technology.

ABSTRACT

The objective of this work is to enable dynamic sharing of software-defined radio (SDR) transceivers through the concepts of hardware virtualization and real-time resource management. SDR is a way to build a digital radio that consists of a software back-end for digital signal processing (DSP) and an analog front-end transceiver for waveform generation and reception. This work proposes the use of a virtualization layer to decouple back-end SDR software from front-end transceivers. With this arrangement, front-ends are said to be virtualized, and it becomes possible to share a limited number of front-ends among many SDR back-ends through different multiplexing techniques.

In the first work, the hardware/software infrastructure needed for such a system is explored. An intelligent resource management algorithm is presented that demonstrates a potential increase in the number of supported SDR back-ends. The second work presents an exploration of this system's application to aircraft telemetry systems and the potential improvements to reliability. The work includes a reliability model for virtualized SDR aircraft telemetry systems as well as simulations demonstrating changes in performance as hardware fails. In the final work, an improved resource management algorithm based on Markov decision process (MDP) is proposed. This approach addresses concerns wireless regulatory agencies and standards bodies may raise regarding performance degradation caused by sharing transceivers. The process of sharing transceivers causes service disruptions to occur whenever the instantaneous demand for front-ends exceeds capacity. This MDP approach provides a feasibility test and a guarantee that all SDRs can stay within their respective wireless specifications. The proposed technique guarantees Pareto efficient distribution of resources. To make this approach possible, a connection is established between dynamic transceiver sharing and equivalent interference.

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Dr. Maciej Zawodniok, for his support throughout the years. I would like thank him for finding unique opportunities that challenged me in unexpected ways. He always provided a positive collaborative research environment. I would like to express a special bit of thanks to my doctoral research committee members, Dr. Kurt Kosbar, Dr. Sahra Sedigh Sarvestani, Dr. Joe Stanley, and Dr. Venkata Sriram Siddhardh Nadendla. I would also like to thank former member, Dr. Ivan Guardiola. I would like to thank my family for their love and support. I would never have made it this far without their support, especially through the latter years of my studies.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION

1. INTRODUCTION

The combination of software-defined radio (SDR) and the modern mobile device (i.e., handheld computer) has the potential for extreme change in the way we interact with radio. If radio capabilities are implemented as software (i.e., software-defined radio), then the software management facilities of a mobile device can be used to add and remove wireless features at a whim. Just as the smartphone revolution put computing and software capabilities always at an arm's reach, the SDR revolution could make a varied array of radio systems just as accessible. Since radio capabilities would no longer be fixed, they could grow and change with user need. With this increase in flexibility, there is also the potential to reduce electronic waste.

The motivation of this work is to extend the capabilities of a potential SDR based mobile device by enabling radio transceivers to be shared among several SDRs on a single device. There is potential for several concurrent transmissions and receptions from several SDRs to be managed on a small number of transceivers. It would be similar to the way modern computers manage hundreds of independent compute processes and thousands of compute threads with only a limited number of processors.

1.1. INTRODUCTION TO SDR

Software-defined radio (SDR) is a method to create digital radio wherein the back-end digital signal processing is implemented with software rather than hardware. A digital radio can be partitioned into a front-end and a back-end. The front-end or transceiver is responsible for transmission/reception of the radio wave, analog signal processing, and digital

signal synthesis/sampling. The back-end is responsible for the encoding or modulation of a signal as well as the decoding or demodulation of a digital signal to and from a data source [33]. The front-end typically utilizes analog signal processing, but may employ digital signal processing, depending on front-end architecture. The back-end is almost exclusively a digital process. Front-ends perform the same tasks regardless of architecture; it is the back-end of a digital radio that distinguishes one type of digital radio from another. In an SDR, the back-end is implemented via a software program rather than a specialized piece of digital processing hardware. Usually, software is more easily changed than hardware. When a software back-end is combined with a flexible radio front-end, it allows the radio to change between many different wireless protocols and functionalities. Since the back-end distinguishes between different types of digital radio, and it is a piece of software, the radio is said to be *software-defined*.

1.2. HISTORY AND ADVANCEMENTS IN SDR

SDR systems and architecture have seen slow advancements in contrast to the rapid evolution of modern digital wireless communication. The origins of SDR can be traced to the mid 1980s. Most of the earliest publications on this topic such as [30, 32] came in the 1990s. In the next two decades, digital radio developed rapidly and became part of daily life. In the same period, the growth of SDR was impeded by the lack of both general-purpose processing power and sufficiently flexible front-ends. However, it did gain some traction as a research and development tool. As processing power has improved and better SDR front-ends have become available, [29, 43] the focus has shifted toward the software side of SDR.

While early SDR engineers saw the importance of being able to switch between different wireless protocols and functionalities [32], they likely never dreamed that modern devices would need to operate multiple wireless protocols concurrently. Modern smart-phones and mobile devices support three or more different wireless protocols (e.g., Wi-Fi,

Bluetooth, LTE, GPS, etc.) operating in parallel. While it would be possible to create a mobile device with each wireless protocol separately implemented as SDR, complications would arise. For example, an SDR based smartphone would consume more energy and require more processing capacity. However, improving software organization and management, can mitigate some of the challenges and provide some advantages to SDR.

The European Telecommunications Standards Institute's (ETSI) Reconfigurable Radio Systems (RRS) standard for SDR [2] manages to implement SDR's promise of flexible, modular radio while reducing some of the drawbacks. RRS demonstrates how improved software organization can make an SDR based smartphone or mobile device not only feasible but advantageous. It allows for the management and operation of multiple SDRs. The standard describes the different components of an SDR and defines application programming interfaces (API) between each component. This includes API between application software and host computer software. This API makes SDRs easy to install, remove, and change. It even describes the use of an app store for SDR applications. RRS defines API that makes it possible for these *SDR apps* to take advantage of hardware acceleration for both common and specialized radio signal processing tasks. This has the potential to reduce the processing overhead of SDR and reduce energy consumption.

However, an SDR based mobile device such as RRS would be fundamentally limited by the number of available front-ends (i.e., transceivers) unless creative solutions are developed. While a user would be free to install as many SDR apps as they desire, the device can enable, at best, one application per transceiver. If front-ends can be shared or multiplexed, then the number of SDR back-end applications can potentially exceed the number of front-ends. There are both economical and environmental motivations for pursing this area of research. Mobile device manufactures may be able to reduce the number of radio front-ends on each device, reducing costs and saving money. If the number of supported

SDR applications has reduced dependency on the number of front-ends, wireless features and capabilities may be added more easily to existing wireless devices. This can extend their useful life, counter planned obsolescence, and reduce waste.

Time-division multiplexing (TDM) for SDR front-ends has been explored to a limited extent. In [5], many of the API and terminology for RRS were developed. In the follow up articles [6, 45], a TDM system for front-ends was presented. SDR applications communicated their intention to transmit or receive through an API called a *spectrum access request.* These requests were scheduled in real-time based on a fixed application priority. If the number of concurrent requests exceed the number of available front-ends, lower priority requests would be dropped. This presents the potential that high-priority SDRs are overserved while low-priority SDRs are underserved. In [50], an API to describe dependencies between spectrum access requests was presented. It allowed the scheduler to make more intelligent decisions, because entire groups of dependent requests could be served or dropped together. While these TDM based scheduling techniques improved hosting capacity, they did not consider the damage that dropped requests would have on the performance and reliability of wireless protocols. Additionally, these methods had no measure for system capacity. A user could cripple the wireless capability of their device by simply activating too many radios.

In [26, 27], a system similar to RRS was presented that featured frequency division multiplexing (FDM) of front-ends. In [26], the authors used the terminology *virtualized SDR* to describe the concept of front-end multiplexing. In [26, 27], the Nyquist bandwidth of a SDR front-end was partitioned by means of digital frequency filtering, similar to carrier aggregation in LTE. This allowed multiple radios signals to be extracted from the reception of a single front-end and distributed to the appropriate back-end. Additionally, multiple radio signals from multiple SDR back-ends could be merged and transmitted from a single SDR front-end. While innovative, this technique has limited use cases, since it requires SDRs to operate on channels within the Nyquist bandwidth of a single front-end.

1.3. ORGANIZATION OF THE DISSERTATION

In this dissertation, a number of novel SDR front-end multiplexing schemes are presented to develop the topic and address concerns. Three papers are presented.

In the first paper, a novel SDR front-end multiplexing scheme was developed based on mixed-integer linear programming (MILP). First, a linear programming model was developed to build an optimized TDM schedule for spectrum access requests. Then, an improved model was presented that incorporates both TDM and FDM. A number of simulations were run comparing the two novel techniques with a fixed priority TDM scheduler and a variation of the fixed priority scheduler with FDM capabilities. These simulations tested the number of spectrum access requests that could be effectively scheduled given different numbers of front-ends and back-ends. The effectiveness of FDM was explored by varying the amount of Nyquist bandwidth available on each front-end.

In the second paper, the concept of SDR front-end multiplexing was applied to aircraft avionics and telemetry systems. The paper presented the idea of an aircraft where all radios systems have been replaced by a consolidated SDR system. In this application, front-ends were multiplexed using both TDM and FDM techniques. A simple reliability model for the proposed SDR was developed. It demonstrates how such a system can be very robust to hardware failures. Finally, a simulation was presented showing the gradual drop off in performance as the system suffers hardware failures. The scheduling strategy for this simulation is a hybrid of fixed priority scheduling and the MILP technique from the first paper.

The third paper presents an improved SDR front-end multiplexing scheme based in Markov decision process (MDP). This paper addressed many of the shortcomings of previous TDM schemes. A relationship was established between scheduling decisions, channel noise, and symbol error ratio (SER). This relationship was used in the decision process to consider the performance impact of each decision. This was used in combination with constraints to ensure each SDR stayed above a respective minimum SER. A component

of this effort was a feasibility test that could be used to evaluate an SDR workload. This test enables the system to check that the number of active SDRs is not above the scheduling capacity of the system.

PAPER

I. TRANSCEIVERS AS A RESOURCE : SCHEDULING TIME AND BANDWIDTH IN SOFTWARE-DEFINED RADIO

Nathan Daniel Price, Maciej J. Zawodniok, Ivan G. Guardiola

Department of Computer Engineering

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Tel: 573–341–6622

Email: ndpr43@mst.edu

ABSTRACT

In the future, software-defined radio may enable a mobile device to support multiple wireless protocols implemented as software applications. These applications, often referred to as waveform applications, could be added, updated, or removed from a software-radio device to meet changing demands. Current software-defined radio solutions grant an active waveform exclusive ownership of a specific transceiver or analog front-end. Since a wireless device has a limited number of front-ends, this approach puts a hard constraint on the number of concurrent waveform applications a device can support. A growing trend in software-defined radio research is to virtualize front-ends to allow sharing and reuse among active waveform applications. This poses a difficult scheduling challenge. This article proposes a new approach in which shared access to front-ends is managed by a mixed-integer linear programming model. This model ties together the technique of time-division sharing and front-end bandwidth channelization. This scheduling model is evaluated in

simulation under several different scenarios and workloads. Simulation results show that the proposed approach reduces hardware contention and missed radio accesses compared to existing techniques.

Keywords: Communication systems, Resource management, Scheduling algorithms, Software radio, Wireless communication

1. INTRODUCTION

The growing number of wireless standards and protocols operated concurrently on today's devices is unsustainable. Wireless applications such as aircraft avionic systems [4, 14], conventional cellular base stations [10], multiple radio access technology (multi-RAT) cellular base stations [5], Internet of things (IoT) hubs, and mobile devices support the concurrent operation of multiple wireless standards and protocols and could benefit from software-defined radio (SDR). Cellular modems support a growing number of wireless bands and protocols in an effort to globalize mobile devices and increase network capacity and performance. Modern mobile devices feature hardware for current generation cellular, legacy cellular, Bluetooth, Wi-Fi, global positioning system (GPS), near-field communication (NFC), and occasionally more. Currently, wireless protocols are supported by discrete transmitter/receiver chains encapsulated in discrete application-specific integrated circuits (ASIC). This approach is inflexible, scales poorly, and represents a missed opportunity for component reuse.

SDR offers a solution to the increasing number of standards, protocols, and modulations in today's wireless devices. Usually accredited to Joseph Mitola [12], the "software-radio" architecture provides flexibility by replacing fixed function hardware with software. A software-defined radio consists of a very generic radio frequency (RF) front-end built in analog hardware and an application-specific digital back-end built in software as illustrated in Figure 1. It is the use of a software back-end that distinguishes software-defined radio from traditional digital radio. Using the same front-end, a developer could make a software-

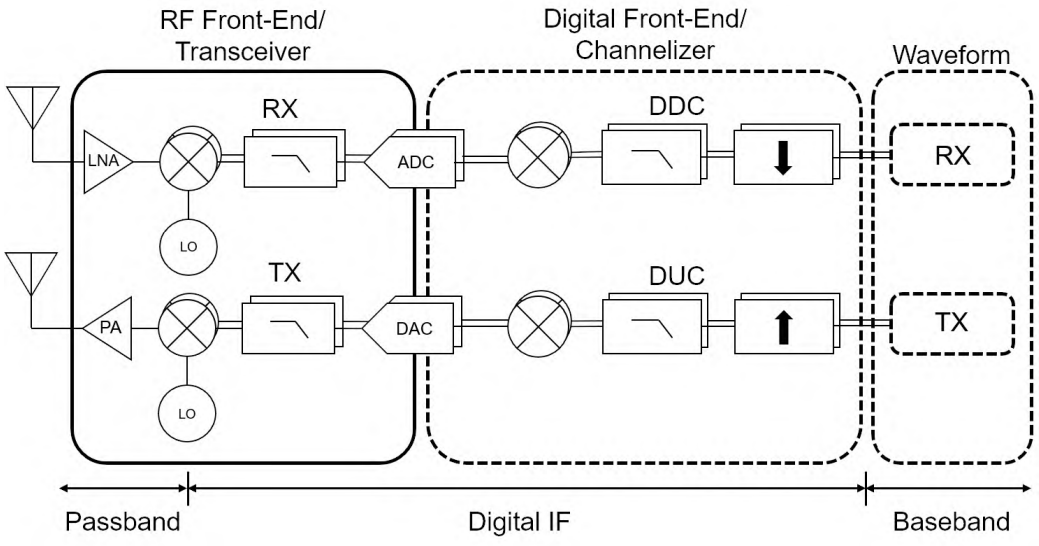


Figure 1. Simplified diagram of a typical direct conversion front-end based SDR architecture featured in many systems. The RF front-end converts signals between passband and baseband. Optionally, signals are digitally mixed to and from an intermediate frequency using a digital front-end or channelizer. Additional filtering and sample rate conversion can also be performed by the channelizer depending on waveform requirements.

defined GPS receiver, a software-defined garage door opener, etc., simply by changing the back-end software. In this way, wireless protocols are no longer an intrinsic feature of a device; rather, they are software applications often referred to as *waveform applications* [15, 22].

Front-end sharing and the coordination of waveform applications in multi-protocol, multi-radio communications systems provides a particularly interesting challenge. Typically, the maximum supported number of concurrent waveform applications is limited to the number of RF front-ends attached to the system since each application requires exclusive access to at least one. The key to supporting greater concurrency in SDR systems is to effectively multiplex or share RF front-ends among multiple waveform applications such that the number of applications is no longer bound to the number of front-ends.

The hardware/software interface is an ongoing challenge for software-defined radio developers. An interface solution is needed that abstracts front-end manufacturer specifics away from waveform developers and cuts down the massive amount of sample data transferring between front-ends and back-end applications. In [11], researchers identify software and driver limitations as one of the key barriers to general accessibility of SDR by the wireless community. Their reasons include the lack of standardization between different front-ends from different manufacturers. Furthermore, they identify the interface between the computing device and the front-end as the bottleneck for current SDR technology. Here, they are specifically referring to the massive amount sample data from each front-end. In [19], the hardware interface and hardware abstraction is also identified as a major challenge.

We advocate for hardware abstraction via the virtualization of RF front-ends using a hypervisor as the solution to the SDR interface problem. Originally proposed in [10], the concept is to present each waveform application with a virtual radio interface (VRI) in place of the RF front-end's actual driver interface. This benefits waveform applications by providing a common interface to all applications regardless of underlying front-end hardware or manufacturer. Front-ends are shared and managed by a hypervisor. Resource sharing conducted by the hypervisor cuts down on the number of required front-ends. This is heavily dependent on effective resource-sharing techniques.

We propose virtualization be achieved with the combination of dynamically time-division multiplexing (TDM) [2] and dynamic front-end bandwidth channelization [10], a form of frequency-division multiplexing (FDM). Waveform applications produce transmission sample streams and consume reception sample streams. Ordinarily, a front-end would be required for each application at all times to either supply or sink these sample streams. However, radios in many modern wireless protocols spend significant periods of time idle where they are neither transmitting nor receiving. These idle periods may be the result of a medium access technology such as time division duplexing (TDD), the result of energy saving techniques such as burst transmissions in IoT, or other technologies. During

these idle periods it is possible to let a different waveform application make use of an idle front-end as highlighted in Figure 2. A single front-end has been shown to be able to effectively service multiple active waveform applications by time division multiplexing in [8]. Off-the-shelf SDR front-ends are often designed to have a large amount of transceiver bandwidth to accommodate a range of different applications. By strategically tuning the center frequency of a wideband front-end, multiple target signals can be band-pass filtered out or *channelized* [10]. Figure 3 illustrates how four target signals could be isolated from two different passband segments. Ideally, this would require only two front-ends and can service four waveform applications.

The combination of these multiplexing techniques provides a difficult scheduling challenge for SDR platforms. Dynamic channelization is subject to the bandwidth of available front-ends and also the spectrum requirements of concurrent spectrum access requests [3]. At one instance, we may find multiple requests that are adjacent in frequency and, therefore, favorable for bandwidth channelization. At another instance, spectrum requests may be spread out across multiple bands or simply spread wider than the bandwidth of a single front-end, preventing any kind of channelization. An effective scheduler should be able to adapt to these changing conditions.

In this article, we propose the use of a constraint-based scheduling model based in mixed-integer linear programming (MILP). This approach allows us to accurately model the complicated conditional nature of dynamic channelization. It also allows for time constraint modeling, making it effective for combining the two mentioned multiplexing techniques. In this article we make the following contributions:

- We describe the details of our version of a virtualized RF front-end. This includes a discussion of the hypervisor, the virtual radio interfaces, and associated API. Additionally, we include a discussion of what scheduling considerations are needed for such a system.

- We introduce two MILP models for scheduling front-end access. We present a basic TDM model as well as model featuring both TDM and FDM in the form of channelization.

- We evaluate two schedulers based on a MILP models as well as two variations of a first-come first-served (FCFS) scheduler under different criteria to test their effectiveness and scalability. Simulation parameters include the number of request-generating applications, the number of available RF front-ends, and front-end bandwidth.

2. PROPOSED NOMENCLATURE

To better distinguish between different capabilities and features on parallel SDR hosting platforms and to better distinguish our work from existing research, we propose a naming convention. This naming convention describes a mapping relationship between waveform generating applications and the RF front-ends. We limit our discussion to single-in, single-out (SISO) radio techniques. For our purposes, an RF front-end can include any architecture of front-end (i.e., RF ADC/DAC, direct conversion, heterodyne, etc.). We categorize the different relationships between front-ends and waveform applications as follows:

1. *Single waveform, single front-end* (SWSF) includes conventional SDR waveform applications wherein an application has exclusive control over a front-end (i.e., a 1:1 relationship). Nearly all current waveform applications written fall into this category.

2. *Single waveform, multiple front-end* (SWMF) describes mapping schemes in which the transmission/reception stream of a single waveform application is simultaneously mapped to multiple front-ends (i.e., a 1:M relationship). This classification is included for completeness. The authors can only speculate at reasons why a scheduler would duplicate a SISO signal across multiple front-ends.

3. *Multiple waveform, single front-end* (MWSF) includes the schemes proposed in this article and similar research. A front-end is shared among several waveform applications (i.e., an N:1 relationship). This can be achieved through different techniques including time-division multiplexing [22] and channelization of front-end bandwidth [10].

4. *Multiple waveform, multiple front-end* (MWMF), describes a relationship that is a composition of MWSF and SWMF techniques (i.e., an N:M relationship). This implies that unique transmission/reception streams are aggregated, blended, and repartitioned.

Each of these classifications can be further categorized as a *static* or *dynamic* assignment. Static assignments occur before run-time as early as the design phase and as late as application initialization. Dynamic assignments occur during run-time. Dynamic assignments can better adapt to changing circumstances. For example, if a user were to launch an additional waveform application on a congested system, an SDR hypervisor could respond by reassigning active applications to make room for the new one. Without dynamic assignment capability, the user's new waveform application might be prevented from running.

3. LITERATURE REVIEW

In this section, we look at existing research endeavors and the techniques used to create a parallel SDR hosting platform. Much of the existing research involving software-defined radio has used SDR as a means to an end for other wireless research. For example, SDR has been used as a prototyping platform, a foundation for cognitive radio [13], etc. However, the goal of this article is to develop SDR systems that support the operation of multiple waveform applications in parallel. Some research groups have proposed and

implemented a parallel SDR systems of the dynamic MWSF type using TDM. These TDM systems use a policy-based scheduler. A few groups have also explored both static and dynamic MWSF using a channelizer.

Typically software-defined radio implementations fall into the category of static SWSF. No common SDR programming framework yet provides front-end multiplexing capability. Typical, waveform applications are written on programming frameworks such as GNU Radio, LabVIEW, and MATLAB/Simulink. These frameworks provide libraries for signal processing but provide no front-end multiplexing capabilities. A developer using any of these frameworks typically connects their application directly with a device driver and obtains exclusive ownership over the device (i.e., a static SWSF design). Another SDR platform, REDHAWK, acts as both a programming framework as well as a hardware abstraction layer [1]. However, REDHAWK does not multiplex RF front-ends. It simply connects waveform applications in a static SWSF manner.

As far as the authors are aware, the earliest research into dynamic MWSF platforms was conducted by a joint venture between Nokia and NXP. The group's research focused mostly on SDR application structure and support infrastructure. The group suggested using an abstraction layer to make SDR applications independent of specific hardware [2]. They used the time-division multiplexing of hardware to create a dynamic MWSF system [3]. Their minimum unit of scheduling was a *spectrum access request*, or a finite-duration transmission or reception event. These spectrum access requests dictated when a transmission or reception should take place. They used FCFS scheduling combined with a a static priority system (i.e., requests from certain applications were preferred) [3, 20]. When two or more waveform applications requested a front-end during the same time period, the request from the higher priority waveform would be chosen while the other would be dropped. To avoid interference between waveform applications, no transmission request could ever be scheduled during a reception request. In [22], the group worked to mitigate dropped requests due to scheduling conflicts by adding support for conditional scheduling.

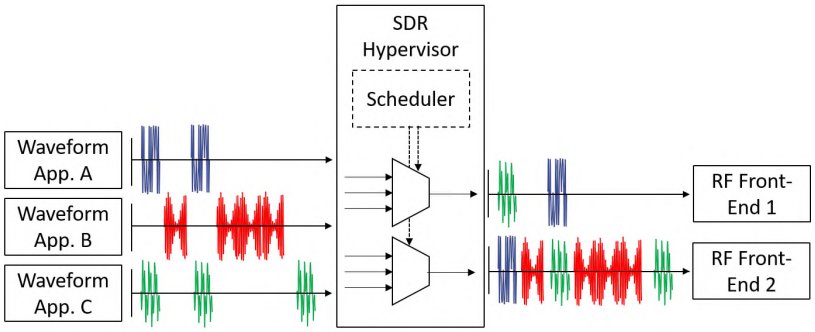


Figure 2. Time division multiplexing (TDM) implementation of multiple waveform, single front end (MWSF) software-defined radio (SDR) platform. The time scale is exaggerated for clarity. Time conflict between waveform applications A and C are resolved by migrating their respective signals to a different front-end. When all front-ends are busy and a conflict occurs, signals will be dropped.

This allowed applications to submit alternate timings for the same request. The scheduler could also be made aware of dependency between requests allowing it to schedule or drop such requests as a group.

Research efforts including [3, 20, 22] contributed to the European Telecommunications Standards Institute's (ETSI) reconfigurable radios systems (RRS) [16–18] and other emerging standards. RRS adopted the same organization and nomenclature as [3, 20, 22]; however, it curiously lack MWSF capabilities. Many of the software constructs in the RRS standard have even been adopted by the IEEE 1900 (DySpan) working group, which focuses on Cognitive Radio (CR) and dynamic spectrum sharing. Despite suggestions in contributing research [22], the RRS specification does not define a scheduling strategy and lacks any serious discussion of front-end multiplexing.

The Canadian research group LASSENA has demonstrated the use of a MWSF platform in software-defined avionics applications[4]. LASSENA's research proposed an avionics platform supporting five different avionics systems implemented as waveform applications [21]. Similarly, this system could be considered dynamic MWSF. LASSENA also implemented a FCFS scheduler with a static prioritization policy. There is also some discussion of channelization receiver bandwidth in [21], but this channelization was to be

statically assigned. The final implementation used a different transceiver architecture and did not feature a channelized receiver [4]. Their demonstrator unit was also limited to operating only two waveform applications concurrently.

In [9, 10], researchers introduced the concept of virtualized radio for SDR interfaces. Their platform, named Hypervisor for Software-Defined Radio (HyDRA), uses the concept of virtual radios. In HyDRA, each waveform application runs inside a software abstraction called a virtual radio while a hypervisor multiplexed access to the front-ends using front-end bandwidth channelization. A configuration API allowed applications to initialize a virtual radio interface by specifying their center frequency and bandwidth. HyDRA uses these specifications to channelize out a segment of the bandwidth from a front-end. It performed its channelization using a frequency domain implementation of band-pass filtering to reduce computation complexity. HyDRA prevents the creation of virtual radios with overlapping spectrum requirements to prevent interference. HyDRA can be considered a dynamic MWSF system since it allowed applications to share a front-end through channelization, and it could even reconfigure the channelization on-the-fly. Dynamic channelization alone shows small benefits outside of very particular use cases as will be shown later.

As far as the authors are aware, the research presented in this literature review represents the only efforts to develop front-end sharing techniques. All dynamic MWSF efforts can be reduced to two approaches at dynamic MWSF mapping: time-division multiplexing and front-end bandwidth channelization. It is the goal of this paper to develop an improved method using both techniques.

4. FRAMEWORK OVERVIEW

In the proposed architecture, a hypervisor similar to HyDRA acts as a middle-man between waveform applications and RF front-ends for the purpose of sharing hardware. Waveform applications communicate their transmission and reception requests to the hy-

pervisor through a software abstraction called a VRI. VRIs feature specialized API tailored to describing dynamic spectrum requirements. The hypervisor accepts these spectrum requests and builds an optimized access schedule.

4.1. VIRTUAL RADIO INTERFACE

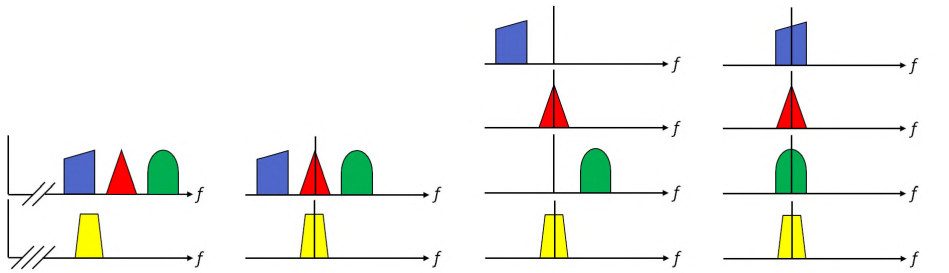
Our proposed solution makes use of VRIs to simplify interactions between waveform applications and front-ends while enabling front-end sharing. The VRI is a software abstraction that emulates a front-end device driver interface to provide communication between waveform applications and the hypervisor[10]. In an initialization phase, a waveform application can request access to the system's radio front-ends. This initial request includes specific RF requirements needed by the application. These requirements include RF band, transmitter power, receiver sensitivity, quality of service, and other details. The hypervisor uses the requirements in this initial request to screen out waveform applications it cannot service. If granted access, the hypervisor returns a VRI to the requesting application. The application can use this interface to request transmissions and receptions as if it were the interface to a real radio front-end. Through this anonymized interface, an application is shielded from hardware specifics. This helps to make waveform applications portable. Behind the scenes, the hypervisor can dynamically route sample streams to front-ends or drop traffic as it deems necessary.

Waveform applications communicate spectrum requests through a VRI using a control API for signaling transmission and reception. Each spectrum request consists of a sample buffer and control arguments. For transmissions, this sample buffer contains the base-band samples to be transmitted. For receptions, this sample buffer would initially be empty, and would be filled by the hypervisor after the reception has taken place. The control arguments indicate when, where, and how transmissions and receptions should take place. At minimum, these arguments indicate whether the request is a transmission or reception, frequency requirements, and timing requirements. Frequency requirements can

be indicated by a low-frequency cutoff and high-frequency cutoff. For example, Wi-Fi channel 6 is indicated by the frequency pair [2426, 2437] MHz. Timing requirements must indicate exactly when a transmission or reception must take place and for how long. This can be indicated using two of the following variables: start time, stop time, and duration. Additional control arguments should be added for an actual implementation; however, timing and frequency are the only arguments required for the optimization model proposed in this paper.

A simple way to implement a VRI would be to use an existing front-end control API for spectrum requests. This gives client applications the illusion that they are communicating directly with a real front-end and makes for universal software adoption. Many existing front-end control APIs including VME bus International Trade Association (VITA) 49, open base station architecture initiative (OBSAI), DigRF, and the common public radio interface (CPRI) have already implemented a complete API for spectrum requests. The authors prefer VITA 49 for many reasons including its growing industry adoption and its time synchronization capabilities [6].

Given a stream of spectrum requests from a VRI, the hypervisor must translate and pre-process the contained samples. The sample format (i.e., sample scaling, sample representation, etc.) and sampling rate must be converted from the waveform application's format to the front-end end format. Sample rate matching is easily one of the most computationally expensive parts of the SDR process as it must operate at the front-end's native sample rate. Fortunately, sample rate matching is often partially accelerated by the the front-end itself [11]. This is also the stage where channelization and front-end linearization are performed. Additional waveform processing could also be performed by the hypervisor here, enabling features such as adaptive gain control, coherent phase matching, and spectrum sensing.



(a) Passband (b) Digital IF Bands (c) Isolated IF Signals (d) Baseband Signals

Figure 3. When read left to write the figure demonstrates channelized reception. Figure (3a) shows three signals received by a single front-end in one band (indicated by the two slash marks on the axis), and a single signal received by a second front-end in a different band (indicated by three slash marks on the axis). Figure (3b) shows the signals after they have been shifted to a digital intermediate frequency by their respective analog front-ends. Figure (3c) shows the signals isolated by means of band-pass filtering. Fig (3d) shows the isolated signals shifted to baseband by means of a digital mixer.

4.2. SCHEDULING

Waveform scheduling with spectrum requests is an interval scheduling problem. Every spectrum request contains a start time and a stop time. The hypervisor must strictly honor these timings by transmitting and receiving spectrum requests during their respective intervals. Ignoring these timings means potentially interfering with a waveform's multiple access scheme. When multiple requests arrive with overlapping intervals, the hypervisor must dynamically map the requests to different front-ends. When the number of overlapping request intervals exceeds the number of front-ends, the hypervisor must drop excess requests.

4.3. CHANNELIZATION

The process of channelization of front-end bandwidth provides a means to increase front-end utilization in an SDR hosting platform. By incorporating channelization into waveform scheduling, the hypervisor now has the option to combine a set of requests if their intervals overlap in time and spectrum. The usefulness of channelization is subject to

the mixture of spectrum requests and the bandwidth of available front-ends. At minimum, channelization provides no benefit but also does not degrade capacity. A best case scenario is one in which multiple requests overlap in time and are adjacent in frequency. For example, a front-end with 20 MHz of bandwidth tuned to 98.1 MHz would receive the entire US FM radio band (i.e., 88.1-108.1 MHz). By breaking down this 20 MHz block, all one hundred FM channels could be received simultaneously using only a single front-end combined with one hundred instances of a software-defined FM receiver back-end. A similar technique could be used to simultaneously transmit over multiple channels.

5. FORMULATION

In this section, we propose MILP model for the scheduling of spectrum requests on an array of heterogeneous transceivers. The objective of our model is to build a feasible assignment schedule in which the greatest number of requests are serviced.

Here we present two variants of the proposed MILP model for scheduling spectrum demands. First, a MILP model is derived that allows for optimized dynamic MWSF operation in time. Next, a modified model that incorporates dynamic channelization into the scheduler is derived. The proposed models schedule over a finite horizon. Both models assume perfect knowledge of all requests in that horizon.

5.1. MILP FORMULATION

Each front-end has a single controllable element where the scheduler is concerned, that is, the carrier frequency oscillator. From recent SDR surveys [19] and [11] we find that most common SDR transceivers are direct conversion, which is also called zero intermediate frequency (IF). Direct conversion transceivers convert between passband signals and baseband signals with a single oscillator. More traditional heterodyning architectures, convert signals in two or more steps, requiring the tuning of multiple oscillators and the

Table 1. Description of input variables.

Spectrum Requests	
Request j's hard start time	$t_s^{(j)}$
Request j's hard deadline	$t_d^{(j)}$
The lower edge of request j's spectrum	$f_{min}^{(j)}$
The upper edge of request j's spectrum	$f_{max}^{(j)}$
Front-End	
FE i's lowest carrier freq.	$c_{min}^{(i)}$
FE i's highest carrier freq.	$c_{max}^{(i)}$
FE i's bandwidth	$b^{(i)}$

coordinating of image frequencies to avoid interference. To simplify matters, most SDR transceivers are designed as direct conversion. Having only a single controllable element also simplifies control for the proposed scheduler. Additional mixing and filtering can be performed by a digital front-end. When used in conjunction with a direct conversion front-end, this setup is sometimes referred to as a digital IF transceiver. A diagram can bee seen in Figure 1. The digital IF architecture will be used in the second formulation.

Using the parameters of each spectrum request, we construct the constraints of our optimization model. From the parameters, the scheduler must know all start times given as the vector $\boldsymbol{t_s}$ and stop times given as the vector $\boldsymbol{t_d}$ to meet time dependencies. It must know the lower frequency edge of each spectrum given as the vector $\boldsymbol{f_{min}}$ and the upper edge of each spectrum given as the set $\boldsymbol{f_{max}}$. Using the timing parameters, we construct constraints regarding the assignment of requests to front-ends. Using the frequency parameters, constraints can be made to ensure requests are assigned to front-ends with appropriate tuning capabilities.

Consider a software defined radio system containing m front-ends and a number of waveform applications. Let the i^{th} front-end have a tunable carrier frequency limited to the range $[c_{min}^{(i)}, c_{max}^{(i)}]$ where $i \in \{1, ..., m\}$ and $c_{min}^{(i)}, c_{max}^{(i)} \in \mathbb{R}_{\geq 0}$. Additionally, let the i^{th} front-end have a fixed bandwidth $b^{(i)}$. During runtime, waveform applications create a

combined total of n spectrum requests, each representing either transmission or reception. Let the j^{th} request have a hard start time $t_s^{(j)}$ and a hard deadline $t_d^{(j)}$ where $j \in \{1, ..., n\}$. As previously mentioned, the j^{th} request requires a finite amount of spectrum defined by the range $[f_{min}^{(j)}, f_{max}^{(j)}]$ where $f_{min}^{(j)}, f_{max}^{(j)} \in \mathbb{R}_{\geq 0}$.

The objective is to assign a maximal subset of n spectrum requests to m RF front-ends. We denote the assignment pair of the i^{th} front end and the j^{th} request with the Boolean assignment variable x_{ij}, and thus

$$x_{ij} = \begin{cases} 1 & \text{if the i}^{\text{th}} \text{ front-end is assigned the j}^{\text{th}} \text{ request} \\ 0 & \text{otherwise.} \end{cases}$$

Similarly we denote c_{ij} to be the carrier frequency selected while the i^{th} front-end serves the j^{th} request. Carrier frequency selections are subject to front-end carrier capabilities:

$$c_{min}^{(i)} \leq c_{ij} \leq c_{max}^{(i)} \quad \forall\, i, j. \tag{1}$$

Assignments are subject to the tuning capabilities of the front-end(s). The lower edge of a front-end's bandwidth envelope $c_{ij} - \frac{b^{(i)}}{2}$ must be less than or equal to the lowest frequency in an assigned request $f_{min}^{(j)}$. The reciprocal also applies; $c_{ij} + \frac{b^{(i)}}{2}$ must be greater than or equal to the highest frequency in the request $f_{max}^{(j)}$.

For an assignment $x_{ij} = 1$, it is implied that the i^{th} front-end will remain in a fixed configuration for the entire period $t_s^{(j)}$ to $t_d^{(j)}$. Spectrum request durations may overlap in time making many assignments mutually exclusive (i.e., a front-end cannot be tuned to two different carrier frequencies at the same time). We denote the time overlap of spectrum request j with spectrum request k using the constant variable δ_{jk}

$$\delta_{jk} = \begin{cases} 1 & \text{if the j}^{\text{th}} \text{ request overlaps the k}^{\text{th}} \text{ request in time} \\ 0 & \text{otherwise,} \end{cases}$$

where δ_{jk} can be assumed to be known *a priori*, since t_s and t_d are known for each request. Given a set of n requests and m front-ends, our complete problem can be stated as follows:

$$\text{Maximize} \sum_{i,j} x_{ij} \tag{2}$$

Subject to:

$$\sum_{i} x_{ij} \leq 1 \qquad \forall\, j \tag{3}$$

$$c_{ij} - \frac{b^{(i)}}{2} \leq f_{min}^{(j)} + (1 - x_{ij})M \qquad \forall\, i, j \tag{4}$$

$$c_{ij} + \frac{b^{(i)}}{2} \geq f_{max}^{(j)} - (1 - x_{ij})M \qquad \forall\, i, j \tag{5}$$

$$x_{ij} + x_{ik} \leq 2 - \delta_{jk} \qquad \forall\, i, j, k \tag{6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, i, j \tag{7}$$

$$c_{min}^{(i)} \leq c_{ij} \leq c_{max}^{(i)} \qquad \forall\, i, j. \tag{8}$$

The objective function (2), which is the double sum of x_{ij}, maximizes the number of assignments. Variations of this objective may include a weighted sum that gives priority to certain applications and/or certain requests. Prioritized demand schedulers are left for subsequent work.

The first constraint (3) ensures that no request is serviced by more than one front-end. That is to say that the row sum of the assignment x_{ij} must be less than or equal to one. This leaves two possibilities for each request: a request may be serviced by exactly one front-end (i.e., $x_{ij} = 1$), or it may not be serviced at all (i.e., $x_{ij} = 0$).

Equations (4) and (5) enforce the frequency tuning limitations of the front-ends. For a front-end i to service a request j, it must be tuned to an appropriate carrier frequency such that the radio's front-end bandwidth can completely cover the request's entire bandwidth. Equation (4) enforces the lower bound while (5) enforces the upper bound. These two constraints are enforced conditionally using the Big M method [7]. When front-end i is not assigned to demand j (i.e., $x_{ij} = 0$), a large constant M on the right-hand side of the equation m ensures (4) and (5) always evaluate true regardless of the left-hand side of the equation. If transceiver i is assigned to request j (i.e., $x_{ij} = 1$), then the second term on the right-hand side of (4) and (5) is reduced to zero, and the constraint must be considered.

Equation (6) ensures that no radio i services more than one request concurrently. When there is overlap in time between a pair of requests j and k, (i.e., $\delta_{jk} = 1$), the right-hand side of (6) evaluates to one meaning front-end i can serve demand j or demand k, but not both. When time overlap is not indicated (i.e., $\delta_{jk} = 0$), both requests may be serviced by the same front-end.

This formulation models the control parameters and constraints of an optimal TDM scheduler for spectrum requests. Using this model, a TDM scheduler can ensure the a maximal amount of spectrum requests are serviced. This helps to increase the number of waveform applications supported by an SDR hypervisor.

5.2. MILP WITH CHANNELIZATION

Our MILP formulation can be further optimized by leveraging channelization. If two signals are adjacent in frequency such that the combined bandwidth of both signals is less than the bandwidth of a single front-end, then both signals can be served simultaneously by that front-end. This technique is used for both receiving and transmitting. In this article, we assume half-duplex front-ends that either transmit or receive, but not both concurrently. With modification, this formulation could also be used for independent transmitters and receivers or even full-duplex front-ends.

We must add a constraint to prevent transmissions and receptions from being combined. In the previous model, one request was permitted per front-end at any time thanks to constraint (7). This made it unnecessary to distinguish between request types. Transmission and reception requests are ineligible to be served by the same front-end at the same time. We introduce the constant $y^{(j)}$ to indicate if a spectrum request is a transmission or reception, where

$$y^{(j)} = \begin{cases} 1 & \text{if the j}^{\text{th}} \text{ request is a transmission} \\ 0 & \text{if the j}^{\text{th}} \text{ request is a reception.} \end{cases}$$

Given a set of n requests and m front-ends, our complete problem can be stated as follows:

$$\text{Maximize} \sum_{i,j} x_{ij} \tag{9}$$

Subject to:

$$\sum_{i} x_{ij} \leq 1 \qquad \forall\, j \tag{10}$$

$$c_{ij} - \frac{b^{(i)}}{2} \leq f_{min}^{(j)} + (1 - x_{ij})M \qquad \forall\, i, j \tag{11}$$

$$c_{ij} + \frac{b^{(i)}}{2} \geq f_{max}^{(j)} - (1 - x_{ij})M \qquad \forall\, i, j \tag{12}$$

$$c_{ij} - c_{ik} \leq (3 - (x_{ij} + x_{ik} + \delta_{jk}))M \qquad \forall\, i, j, k \tag{13}$$

$$c_{ij} - c_{ik} \geq -(3 - (x_{ij} + x_{ik} + \delta_{jk}))M \qquad \forall\, i, j, k \tag{14}$$

$$x_{ij} + x_{ik} \leq 2 - \delta_{jk}(y^{(j)} + y^{(k)} - 2y^{(j)}y^{(k)}) \qquad \forall\, i, j, k \tag{15}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, i, j \tag{16}$$

$$c_{min}^{(i)} \leq c_{ij} \leq c_{max}^{(i)} \qquad \forall\, i, j \tag{17}$$

The MILP model for spectrum request scheduling with front-end channelization shares the basics of the previous scheduling model but with added constraints. The objective function and the first three constraints, (10), (11), and (12), remain the same. By doing away with constraint (6), this model allows multiple requests to be concurrently scheduled on the same front-end.

When a single front-end supports multiple requests using channelization, a common carrier frequency must be used. Since a front-end can be tuned to only one carrier frequency at a time and cannot be reconfigured while servicing a request, this frequency must satisfy all requests simultaneously. Constraints (13) and (14) conditionally enforce this through the use of the Big M method. Channelization occurs when two requests j and k are assigned to the same front-end ($x_{ij} = 1$ & $x_{ik} = 1$) and they overlap in time ($\delta_{jk} = 1$). This requires request j and k use a common carrier frequency ($c_{ij} = c_{ik}$). When $x_{ij} = x_{ik} = \delta_{jk} = 1$, the right-hand-side term in both (13) and (14) reduces to zero. Therefore, $c_{ij} - c_{ik} \geq 0$ and $c_{ij} - c_{ik} \leq 0$. This is only true if $c_{ij} - c_{ik} = 0$ or $c_{ij} = c_{ik}$. Under all other circumstances, constraints (13) and (14) always evaluate true due to the large constant M.

Constraint (15) ensures transmissions and receptions are not combined on the same front-end. Should two requests have no overlap in time ($\delta_{jk} = 0$), the right hand side reduces to two allowing both requests j and k to be serviced. When an overlap in time is indicated ($\delta_{jk} = 1$), the right-hand side of the equation can be approximated as $2 - XOR(y_j, y_k)$. When there is a type mismatch the right-hand side, (15) reduces to one forcing mutual exclusion of the two requests.

With these modifications, our scheduler can potentially service more waveform applications than the previous MILP model. Capacity increases are subject to the number of spectrum requests adjacent in frequency and the bandwidth of available front-ends.

Table 2. Simulation parameters.

Parameter	Value
Number of Waveforms n	1-10
Number of Front-Ends m	1-4
Front-End Bandwidth b	40, 80, 160, 240 MHz
Simulation Time t	1s
Request Generation Rate per App. λ	4×10^{-6}
Mean Idle time per App	50 ms
Request Generation Rate overall	$m\lambda$
Mean Spectrum Request Duration	$\frac{1}{\mu}$

6. SIMULATION SETUP

In this section, we discuss the simulation setup used to evaluate the different spectrum request scheduling optimization techniques presented in this paper. Included is a discussion of simulation parameters and random scenario building. All simulations were conducted using MATLAB. The objective of these simulations is to evaluate which criteria most affect the performance of MWSF schedulers. The number of front-ends, maximum front-end bandwidth, and the number of waveform applications were varied for each simulation. We predicted the number of front-ends would have the largest effect on TDM only scheduling techniques. We also predicted that greater front-end bandwidth would lead to a proportional increase in performance in scheduling techniques that feature channelization.

This simulation attempts to translate the behavior of modern digital protocols such as Wi-Fi and LTE to spectrum requests. The simulated waveform applications generated spectrum requests that were finite duration and relatively short. Both transmission and reception requests were allowed. Each waveform application had a fixed carrier frequency and fixed channel bandwidth for the duration of simulation. These were chosen at random before the beginning of the simulation. Carrier frequencies were chosen at random over a range of 500 MHz. Likewise, bandwidth requirements for each waveform were chosen

at random over and ranged 1-40 MHz. Bandwidth and carrier frequency were chosen randomly according to a uniform distribution to avoid biasing the results to favor schedulers with channelization that they benefit from tightly packed channels. All requests created for a waveform application had that application's carrier frequency and bandwidth.

A series of scenarios were randomly generated each featuring a unique set of waveform applications and that produced a list of spectrum requests. Once the simulation variables were fixed, we generated 500 unique scenarios. For each waveform application, a list of spectrum requests were generated randomly according to a Poisson process. A series of spectrum request start and stop times were generated as a Poisson arrival process with parameter λ and were therefore exponentially spaced. The duration of each spectrum request was exponentially distributed with mean $\frac{1}{\mu}$. A uniform random variable determines if each request is a transmission or reception. Requests were generated for a simulated period of one second. Due to the nature of the Poisson process, there exist idle period between requests. Many wireless protocols are never actually idle. They are always transmitting or receiving. Some protocols, however, feature a power saving state where they do neither. In this simulation, all applications feature idle periods. The collection of all spectrum requests from all waveform applications constitute a single common Poisson process with parameter $n\lambda$ where n is the number of waveform applications. A complete list of all simulation variables are listed in 2.

Each scenario was simulated and processed using a number of different scheduling techniques including both proposed MILP schedulers. The following scheduling methods were simulated: SWSF, unprioritized FCFS (MWSF), unprioritized FCFS with dynamic channelization (MWSF), the proposed MILP (MWSF), and the proposed MILP with dynamic channelization (MWSF). All MILP models were solved using MATLAB's built-in MILP solver. Each of the 80 combinations of simulation variables were simulated 500 times. This number of simulations allowed for a tight error bound for all schedulers on a 95% confidence interval.

Any MWSF scheduler should attempt to maintain a minimum performance. When operating a MWSF scheduler there will naturally be timing conflicts between spectrum requests that cannot be resolved except by dropping requests. These dropped requests are equivalent to corrupted transmissions or receptions. Fortunately, most wireless protocols feature retransmission schemes and tolerate a certain percentage of dropped frames. For example, a wireless protocol may allow 10% of transmitted frames to be corrupted due to channel noise and interference. For MWSF systems, the total percentage of dropped frames due to interference and due to timing conflicts in the schedule should be less than the protocol's tolerance. If we assume a very clear channel, we can use all of this tolerance for scheduling conflicts. Using a 10% figure as a guide, we targeted a threshold of 90% requests serviced.

In this simulation, MILP schedulers had complete knowledge of all requests. Therefore, MILP based schedulers should always return schedules servicing a maximal number of requests. In a real-time scenario with limited knowledge of upcoming requests, MILP schedulers will likely make decisions that are suboptimal.

7. RESULTS

In this section, we discuss the numerical results from the simulation. Results are presented as the average of 500 unique simulations. Plots include error bars for 95% confidence interval. Given that there are three different variables (i.e., number of waveform applications, number of front-ends, and front-end bandwidth), we evaluate the results by holding the other two parameters constant and evaluating the percentage of spectrum requests served. Performance results were returned in terms of percentage of requests served since the exact number of requests varied slightly in each simulated scenario.

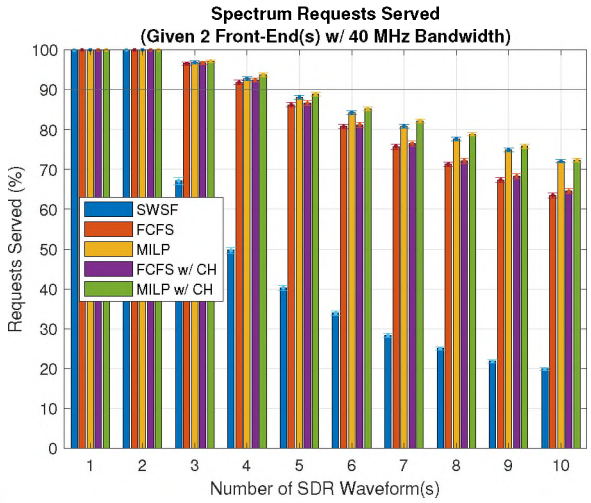


Figure 4. These graphs show the results of operating a parallel SDR hosting platform using different scheduling techniques. This setup featured two 40 MHz front-ends. Load was increased by upping the number of concurrent waveform applications. Notice that all MWSF schedulers allowed up to four applications to operate while servicing 90% of the requests.

7.1. THE BENEFITS OF MWSF SCHEDULING

We found that both MILP and FCFS schedulers improved the number of supported waveform applications on SDR hosting platforms when compared to SWSF based SDR. Figure 4 demonstrates a SDR setup that is typical of what can be found in a research lab. It features two front-ends each with 40 MHz of front-end bandwidth. Since all waveform applications generated requests according to identical distributions, they generate approximately the same number of requests. SWSF assigns each application to a single front-end. It therefore scales at a predictable rate of

$$\frac{\text{No. of FE}}{\text{No. of waveform apps.}} \tag{18}$$

All MWSF schedulers maintained the 90% target threshold for up to four waveform applications using only two front-ends. The SWSF scheduler maintained the threshold for only two waveforms.

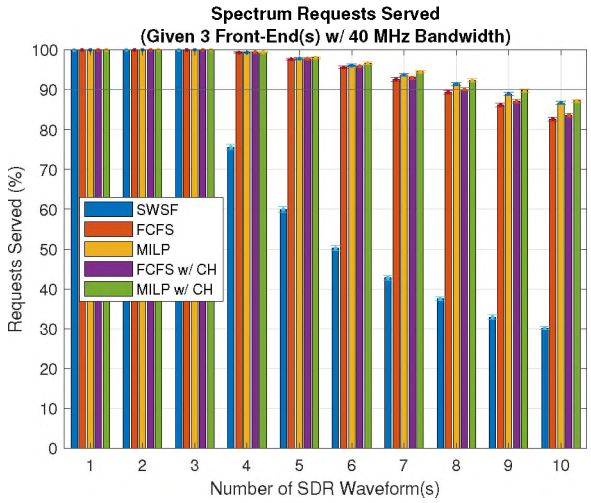


Figure 5. These graphs show the results of operating a parallel SDR hosting platform using different scheduling techniques. This setup featured three 40 MHz front-ends while the second featured three. Load was increased by upping the number of concurrent waveform applications. Notice that all MWSF schedulers allowed up to four applications to operate while servicing 90% of the requests.

As hardware specifications were increased, MWSF scheduling performance scaled as a factor of both front-end count and front-end bandwidth. In Figure 4, MWSF scheduled four applications while maintaining the target 90% service threshold. In Figure 5, FCFS and MILP schedulers maintained the target performance threshold for seven applications using three front-end while their variants with channelization supported an additional eighth application. In Figure 6 we observed that by quadrupling the bandwidth to 160 MHz, the bandwidth channelization variants of FCFS and MILP both increased capacity enough for an additional two waveform applications. This brought the total support to ten applications using three front-ends while maintain the 90% target service threshold.

FCFS and MILP schedulers represent the performance bounds for all MWSF schedulers. Since the proposed MILP scheduler had perfect knowledge of all requests it should always return a schedule serving the maximum number of requests. MILP, therefore,

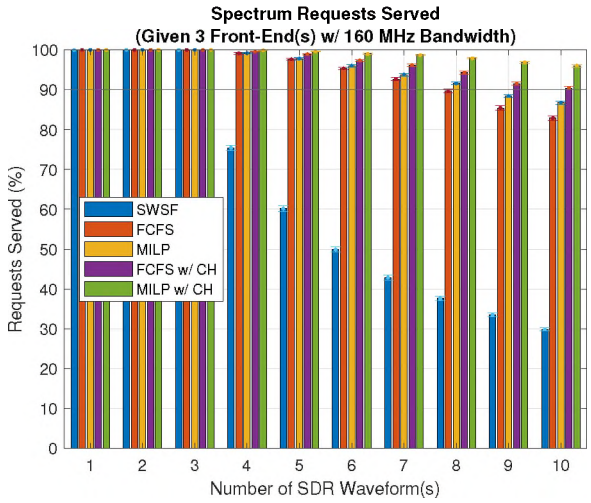


Figure 6. This graph shows the results of operating a parallel SDR hosting platform using multiple scheduling techniques. This setup featured three front-ends each with 160 MHz front-end bandwidth. Load was increased by upping the number of concurrent waveform applications. Each waveform application required up to 40 MHz of bandwidth.

represents the ceiling for MWSF performance. The simulated FCFS scheduling is computationally cheap but returns suboptimal results. Any MWSF scheduling technique represents a trade-off between the low computational cost of FCFS and maximal results of MILP.

7.2. PERFORMANCE EFFECTS

When tested in isolation, dynamic channelization of front-end bandwidth showed modest improvements that were subject to front-end bandwidth. In Figure 7, the number of waveform applications and the number of front-ends were held constant at ten and one respectively while the bandwidth was increased. This best demonstrates dynamic channelization's dependency on bandwith. Improvements can be as a small as 1-2% for FCFS and 2-3% for MILP at minimum bandwidth and as large as 10% for FCFS and 12% for MILP when bandwidth is increased to 240 MHz. Spectrum demands were limited to a

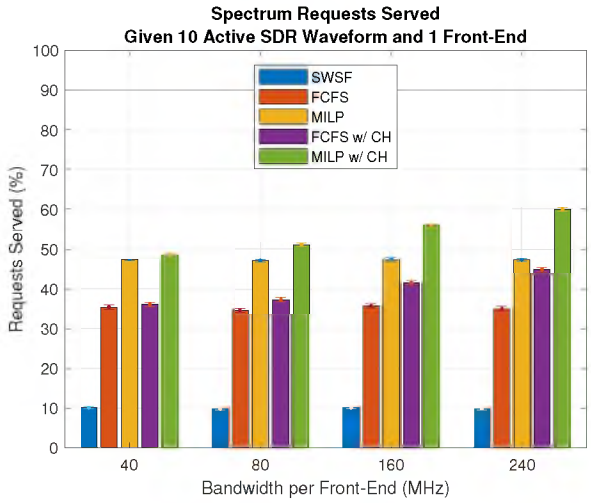


Figure 7. This graph shows demonstrates performance improvements had by increasing front-end bandwidth. This platform featured a single front-end with an increasing amount of front-end bandwidth. Load was held high at a constant 10 waveform applications. Each waveform application used up to 40 MHz of bandwidth. Schedulers using bandwidth dependent multiplexing techniques, channelization in this case, benefited as bandwidth was increased.

500 MHz range. Thus, a single front-end with 240 MHz represents 48% coverage over all of the accessible spectrum. Increased coverage improves the chances that multiple spectrum requests fall withing the bandwidth of a single front-end.

Unlike front-end bandwidth, increasing the number of front-ends in greatly improved performance. Figure 8 best demonstrates the benefit of additional front-ends. Three front-ends ensured that at least 80% of requests were served for all ten applications using any of the dynamic allocation techniques. The largest single improvement was made by moving from one to two front-ends for FCFS, resulting in a performance difference of 28%. There was little to no benefit from dynamic channelization in this figure since each front-end was limited to 40 MHz (i.e., the minimum). Three front-ends with 40 MHz of bandwidth represent 120 MHz of bandwidth or 24% coverage of the range when tuned with no overlap. Having multiple independent tuners with less bandwidth was a more effective strategy than a single wideband tuner.

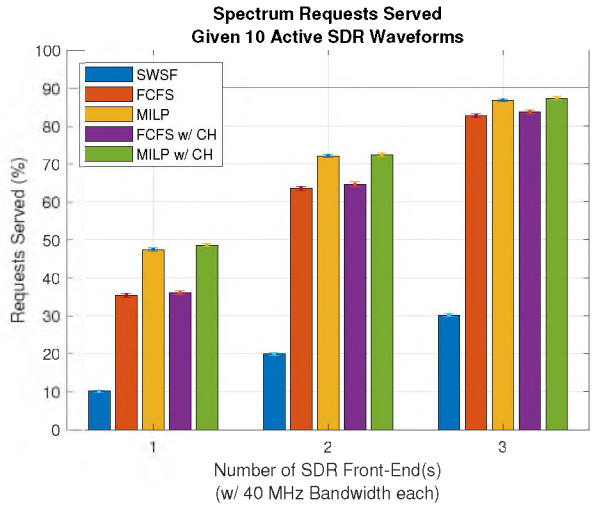


Figure 8. This graph shows demonstrates performance improvements had by increasing the number of front-ends. This platform featured an increasing number of front-ends each with a fixed 40 MHz bandwidth. Load was held high at a constant 10 waveform applications.

8. CONCLUSION

In our testing, we found that front-end count had the most significant performance impact on MWSF scheduling's performance. In all given examples, MWSF scheduling based on TDM at least doubled the number of supported waveform applications while servicing at least 90% of requests when compared to SWSF. The addition of dynamic front-end channelization to both presented MWSF schedulers increased performance to a lesser extent. By quadrupling the starting bandwidth, channelization added support for an additional one to two applications in the presented scenarios. Our presented MILP scheduling model returned an optimized schedule and represents the performance bound for MWSF scheduling.

Future work, should address the topic of quality of service and fairness. Scheduling methods in this article featured no prioritization for individual spectrum requests or waveform applications. The presented techniques weighted the number of requests serviced regardless of quality, type, or origin. A rogue application could manipulate its own priority in such a system by simply generating a great number of requests. Unlike existing research,

no effort was made to prevent transmission/reception collisions. In the authors' opinion it is the responsibility of the individual applications to avoid causing interference. Nevertheless, the hosting platform is in the best position to prevent interference before it happens.

REFERENCES

[1] 'REDHAWK,' https://redhawksdr.org/, 2020.

[2] Ahtiainen, A., Berg, H., and Westmeijer, J., 'Architecting Software Radio,' in 'Proceeding of the SDR '07' Technical Conference,' 2007 .

[3] Ahtiainen, A., van Berkel, K., Kampen, D. V., Moreira, O., Piipponen, A., and Zetterman, T., 'Multiradio Scheduling and Resource Sharing on a Software Defined Radio Computing Platform,' in 'Proceedings of the SDR '08 Technical Conference,' 2008 .

[4] Amrhar, A., Kisomi, A. A., Zhang, E., Zambrano, J., Thibeault, C., and Landry, R., 'Multi-Mode reconfigurable Software Defined Radio architecture for avionic radios,' in '2017 Integrated Communications Navigation and Surveillance (ICNS) Conference,' 2017 pp. 2D1–1–2D1–10, doi:10.1109/ICNSURV.2017.8011903.

[5] Chandrashekar, S., Maeder, A., Sartori, C., Höhne, T., Vejlgaard, B., and Chandramouli, D., '5G multi-RAT multi-connectivity architecture,' in '2016 IEEE International Conference on Communications Workshops, ICC 2016,' Institute of Electrical and Electronics Engineers Inc., ISBN 9781509004485, 2016 pp. 180–186, doi: 10.1109/ICCW.2016.7503785.

[6] Cooklev, T., Normoyle, R., and Clendenen, D., 'The VITA 49 Analog RF-Digital Interface,' IEEE Circuits and Systems Magazine, 2012, **12**(4), pp. 21–32, ISSN 1531-636X, doi:10.1109/MCAS.2012.2221520.

[7] Hillier, F. S. and Lieberman, G. J., *Introduction to Operations Research*, McGraw-Hill, New York, NY, USA, 10 edition, 2015, ISBN 978-0-07-352345-3.

[8] Kiminki, S., Saari, V., Pärssinen, A., Hirvisalo, V., Immonen, A., Ryynänen, J., and Zetterman, T., 'Design and Performance Trade-offs in Parallelized RF SDR Architecture,' in 'Proceedings of the 6th International ICST Conference on Cognitive Radio Oriented Wireless Networks and Communications,' IEEE, ISBN 978-1-936968-19-0, 2011 doi:10.4108/icst.crowncom.2011.245888.

[9] Kist, M., Rochol, J., Dasilva, L. A., and Both, C. B., 'HyDRA: A hypervisor for software defined radios to enable radio virtualization in mobile networks,' in '2017 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2017,' Institute of Electrical and Electronics Engineers Inc., ISBN 9781538627846, 2017 pp. 960–961, doi:10.1109/INFCOMW.2017.8116510.

[10] Kist, M., Rochol, J., DaSilva, L. A., and Both, C. B., 'SDR Virtualization in Future Mobile Networks: Enabling Multi-Programmable Air-Interfaces,' in '2018 IEEE International Conference on Communications (ICC),' IEEE, ISBN 978-1-5386-3180-5, 2018 pp. 1–6, doi:10.1109/ICC.2018.8422643.

[11] Machado, R. G. and Wyglinski, A. M., 'Software-Defined Radio: Bridging the Analog-Digital Divide,' Proceedings of the IEEE, 2015, **103**(3), pp. 409–423, ISSN 0018-9219, doi:10.1109/JPROC.2015.2399173.

[12] Mitola, J., 'The software radio architecture,' IEEE Communications Magazine, 1995, **33**(5), pp. 26–38, ISSN 01636804, doi:10.1109/35.393001.

[13] Mitola, J. and Maguire, G., 'Cognitive radio: making software radios more personal,' IEEE Personal Communications, 1999, **6**(4), pp. 13–18, ISSN 10709916, doi:10.1109/98.788210.

[14] Price, N. and Kosbar, K., 'Decoupling Hardware and Software Concerns in Aircraft Telemetry SDR Systems,' 2018.

[15] Reinhart, R. C., Kacpura, T. J., and Handler, L. M., 'TELECOMMUNICATIONS RADIO SYSTEM (STRS) ARCHITECTURE STANDARD : NASA-STD-4009A w/CHANGE 1,' 2018, **4009**, pp. 1–201.

[16] RRS, 'EN 303 146-2 - V1.2.1 - Reconfigurable Radio Systems (RRS); Mobile Device (MD) information models and protocols; Part 2: Reconfigurable Radio Frequency Interface (RRFI),' Technical report, 2016.

[17] RRS, 'EN 303 146-4 - V1.1.2 - Reconfigurable Radio Systems (RRS); Mobile Device (MD) information models and protocols; Part 4: Radio Programming Interface (RPI),' Technical report, 2017.

[18] RRS, 'EN 303 146-1 - V1.3.1 - Reconfigurable Radio Systems (RRS); Mobile Device (MD) information models and protocols; Part 1: Multiradio Interface (MURI),' Technical report, 2018.

[19] Sklivanitis, G., Gannon, A., Batalama, S. N., and Pados, D. A., 'Addressing next-generation wireless challenges with commercial software-defined radio platforms,' IEEE Communications Magazine, 2016, **54**(1), pp. 59–67, ISSN 0163-6804, doi:10.1109/MCOM.2016.7378427.

[20] van Berkel, K., Burchard, A., van Kampen, D., Kourzanov, P., Moreira, O., Piipponen, A., Raiskila, K., Slotte, S., van Splunter, M., and Zetterman, T., 'A Multi-radio SDR Technology Demonstrator,' in 'Proceedings of the SDR '09' Technical Conference,' 2009 .

[21] Yeste-Ojeda, O. A., Zambrano, J., and Landry, R., 'Design Of Integrated Mode S Transponder, ADS-B and Distance Measuring Equipment Transceivers,' 2016 .

[22] Zetterman, T., Piipponen, A., Raiskila, K., and Slotte, S., 'Multi-radio coexistence and collaboration on an SDR platform,' Analog Integrated Circuits and Signal Processing, 2011, **69**(2-3), p. 329, ISSN 0925-1030, 1573-1979, doi:10.1007/s10470-011-9713-7.

II. DECOUPLING HARDWARE AND SOFTWARE CONCERNS IN AIRCRAFT TELEMETRY SDR SYSTEMS

Nathan Daniel Price, Kurt Kosbar

Department of Computer Engineering

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Tel: 573–341–6622

Email: ndpr43@mst.edu

ABSTRACT

Prior work has shown that software defined radio has the ability to reduce the size, weight, power and cost of telemetry and avionics. We propose a virtualized transceiver architecture that supports multiple concurrent software defined radio (SDR) applications running on shared SDR hardware. This paper applies the concept of virtual transceivers to SDR for telemetry and avionics. The proposed design allows for transceivers to be shared between different SDR applications by taking advantage of time separation and frequency adjacency. This paper addresses the system layout, hardware selection, and software organization. Improvements include a scalable and considerations for both redundancy and upgradability.

1. INTRODUCTION

Software defined radio (SDR) systems have begun to make an impact in military and space applications. Examples include the Joint Tactical Radio System (JTRS) and NASA's Space Telecommunications Radio System (STRS). Software defined radio is an excellent

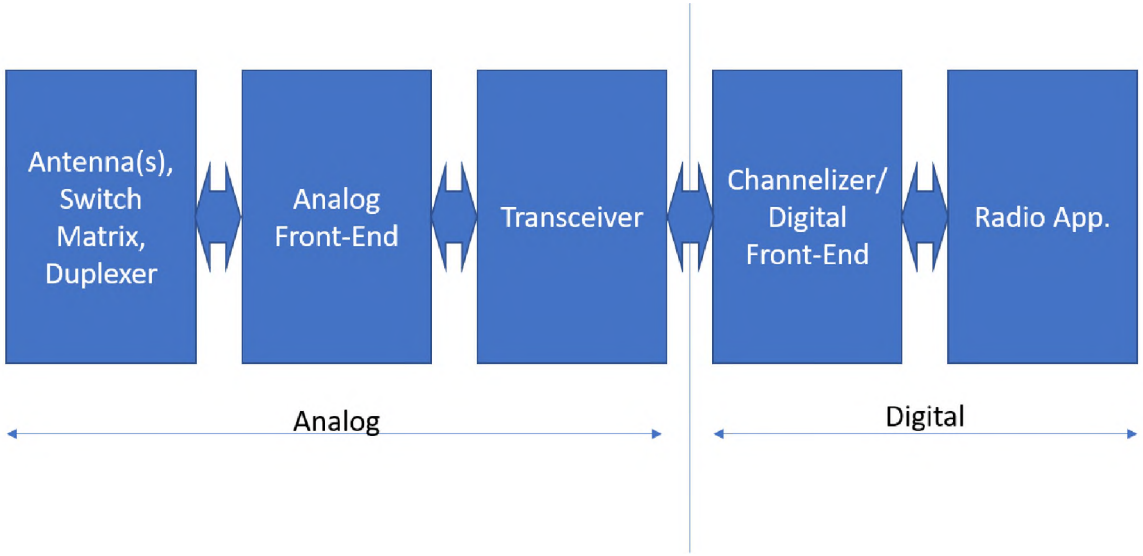


Figure 1. Conceptual block diagram of a software defined radio.

fit for avionics and telemetry as it makes it possible to support multiple radio bands and protocols with the versatility of software. SDR systems have the potential to reduce the size, weight, power, and cost (SWaP-C) in aeronautical radio equipment as well[1, 1, 3, 6].

Software defined radio systems support waveform-generating applications sometimes abbreviated as waveforms. These waveform-generating applications use digital signal processing to both produce and consume streams of digital samples representing baseband radio waves. It is the responsibility of transceiver hardware to realize these baseband sample streams into analog signals and to translate these signals to RF frequencies and vice versa. Key components of an SDR are shown in Figure 1.

1.1. PROBLEMS ADDRESSED

The trend of tight coupling of waveform-generating applications and transceivers in SDR platforms can lead to inflexibility in SDR systems as a whole. For example, statically offload DSP onto FPGAs integrated into SDR transceivers [3] can cause extremely tight coupling between waveforms and transceivers. This becomes a problem when the waveform-

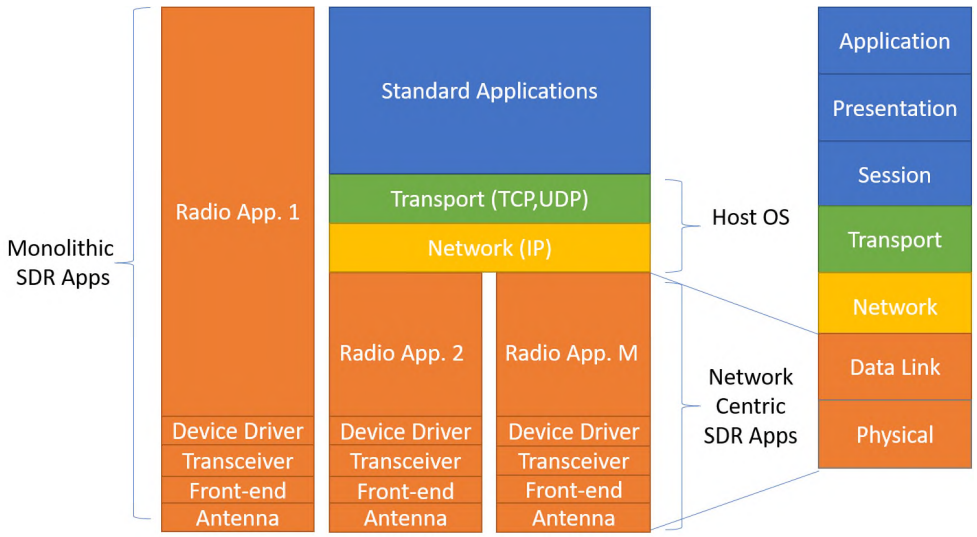


Figure 2. Software organization of classic SDR applications.

generating application is no longer distinguishable from the hardware it runs on. At this point the platform is better described as a firmware defined radio. When it come times to upgrade such a system, developers may find it difficult to add or change functionality. It could then be more economical to replace the entire system, which is the antithesis of a well-designed software defined radio system.

We propose the introduction of a hardware abstraction layer between waveform-generating applications and transceiver hardware. This separation leads to modularity between transceiver and computational hardware in a plug and play architecture. SDR applications communicate with the abstraction layer through a common API. The abstraction layer handles the timing, format, and delivery of signals as well as the configuration and control of transceivers. Additionally, it facilitates time division sharing of transceivers between different applications. Primary benefits of this abstraction layer include scalable performance overhead and a scalable increase in system reliability. The performance of a waveform measured in terms of contention-free accesses free accesses of transceivers

increases with the number of system wide transceivers. Likewise, reliability of waveforms measured in terms tolerance of failed transceivers increases with the number of system wide transceivers.

In this paper, we present the design and characteristics of the proposed software defined radio architecture. Described are design details in both hardware and software organization. Features of this architecture include transceiver multiplexing and resilience to device failure. The proposed architecture is simulated in a number of scenarios to demonstrate the performance and reliability scaling for ARNS and telemetry applications.

2. PROPOSED ARCHITECTURE

This section describes the hardware and software design of the proposed SDR architecture. First, we give an overview of the design and its functionality. We present software changes that serve to separate hardware and software development concerns. This includes a discussion of the responsibilities, features, and benefits of the proposed radio transceiver abstraction layer and a discussion of interlayer API. This is followed by a discussion about alterations to hardware that increase system flexibility.

2.1. SOFTWARE

The fundamental principle of this architecture is the abstraction of individual transceivers from waveform generating applications. Figure 2 illustrates the software organization of a classic SDR. Included are both network-centric applications featuring integration into the host operating system TCP/IP stack, as well as stand-along, monolithic applications. The waveforms are interacting directly with hardware drivers to control transceivers directly. We propose they interact with a common abstraction layer, as illus-

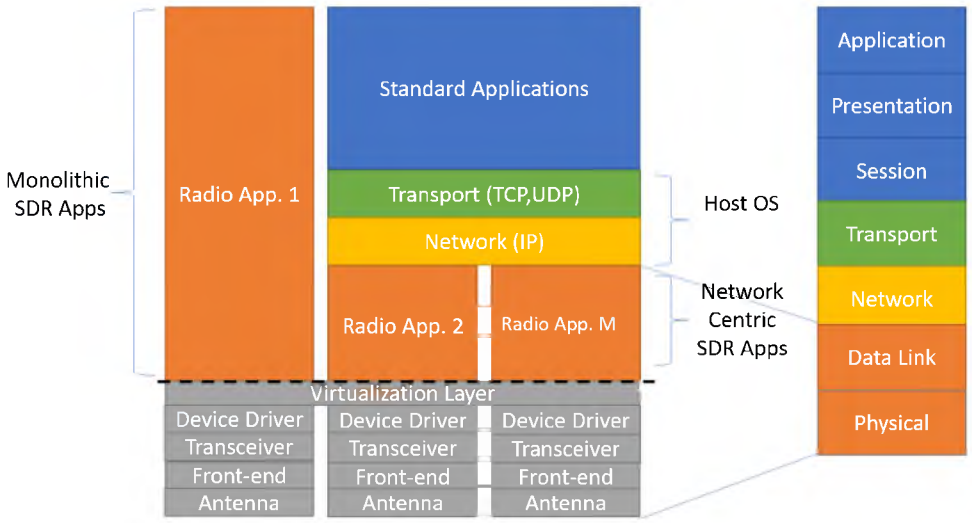


Figure 3. Software organization of SDR applications using virtualized transceivers.

trated in Figure 3. Gray layers in this figure are fully abstracted from SDR applications. We envision these interactions take place over a network socket-like API. We refer to each unique interaction, either a transmission or reception, as a spectrum demand.

2.1.1. Application Programming Interface. The API for each spectrum demand contains a sample buffer and minimal set of descriptors. For transmission demands, the sample buffer would contain samples to be sent, however for reception demands the buffer would be empty. For all spectrum demands, a waveform generating application would submit the following descriptors:

- Start Time: When the spectrum demand is to take place
- End Time: When the spectrum demand is to conclude
- Sample Rate: Sample rate of the submitted or requested samples in samples/second
- Lower frequency bound: The lower frequency edge of the requested or submitted spectrum in hertz.
- Upper frequency bound: The upper frequency edge of the requested or submitted spectrum in hertz.

- Reference power level: Relates samples with magnitude 1 to equivalent passband power level in dBm or Watts.

These descriptors, are inspired from low-level SDR peripheral interface protocols such as Vita 49 [2], however device specifics have been removed.

In a full implementation, supplementary descriptors could be added to spectrum demands to suite additional needs. Regulatory considerations would likely dictate additional mandatory fields. For example, transmission demands would likely need to specify emission specifics in the form of maximum total radiated power (TRP), maximum antenna gain, maximum out-of-band emissions, etc. Other practical considerations would likely dictate additional fields. For example, reception demands would likely need to specify a minimum signal quality metric in the form of spurious free dynamic range (SFDR), noise spectral density (NSD), or some other quality metric.

This abstract API allows for both transmissions and receptions to be handled asynchronously and optionally be event driven. For transmission demands, SDR application are free to generate a complete transmission demand and submit it for future transmission indicated in by the descriptors (i.e. asynchronous operation). Alternatively, the application could submit a transmission demand with an empty sample buffer and wait for a notification to begin start writing samples. This event notification would trigger before the actual start time of demand to compensate for transmission path delays. Once notified, the application must produce samples at least as fast the sample rate configured in the request to avoid underflow. Similarly, an application can submit a reception demand and check the sample buffer at its convenience, asynchronously. As with transmissions, the application can wait for an event indicating the first sample has arrived. This event would naturally occur after the start time in the reception request due to latency in the receive path. The application can read samples from the sample buffer no faster than the specified sample rate.

Much like a spectrum demand describes an individual transaction, each SDR transceiver must be registered with a complete set of the devices capabilities and descriptors. With a complete description of the device the abstraction layer can use this information to route spectrum demands to compatible transceivers. Tuning capabilities, front-end bandwidth, output gain, input gain, and duplex configuration are a just few examples of essential descriptors. This can also be extended to include any additional information the manufacture wishes to include.

2.1.2. Scheduling. The abstraction of spectrum demands allows the sharing of transceiver resources and hardware. We envision the abstraction layer containing a full scheduler to manage resource sharing. In the simplest case, an SDR system would contain a single transceiver. A user may run multiple SDR applications that each generate spectrum demands. The scheduler can then time share the transceiver by quickly reconfiguring the transceiver between different demands. Reconfiguring includes tuning to a different frequency, switching between transmission and reception, changing amplifier gain, switching between antennas, etc. When there is contention, demands occurring at the same time with conflicting requirements, the scheduler must pick a single demand to serve based on some preestablished priority system.

The scheduler may use a static or dynamic prioritization of spectrum demands to handle resource contention. A static priority system would choose demands based on known static properties. For example, it may favor demands from a specific application, demands that are short, etc. Alternatively, a scheduler may use a dynamic priority system in which priorities change. For, example a scheduler may attempt to ensure that at least 90% of spectrum demands are executed from a specific application.

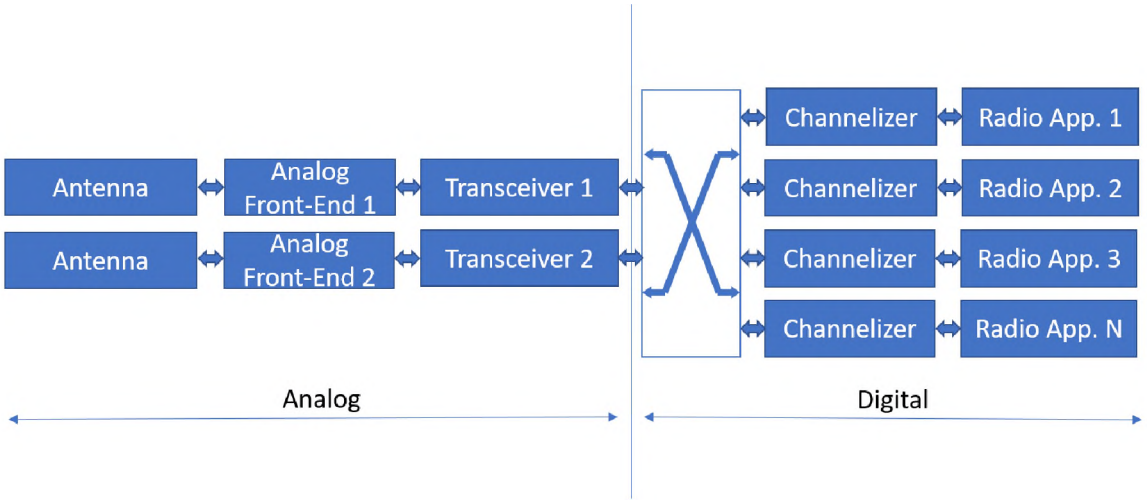


Figure 4. Conceptual block diagram for SDR system supporting multiple applications.

2.2. HARDWARE

Hardware considerations for the proposed design are largely the same as conventional SDR, however the flexibility of the system makes certain aspects easier. Analog transmission/receptions path components must be chosen to support desired frequency bands and front-end bandwidth. ADC/DAC sampling frequencies and bit precision must be selected by considering the needs of every radio application. Should no single radio support the needs of all radio applications, an array of receivers can be assembled. Figure4 illustrates a conceptual block diagram for an SDR system supporting multiple applications. This configuration allows for applications to be dynamically assigned to transmission/reception paths. Each applicatoin is facilitated by a dedicated, digital channelizer implemented in either software or hardware. The scheduler will automatically assign the correct transceiver to the correct application. Alternatively, duplicate transceivers can be added should many applications have similar requirements and large duty cycles.

The proposed design generally calls for a channelizer for each application, which has several benefits. Signals of interest can be extracted from incoming sample streams by digitally mixing and bandpass filtering (channelizing) out the respective blocks of spectrum.

The resultant sample streams can then be routed to the appropriate SDR applications. Sample rate conversion, sample precision conversion, sample format should also be integrated into this step. By reconfiguring the channelizers, SDR applications can easily be migrated between different transceivers.

In certain scenarios, it may be possible to tune a transceiver such that it may receive several signals at once. By carefully steering the front-end bandwidth of a receiver, multiple spectrum demands can be targeted with a single transceiver. This is subject to the dynamic range of ADC and the amplitude of incoming signals [1]. Spectrum demand descriptors should be extended to include such considerations if this operation is to be perform these operations dynamically by the scheduler.

Digital channelizers for each application comes at high computational cost. As samples enter and exit the DAC and ADC they must be stepped-down and stepped-up in real-time respectively. This process must be repeated for each application. Commercial SDR transceivers typically integrate a digital channelizer to save the host computer from performing this operation, however this makes channelizing out multiple signals for different applications[4]. Ideally, commercial SDR transceivers would include multiple hardware channelizers.

2.2.1. Reliability. In this section, we provide a simple model for the reliability of an SDR applications using our proposed architecture. Consolidating multiple avionics systems together can increase the risk of multiple systems failure, however the modularity of the proposed architecture can help mitigate the risk.

Given an implementation of our architecture supporting multiple SDR applications, we can model the reliability of an SDR application as a series-parallel system[5]. The system is comprised of a host computer running our applications with reliability R_c connected to m transceivers each with reliability R_{xcvr}. The reliability of an individual application R_{SDRi} can the be modeled as

$$R_{SDRi} = R_c[1 - (1 - R_{xcvr})^m] \tag{1}$$

if we assume all transceivers are compatible with the application. This is a worst case scenario as it features a single point of failure, the host computer. The system reliability is no greater than R_C

By including additional host computers as warm spares with reliability R_w we can improve the reliability to

$$R_{SDRi} = [1 - (1 - R_c)(1 - R_w)^{k-1}][1 - (1 - R_{xcvr})^m] \quad (2)$$

where k is the total number of host computers. In this way reliability can scailed in accordance with requirements.

3. SIMULATION

This section provides a simulation to demonstrate the performance of an abstracted multi-radio SDR system for avionics and telemetry. As previously discussed, each SDR application generates transmission and reception requests and it is the scheduler's duty to dynamically assign the requests to the available transceivers.

3.1. SETUP

This simulation will feature four avionics protocols as well and one telemetry protocol. In this simulation we will support the following avionics systems: distance measuring equipment (DME), air traffic control radar beacon system (ATCRBS), automatic dependent surveillance — broadcast transmitting (ADS-B Out) and receiving (ADS-B IN), a Mode S transponder. This part of the simulation is identical to that given by [6]. Additionally, we support a single serial streaming telemetry (SST) transmitter to be representative of telemetry needs.

Table 1. Transmitting and receiving characteristics as they pertain to scheduling.

Summary of Avionics Protocols and Spectrum Demands		
Protocol Name	Transmission Duration (Periodicity)	Reception Duration (Periodicity)
DME	15.5 μs (125-150 Hz)	150 μs (following each TX)
ATCRBS	20.3 μs (following each RX)	Continuous
ADS-B Out	120 μs (5-10 Hz)	N/A
ADS-B In	N/A	Continuous
Mode S	120 μs (following each RX)	Continuous
SST	Continuous	N/A

Our scheduler prioritizes spectrum demands based on their time duration. Finite duration transmission and receptions are the highest priority. SDR applications requiring constant reception will be assigned any time reaming up to an entire dedicated radio per application if available. Applications requiring constant transmission are the lowest priority and will receive any time after constant reception applications are assigned up to an entire dedicated radio available. Time duration requirements for each protocol in our simulation are summarized in Table 1.

3.1.1. Assumptions. We will assume that ATCRBS, ADS-B In, and Mode S reception demands can be met by the same transceiver. ATCRBS and Mode S ground stations both interrogate aircraft on 1030 MHz while ADS-B In arrives on 1090 MHz. All three of these demands are on-going. We will make use of the channelization technique mentioned earlier to receive all three protocols using one transceiver.

We will assume that ATCRBS replies, ADS-B Out, and Mode S replies will block the reception of other each other's reception. The transmissions made by these protocols are in the 100-500w range. Like [6] we assume that these in-band transmissions will cause saturation in our own receiver(s). Though we are not specifying which DME channel we are using, we will also assume that transmission made by DME will block in this band as well.

Table 2. Portion of transmission and reception demands met per application.

Summary of Spectrum Demands Met (%)						
Radio Count	1		2		3	
	TX	RX	TX	RX	TX	RX
DME	98.2	28.1	100	55.5	100	82.5
ATCRBS	68.2	98.2	69.6	97.9	70.2	97.4
ADS-B Out	100	N/A	100	N/A	100	N/A
ADS-B In	N/A	90.7	N/A	92.3	N/A	93.2
Mode S	91.7	95.9	84.9	95.1	83.7	96.7
SST	N/A	N/A	97.6	N/A	100	N/A

3.2. RESULTS

Table 2 summarizes the portion of transmission and reception demands that were met per application. Results are given for three different simulations featuring an increasing number of shared tranceivers. We see that transmission performance for finite duration demands initiated by the aircraft are excellent. The performance improved slightly with a greater number of transceivers available.

Mode S and ATCRBS show what appear to be contradicting trends. For Mode S, as the radio count increased the transmission performance degraded while ATCRBS showed the opposite trend. These protocols likely are approaching an equilibrium performance.

Reception demands tended to increase in performance as the radio count increased. DME's finite duration reception cycle improved drastically as the radio count went up indicating it was likely suffering from resource contention.

Continuous transmission reception demands stayed relatively constant showing only a slight increase or decrease in performance with exception of the SST. This was expected for continuous reception demands as the total scheduled time for finite duration demands compromised only 3%-4% of the entire simulation. Due to scheduling priorities already mentioned it is impossible for the SST to run with fewer than 2 radios on the system.

4. CONCLUSION

In this paper we proposed a new architecture for hosting multiple SDR programs using shared hardware. We detailed what software and hardware changes as well as benefits to SDR application programmers. Additionally, we demonstrated how the improved modularity can lead to an increase in radio system reliability and demonstrated performance scaling in a resource bound system.

REFERENCES

[1] Amrhar, A., Kisomi, A. A., Zhang, E., Zambrano, J., Thibeault, C., and Landry, R., 'Multi-Mode reconfigurable Software Defined Radio architecture for avionic radios,' in '2017 Integrated Communications Navigation and Surveillance (ICNS) Conference,' 2017 pp. 2D1–1–2D1–10, doi:10.1109/ICNSURV.2017.8011903.

[2] Cooklev, T., Normoyle, R., and Clendenen, D., 'The VITA 49 Analog RF-Digital Interface,' IEEE Circuits and Systems Magazine, 2012, **12**(4), pp. 21–32, ISSN 1531-636X, doi:10.1109/MCAS.2012.2221520.

[3] Don, M. and Ilg, M., 'Advances in a Low-Cost Software-Defined Telemetry System,' in 'International Telemetering Conference Proceedings,' International Foundation for Telemetering, 2017 .

[4] Sklivanitis, G., Gannon, A., Batalama, S. N., and Pados, D. A., 'Addressing next-generation wireless challenges with commercial software-defined radio platforms,' IEEE Communications Magazine, 2016, **54**(1), pp. 59–67, ISSN 0163-6804, doi: 10.1109/MCOM.2016.7378427.

[5] Trivedi, K. S., *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, John Wiley & Sons, Inc., New York, 2 edition, 2002, ISBN 0-471-33341-7.

[6] Yeste-Ojeda, O. A., Zambrano, J., and Landry, R., 'Design Of Integrated Mode S Transponder, ADS-B and Distance Measuring Equipment Transceivers,' 2016 .

III. OPTIMIZING TRANSCEIVER REUSE IN A MULTI-RADIO SOFTWARE-DEFINED RADIO PLATFORM BY MARKOV DECISION PROCESS

Nathan Daniel Price, Dr. Maciej Zawodniok

Department of Computer Engineering

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Tel: 573–341–6622

Email: ndpr43@mst.edu

ABSTRACT

This paper proposes a real-time control scheme for a software-defined radio (SDR) transceiver multiplexing system that provides service guarantees. Developing standards have detailed a well-organized system for hosting multiple SDR applications. Prior work has shown a potential to increase the number of hosted SDR applications by sharing or multiplexing radio front-end transceivers. Time sharing techniques have shown the most potential. However, service disruptions caused by over budgeting can lead to serious wireless performance degradation. This paper proposes a real-time scheduling technique based on a randomized, multi-objective Markov decision process to overcome this issue. First, a relationship is developed between service disruptions and channel interference. Next, this relationship is put to use in a flexible multi-objective Markov model. The wireless performance of each SDR application is modeled as a unique objective. An advanced solution technique is applied that provides unique satisfaction constraints for each objective. Simulations of the control scheme in operation are provided for multiple scenarios.

1. INTRODUCTION

Future software-defined radio (SDR) systems [24] will incorporate incredible flexibility. Developing standards [2] look to make adding and changing wireless capabilities in smartphones and mobile devices the same as installing new applications from an app store. These *SDR apps* will be managed by an abstraction layer that sits between the SDR app and the mobile device's radio hardware resources (i.e. radio front-ends or transceivers). With this next generation of SDR, users could pick and choose wireless capabilities as desired. The number of supported SDR apps would be limited only by the number of radio front-ends. However, several works [4, 5, 26, 28, 32] propose that this limitation could be partially overcome by dynamically sharing radio front-ends among SDR applications. In some literature [20, 21], this system architecture is even referred to as a virtualized SDR since it resembles virtual machines running on a hypervisor.

Operating SDR applications on shared radio front-ends introduces new challenges and concerns. Ordinarily, wireless devices must undergo testing, evaluation, and certification with wireless standards bodies and regulatory agencies. Testing ensures devices operate correctly and do not cause harmful interference. Conventional testing becomes impossible when users can add new wireless capabilities in unique, untested combinations to SDR based mobile devices. Inevitably, certain combinations of demanding SDR apps will contend for radio resources resulting in performance losses. Radio resource management systems for this new type of SDR must, therefore, have the capability to independently verifying that SDR apps operate within their respective specifications despite the presence of resource contention.

Virtualized SDR radio systems have many applications. Aircraft avionics systems operate multiple radio communication and radio navigation systems[16]. Virtualized SDR allows for the consolidation and potentially a reduction in radio equipment. Mobile devices such as laptops, tablets, and phones regularly operate multiple wireless protocols simultaneously. Virtualized SDR can add tremendous amount of flexibility and upgradability to these

devices. Wireless network infrastructure can also benefit from virtualized SDR systems. It enables cellular networks to more easily role out support for new wireless protocols via software updates. It can provide a foundation for multiple radio access technology (Multi RAT) based networks [11]. Virtualized SDR systems also benefit small scale wireless network infrastructure. It enables wireless local area network (WLAN) access points to become convergence devices that support multiple wireless protocols simultaneously.

In this article, the focus is on a novel dynamic decision making strategy for managing radio front-ends. The proposed solution is based on Markov decision process (MDP). We develop a Markov model that considers the RF performance requirements of each SDR application and considers the impact each resource assignment decision has on overall performance. We applied a well-developed MDP decision strategy [8, 9] that identifies feasible goals, guarantees a respective performance minimum for each SDR app, and ensures the solution is approximately Pareto efficient. Contributions of this article are as follows:

1. We propose a MDP model for dynamic front-end sharing in virtualized SDR systems. The state space of the model can be adapted to model wireless protocol behavior at varying levels of detail. The cost function considers the performance impact on physical layer performance caused by sharing radio front-ends. It ties together physical layer performance, the wireless channel model, and front-end assignment decisions. Most importantly, it considers the impact of overbooking front-end resources.

2. The proposed method identifies feasible workloads and can guarantee performance minimums. The methods applied from [8, 9] allow us to test an SDR workload (i.e., a unique combination of SDR apps) against physical layer performance requirements. Workloads deemed feasible come with the guarantee that a two-mode, stochastic control strategy can be used. It ensures all virtualized SDRs will meet or exceed their

respective minimum performance requirements. This type of performance guarantee provides a first-step in the exploration of achieving wireless certification for a true software-only radio.

3. The MDP strategy proposed for solving the Markov model produces a simple, approximately Pareto efficient, real-time stochastic control strategy for feasible workloads. This control strategy can easily be run in real-time. Decisions made at each epoch are chosen according to a custom, multinomial random distribution. The solution is approximately Pareto efficient which ensures the Hypervisor makes the most utility out of the available radio front-ends.

Following this introduction is a literature survey in Sec. II. A relationship is developed between channel interference and service disruptions in Sec. III. The Markov model is constructed in Sec. IV. The simulation setup is discussed in Sec. V. Results are discussed in Sec. VI, and the paper is concluded in Sec. VII.

2. LITERATURE SURVEY

The reconfigurable radio systems (RRS) set of standards [2] in development by ETSI for SDR systems, propose an app store model for SDR. In the app store model, a user purchases a wireless computer, such as mobile device, that is intrinsically void of wireless capability without software. This SDR mobile device would feature an array of generic radio front-ends. These front-ends may be suitable for multiple wireless protocols or may be specialized to a particular use case or wireless band. This wireless computer acquires wireless functionality once the user installs an SDR application from the app store.

Several existing works [4, 5, 28, 32] propose the abstraction layer connect SDR applications to radio front-ends dynamically on an as needed basis. In [20, 21], software-defined radios are said to be virtualized when they interact with shared radio front-ends through an abstraction layer as it is similar to the way virtual machines interact with a shared

processor. This type of radio front-end abstraction is even referred to as an SDR hypervisor. This approach has the advantage of making the number of concurrent SDRs less dependent on the number of front-ends.

In [5, 28, 32], radio front-ends were shared using time division multiplexing (TDM) of radio front-ends. Specifically these works used an on-demand fixed priority scheduler. When an SDR application initiated a transmission or reception event, the abstraction layer (i.e., hypervisor) would dynamically connect the SDR application to the first available front-end. When the event concluded, the front-end would become available again. If the supply of front-ends was exhausted, new transmission or reception events would be handled according to application priority. Thus, upcoming radio events from high-priority SDR applications may interrupt and preempt active low-priority radio events from low-priority SDR applications. Upcoming low-priority transmission and reception events may be dropped entirely if no front-end is available.

In our previous work [26], we worked to increase the capacity of virtualized SDR systems through the use of mixed-integer linear programming. We presented a linear programming model that could build an optimized front-end assignment schedule, given a finite list of transmission and reception events. This method showed improved capacity over the type of fixed priority scheduler in [32] and related works. However, since it could only optimize over a finite list of upcoming radio events, its assignment decisions could be somewhat myopic. For this reason and the required computational complexity, it could not be considered for use in real-time.

Like previous methods [5, 28, 32], our linear programming method [26] did not consider the impact sharing radio front-ends would have on the performance of wireless protocols. In [19], the impact of shared radio front-ends between Wi-Fi and the Long Term Evolution (LTE) cellular protocol was studied. This work focused mostly on multi-radio assignment trade-offs for MIMO systems and the impact of in-band interference.

We were drawn to a MDP after seeing a the behavior of wireless protocols modeled as Markov chains and MDPs. A number of works such as [7, 22] have modeled wireless retransmission schemes using a Markov model. This type of research was extended in works such as[23, 25] to model complex retransmission interactions between different wireless protocols. In section 5, we leverage the existence of such models for use in construction of a front-end utilization model. A number of these are simplified and composited to model the collective front-end utilization of the system. Thus, the collective front-end utilization model forms the state space for our system, and resouce allocation decisions form the actions space.

We arrived at the unique MDP decision policy developed in [8, 9] after a long search through MDP literature. Classical approaches to MDP include dynamic programming solutions (e.g., value iteration and policy iteration) and linear programming solutions [6, 27]. Our problem resembled a restless multi-armed bandit problem. Classic approaches include index policies such as those found in [14, 30]. Complications arose in adapting index techniques. The multi-objective nature of our problem and the K-out-of-M resource rationing nature of problem were the most problematic issues. Furthermore, our goal was to include performance guarantees for each objective. This lead to interest in works such as [13]. However, in our research, we found dynamic programming solutions came with a number of restrictions that we couldn't be sure our model could satisfy as it was still being developed. Additionally, we were concerned with long-term averages of radio metrics rather than discounted stochastic estimates. We found dynamic programming solutions to long-term average problems were a less developed area of research. This led us to linear programming solutions such as those found in [8, 9]. This solution met with our requirements for a decision policy concerned with the long-term averages of multi-objectives with constraints.

3. MODEL OF OPERATION

In this section, we describe the general operation of a multi-radio SDR system and the operation of the new elements proposed in this article. General operation is similar to that of RRS from [2] and HyDRA from [21]. New elements include an SDR workload feasibility test and an SDR application gating mechanism.

3.1. USER INTERACTION

Multi-radio SDR devices as described by [2] will install SDR apps like conventional mobile apps and operate them like convention digital radios. Users will be able to install and manage SDR applications using the same types of app stores and package management facilities already found in smartphones and mobile devices. Once installed, SDR applications can be toggled on and off like conventional digital radios already found in mobile devices. Traditional thinking would limit the number of SDR applications that the user may enable to the number of available front-ends. However, in a virtualized multi-radio SDR device such the system in [21, 26, 32] the user may continue to enable SDR applications since front-ends are shared among all applications. As the available front-ends are shared across more and more applications, radio performance will drop.

We propose the addition of an SDR application gating mechanism that works with the virtualization layer to limit the user to feasible SDR workloads only. In our proposed approach, when the user enables an additional SDR application, it is first placed into a proposed workload with all other enabled SDR applications. This proposed workload is tested to ensure all SDR apps meet their respective radio performance metrics. If the workload passes the feasibility test, a new control policy is generated and sent to virtualization layer. The additional SDR application is then launched. If the proposed

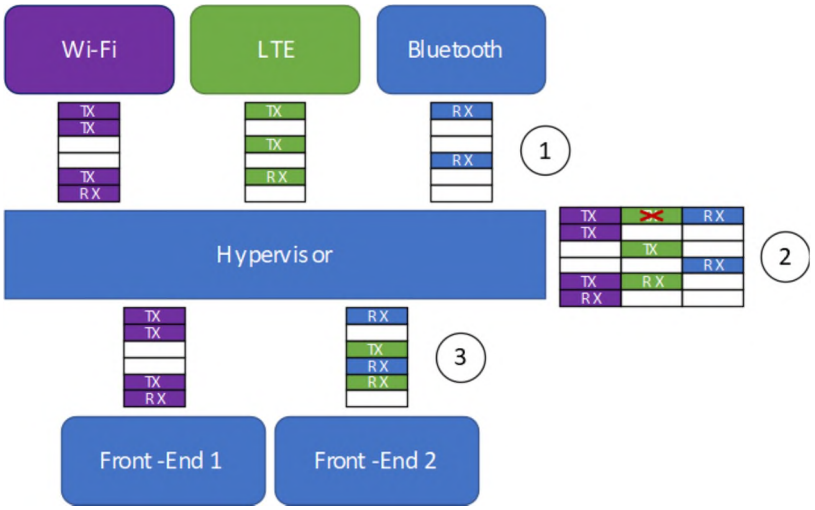


Figure 1. Overview of SDR hypervisor systems: (1) SDR applications (Wi-Fi, LTE, Bluetooth) issue spectrum requests to SDR hypervisor (2) SDR hypervisor converts requests into discrete time slot reservations and resolves scheduling conflicts (3) Sample streams are sent to/from front-ends

workload is infeasible, the user receives an error message and the SDR application remains disabled. Feasibility tests may be conducted in advanced for any combination of SDR applications installed the device. The results may be saved for future use.

3.2. GENERAL OPERATION

Each enabled SDR application is expected to generate and consume baseband sample streams, and in turn, the applications expect the hypervisor (i.e., virtualization layer) to route these streams to and from front-ends. Existing works [28, 32] describe an application programming interface (API) and a model of operation where discrete transmission and reception events are encapsulated into *spectrum access requests*. In this API, these spectrum access requests are containers for baseband sample streams and descriptive meta data. SDR applications issue spectrum requests to the SDR Hypervisor as illustrated in Figure 1 (1). Spectrum request meta data includes details such as carrier frequency, bandwidth, sampling rate, timing reference, etc. As the spectrum requests arrive from SDR applications, the

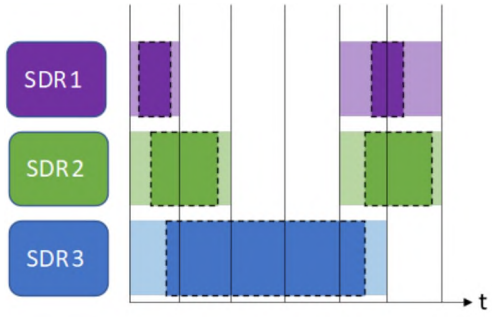


Figure 2. SDR spectrum access requests translate into discrete time slot reservations.

hypervisor buffers them and builds a schedule illustrated in Figure 1 (2). At the appropriate time, the hypervisor services or drops each request Figure 1 (3). To service a transmission request, the hypervisor delivers the sample stream contained in the request to an available front-end transmitter where it is synthesized into an actual radio waveform and sent. To service a reception request, the hypervisor retrieves a sample stream generated from an available front-end receiver and delivers it to the corresponding SDR application. Each front-end can service a single request at a time.

In this article, we modify the spectrum request model to make it more convenient for use with MDP. Our hypervisor uses a discrete-slotted-time approach. Each radio front-end is assigned to an SDR application for a fixed interval or time slot. Assignment decisions are made just prior to the start of each interval and are held constant for the entire duration. Timing is synchronized across all front-ends. Thus, an assignment decision is made for all front-ends at each interval. Spectrum requests may span multiple time slots. Request duration will need be rounded up to the nearest whole time-slot to avoid cutting off a request. This rounding is illustrated in 2. Notice, that both of SDR 1's requests are the same duration, but they require different numbers of slots depending on where they align in time. Though a spectrum request may be known to span multiple time slots, assignments are never made more than a single time slot into future.

4. SER ESTIMATION

Wireless protocols often use sensitivity and frame error ratio (FER) as standards for physical layer receiver performance. Sensitivity refers to the minimum signal power for which a receiver should properly decode a signal. FER is the ratio of corrupted data frames (i.e., packets) to total number of data frames. A receiver is expected to decode a signal with a specified minimum signal power (i.e., the sensitivity) with an FER less than or equal to a specified maximum FER.

There is a direct relationship between the scheduling decisions made by our SDR hypervisor and perceived signal-to-noise ratio (SNR). When the SDR hypervisor consistently provides a front-end to an SDR application, the hypervisor and front-end sharing is essentially transparent. Thus, the channel has the greatest effect on communications. If the SDR hypervisor fails to make an assignment during a time slot, it is as if there is no signal power present and only noise. The resultant SNR could be called 0 or $-\infty$ dB for the duration of any missed assignment. Additionally, the selection of front-end may influence SNR. Front-ends contribute noise and interference in addition to the environmental noise. Different makes and models of front-ends contribute different amounts of noise and interference. Therefore, the SDR hypervisor must also consider which front-end is assigned when the available front-ends are not identical.

The SDR hypervisor can influence FER indirectly with its front-end assignments. Since FER is a function of the SNR, the SDR hypervisor can use its per-time slot SNR influence to manipulate the overall FER. As previously stated, a frame may span a number of time slots and control decisions are made every slot rather than every frame. This makes direct control over FER difficult. However, it is possible to directly manipulate the per frame symbol error ratio (SER) as a way to control the FER. For a known modulation and channel model, there are established equations to relate SNR to SER [15]. These equations are typically given as a function of normalized carrier-to-noise ratio $\frac{E_s}{N_0}$. Normalized carrier-

to-noise ratio can be equated to SNR,

$$\frac{E_s}{N_0} = \frac{S}{N}\left(\frac{B}{R}\right),$$

where R is the baud rate (i.e., symbol rate) and B is the signal bandwidth. By managing the mean SER, the SDR hypervisor can indirectly control the overall FER. If we assume the SNR is constant for the duration of a time slot, then mean SER per time slot SER_μ can be defined as

$$SER_\mu = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} SER_i,$$

for n discrete time slots.

Details from a wireless specification can be used to determine the maximum overall SER allowable to maintain the specified FER for a wireless standard. We assume there is no more than a single frame per-time slot and that conditions are constant for the duration of the time slot. If a frame is composed of L symbols and the symbols are assumed independent, a maximum acceptable SER (SER_{max}) can be determined,

$$FER \approx 1 - (1 - P_s)^L,$$

where P_s is the probability of a symbol error. Then

$$SER_{max} \approx P_s \approx 1 - (1 - FER_{max})^{1/L}.$$

From this calculation, we can determine the SER_{max} for any of the hosted SDR applications.

Rather than measure the SNR for each time slot, a worst case scenario SNR can be assumed. It would be difficult for a hypervisor to measure SNR for each hosted SDR, since this calculation is unique to each wireless protocol and modulation. A simple solution is to assume a worst-case SNR for each delivered time slot. Again, an SNR of 0 could be assumed when no assignment is made. Worst case SNR is when signal power is weakest

and noise power is strongest. Sensitivity can be used for minimum signal power P_{min}, since it is effectively worst case signal power. If the signal power is lower than the sensitivity, it is outside of the specification and is not expected to be decoded. Worst case noise power N_{max} of a wireless receiver (i.e., the noise floor N_{floor}) can be estimated as a function of intrinsic properties of a wireless front-end. Given N_{max}, the delivered SNR for a particular time slot can be estimated (SNR_{est}) as

$$SNR_{est} = \frac{P_{min}}{N_{max}}.$$

Several factors play a role in the calculation of the noise floor N_{floor}. The major contributor of noise, thermal noise, is typically calculated as

$$N = KTB,$$

where

$$K = \text{Boltzmann's constant} = 1.38 \times 10^{-23} \frac{W}{K - Hz}$$

$$T = \text{temperature in kelvin, typically } 290K$$

$$B = \text{bandwidth in } Hz.$$

Noise bandwidth is usually the same as the signal bandwidth. Additional noise will be added by circuit components in the receiver's wireless front-end. The noise contribution of a circuit component is normally given as a ratio of the device's output noise power to the device's input noise power. This is otherwise known as *noise factor F* or *noise figure NF* when given in dB. A composite noise factor F_{comp} can be calculated for all circuit components in a wireless front-end using the Friis formula for noise:

$$F_{comp} = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + \cdots + \frac{F_n - 1}{G_1 G_2 \cdots G_{n-1}}.$$

Here F_i and G_i refer to the noise factor and gain of a circuit component i respectively. The sampling noise of the analog-to-digital converter (ADC) also contributes to the composite noise factor of a front-end and should be included in F_{comp} as well. The F_{comp} is usually given on datasheets for off-the-shelf wireless front-ends. For a wireless front-end given bandwidth B, noise floor N_{floor} can be calculated,

$$N_{floor} = (KTB)F_{comp}.$$

Alternatively, distortion calculations may be used in place of noise figure. Non-linear distortion such as harmonic distortion contributes significant interference in electronics. This is usually described as signal-to-noise-and-distortion ratio (SINAD) [17]

$$SINAD = \frac{S}{N + D}.$$

SINAD appears on data sheets for many electronics such as ADCs and front-ends.

Interference tolerance is often listed in a wireless specification and should also be considered in SNR estimation. Interference tolerance is expressed as a maximum average interference power I. This value may be significantly larger than the N_{floor}. In such cases, either I may be substituted for N_{floor} or signal-to-interference-plus-noise ratio (SINR),

$$SINR = \frac{S}{I + N},$$

may be used in place of SNR.

The previous overall signal quality concerns can be applied to transmitters as well. Transmitter SNR requirements are usually more vaguely outlined in a wireless specification than receiver SNR. The topic of physical layer transmitter signal quality is usually dominated by concerns about non-linear distortion. Non-linear amplifiers used in modern wireless transmitters generate large amounts of spurious emissions and distortion in the form of

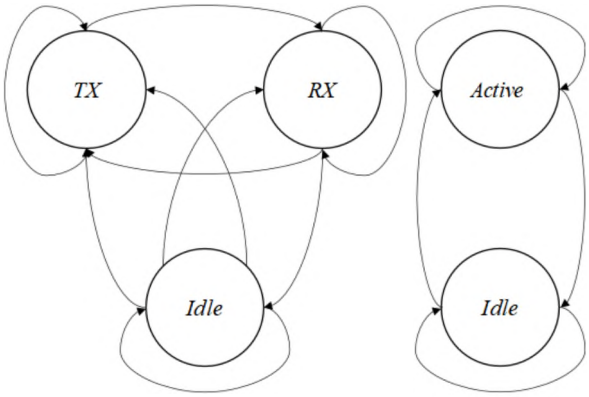
intermodulation and harmonics. Wireless specifications and government regulatory bodies usually set bounds for total transmitter emissions in the form of a power spectral density mask. Wireless implementations must keep their emissions within these bounds to be certified.

Non-linear distortion effects can be mitigated using amplifier linearization techniques [3, 18, 29, 31]. Ideally, amplifier linearization should be a standard responsibility of all multi-radio SDR systems. All transmissions from all SDR applications would, therefore, be linearlized. Receiver distortion can be mitigated by linearzation techniques as well. Non-linear distortion will not be considered as a part of the control scheme presented in this article.

5. MDP

The SDR hypervisor balances the overall utility with respective SER requirements. In the previous section, it was established how the hypervisor's assignment decisions impact overall SER and why this indirectly impacts FER. In this section, we describe a MDP based decision making strategy. This strategy is developed *a priori* using the predicted SNR/SER methods from the previous section. This MDP works by considering the SER requirements of SDR applications and the impact each front-end assignment will have on the overall SER.

We now formulate the dynamic front-end assignment scheme using a MDP. A MDP consists of multiple elements: decision epochs, states, actions, transition probabilities, rewards, and an objective. The hypervisor constantly collects spectrum requests from SDR applications and translates them into corresponding discrete, fixed-duration time slots. Assignment decisions are made just prior to the start of a time slot. The sequence $T = \{1, 2, ...\}$ denotes successive discrete time slots and with them decision epochs. A wireless device may operate for days or weeks on end. Thus, the decision making process



(a) 3-State Model (b) 2-State Simplified Model

Figure 3. State transition diagram for a single MDP sub-process (SDR application). (a) shows the 3-state model. (b) shows the simplified 2-state model.

considers an infinite horizon. Front-end assignments are constant for the entire duration of a slot. Assignments are mutually exclusive, meaning a front-end is assigned to a single SDR application during a time slot.

5.1. STATE SPACE

An SDR application requests access to a front-end depending on its state. The decision to transmit or receive is determined by the wireless protocol. It is common for wireless protocols to base the decision to transmit or receive on the success or failure of the previous transmission or reception. The probability of success or failure of a transmission or reception is influenced by channel conditions such as noise and overall channel busyness. Given fixed channel conditions, a highly detailed Markov model can be built for a wireless protocol [25]. Front-end assignment directly influences the amount of perceived noise by an SDR application. Therefore, front-end assignment influences the state transition probability function and forms the action space.

The states in this MDP can be reduced into the superstates *transmit* and *receive*. States in the wireless protocol that require access to a front-end to transmit can be reduced into the *transmit* superstate. States in the wireless protocol that require access to front-end to receive can be reduced into the *receive* superstate. States that do not require a wireless front-end can be grouped into a superstate that we label as *idle*. This 3-state model can be seen in 3a. Since transmission and reception equally require access to a front-end, the 3-state model can be further reduced by grouping the transmit and receive states into a superstate that we label as *active* as seen in 3b. The reduction in states may reduced the model's accuracy, but will reduce the size of the state space. The state space for the i^{th} application is therefore defined as

$$S_i = \{idle, active\}.$$

The state transition probability function for the i^{th} SDR application

$$P =: S \times A \times S \rightarrow [0,1]$$

denotes the probability $P(s_i'|s_i, a)$ where $s_i, s_i' \in S_i$. This simplified MDP is illustrated in 3. State transition probability matrix for some action is defined as

$$P_{|A} = \begin{pmatrix} p_{ii} & p_{ia} \\ p_{ai} & p_{aa} \end{pmatrix},$$

where $p_{ii} + p_{ia} = 1$ and $p_{ai} + p_{aa} = 1$.

5.2. ACTION SPACE

The action space A is formed by all possible permutations of front-end assignment in all possible states. The action space for a MDP is labeled as $A = \{a_1, a_2, ..., a_n\}$. Each action represents one possible assignment permutation. Front-end assignments last exactly one time slot and are mutually exclusive. An SDR application may be assigned to, at most, one front-end. Likewise, each front-end may be assigned to, at most, one SDR application. An SDR application may also go unassigned. If there are M SDR applications and N front-ends, then, at most, N out of M SDR applications can operate per-time slot. Fewer than N SDR applications may also be assigned when there are idle SDR applications. The action space for two front-ends and two back-ends are depicted in 4. We follow the convention in [9] and assume that each action is enabled in exactly one state. In our model, generally all assignments are possible in all states making size of the action space $A : S \times$ All Assignments. A function $Act(s)$ where $s \in S$ determines which actions are enabled in a given state s. Nonsensical assignments may be removed to reduce the size of the action space. For example, an SDR application should never be assigned to a front-end that cannot tune to the application's requested carrier frequency nor to a front-end with insufficient bandwidth.

Together the action space and the state space created for each SDR application form a set of weakly coupled MDPs. Generally speaking, a weakly (loosely in some literature) coupled MDP consists of a set of independent MDPs bound by constraints on the action space [12]. In our model, the constraints refer to the mutually exclusive assignments. The state spaces of S_i of each SDR application are assumed independent. The composite state space is defined as $\boldsymbol{S} = (S_1, S_2, ..., S_m)$ with dimension $S_1 \times S_2 \times ... \times S_m$. The state transition probability function is defined as

$$\boldsymbol{P} = [\boldsymbol{P}^m] =: \boldsymbol{S} \times A \times \boldsymbol{S} \rightarrow [0, 1],$$

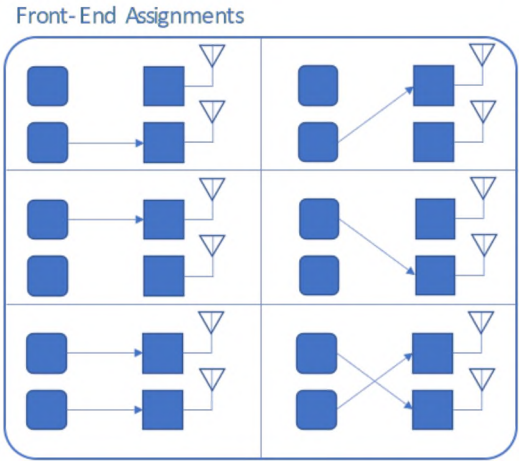


Figure 4. The two SDR applications (i.e., back-ends) on the left are connected to two front-ends transceivers on the right in one of several possible permutations. This forms the action space of the MDP.

where the state transition probabilities take on the values

$$\boldsymbol{P}(\boldsymbol{s}'|\boldsymbol{s},a) = P_1(s_1'|s_1,a)P_2(s_2'|s_2,a)...P_m(s_m'|s_m,a),$$

with $\boldsymbol{s}', \boldsymbol{s} \in \boldsymbol{S}$.

5.3. REWARD FUNCTION

The reward function R for each Markov sub-process (i.e. SDR application) is calculated in terms of SER. A reward function $R(a,s)$ is typically defined as a function of state and action; however, each action is only available in a single state as per [9] the state may be dropped as an argument $R(a)$. In this formulation, a reward is paid for performing an action in the current state. Rewards are paid in terms of estimated SER. Based on the previous discussion, an SER estimate can be made for each pairing of front-end and SDR application. Each SDR application forms an independent Markov sub-process with its own reward function that counts toward their respective overall SER calculation. Rewards are therefore returned as a vector. The overall reward function is defined $\boldsymbol{R} = [\boldsymbol{R}^m] =: A \rightarrow \mathbb{R}^m$.

Since overall SER is calculated as an average, we adopt an average expected total reward function. The objective of each Markov sub-process is to minimize their respective SER values. The expected total reward function is given in terms of the limit-average function $lr(\boldsymbol{R}) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{R}(A_t)$. Since we wish to minimize SER, our objective is to minimize limit superior

$$lr_{sup}(\boldsymbol{R}) = \lim_{T\to\infty} sup \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{R}(A_t)$$

as in [8, 9].

5.4. DECISION POLICY

In a MDP the controller must make a decision to perform an action at each time epoch according a to decision policy made up of decision rules. A decision rule is a function that determines what action the controller should perform. There is a decision rule for each state. A decision policy is the set of all decision rules. Several different types of decision rule may be used by a controller in a MDP. In this article, we adopt a policy developed in [8, 9]. By popular definition [27] all decision rules used in this strategy can all be said to be *randomized*. The policy maintains a memory that determines what decision rule should be used at each state. In [9], a policy σ is said to consist of three elements $\sigma = (\sigma_u, \sigma_n, \alpha)$.

- σ_n is a function that determines the next action to be taken $\sigma_n : S \times M \to Dist(A)$, and could be called a decision rule by [27]'s definition.
- σ_u is a randomized function that updates the state of the memory $\sigma_u : A \times S \times M \to Dist(M)$.
- α is a function that determines the initial memory state.

The policy σ switches between two memory states. The decision rule σ_n is a function of the memory state and therefore there are two possible decision rules for each state in the MDP. In [9], multiple implementations of the decision rules are given. Of these, we adopted a randomized decision rule for both policy memory states. The policy's initial memory state determined by α is a strictly a randomized function.

The decision policy in [8, 9] is formed by using the concept of maximal end components (MEC). An end component is a union of a set of actions B and a set of states T where (1) $P(s'|s,a) > 0$ then $s', s \in T$ and $a \in B$ and (2) there is a sequence of actions and states (i.e., a path) connecting any state in T to any other state in T via a sequence of actions and states that are also in the end component[9]. An end component is maximal when it is not fully or partially contained in another end component. The elements within an MEC are said to be strongly connected. The set of all MECs in an MDP are denoted $\boldsymbol{MEC}$. We used a graph decomposition technique from [10] to break down the MDP into MECs.

The decision policy in [9] operates in two phases. In the initial phase, the policy navigates from one MEC to another with a specified probability of being in each MEC at any given time. This probability is controlled via a set of decision rules unique to phase one. Eventually, the policy transitions to the second phase based on σ_u. In the second phase, the policy navigates such that it remains in the current MEC forever. It visits the states in the final MEC with a specified probability. This probability is also controlled via a set of decision rules that are unique to phase two. The probability of performing any action or being in any state in phase two is then the probability of ending up in the relevant MEC after phase one and the probability of performing relevant actions given the phase two decision rules.

5.5. REWARD EXPECTATIONS & GUARANTEES

The decision rules in [4, 8] ensure certain long-run reward expectations via constraint. In [9], an expectation constraint $\boldsymbol{exp}$ is set for each objective. It ensures that the average long-run reward of many runs is at least exp_i for the i^{th} objective respectively. That is to say, that the long-run reward vector of a single run of the policy may fall bellow the constraint vector $\boldsymbol{exp}$. However, if the results of several runs are averaged, the value should be greater than or equal to $\boldsymbol{exp}$. The MEC arrived at in phase 2 is the largest influence on the average long-run reward. In [8], an additional constraint vector called a satisfaction constraint $\boldsymbol{sat}$ was defined for each objective. This constraint ensures the long-run reward for the i^{th} objective is greater than or equal to sat_i for all MECs and, therefore, all runs of the policy. We use both the $\boldsymbol{exp}$ and $\boldsymbol{sat}$ constraints to ensure that mean SER for each SDR application is within specification.

5.6. FEASIBILITY

In [4], the decision policy was constructed using linear programming. The linear program contains variables y_a, y_s, x_a.

- y_a is the expected number of times an action is taken until phase two is reached.
- y_s is the probability of transitioning to phase two upon reaching some state s.
- x_a is expected frequency of performing action a.

To test the feasibility of an SDR workload, construct the MDP model developed in the previous section using application specific values. Solve the following set of linear programming constraints: equation 2-7. An SDR workload is feasibly if there exists a positive value solution vector for all variables.

$$\min \sum_{1 \le i \le k} w_i \sum_{a \in A} x_a \cdot r_i(a) \tag{1}$$

$$\mathbf{1}_{s_0}(s) + \sum_{a \in A} y_a \cdot \delta(a)(s) = y_s + \sum_{a \in Act(s)} y_a \qquad \forall s \in S \tag{2}$$

$$\sum_{s \in C \in \boldsymbol{MEC}} y_s = 1 \tag{3}$$

$$\sum_{s \in C} y_s = \sum_{a \in C} x_a \qquad \forall C \in \boldsymbol{MEC} \tag{4}$$

$$\sum_{a \in A} x_a \cdot \delta(a)(s) = \sum_{a \in Act(s)} x_a \qquad \forall s \in S \tag{5}$$

$$\sum_{a \in A} x_a \cdot r_i < exp_i \qquad \forall 1 \leq i \leq k \tag{6}$$

$$\sum_{a \in C} x_a \cdot r_i(a) < \sum_{a \in C} x_a \cdot sat_i \qquad \forall C \in \boldsymbol{MEC}, \quad \forall 1 \leq i \leq k \tag{7}$$

$$x_a, y_a \geq 0 \qquad \forall a$$

$$0 \leq y_s \leq 1 \qquad \forall s$$

Equation 2 describes the transition probabilities in phase 1. The left-hand equation is the expected number of times a state is entered while the right is expected number of times the state is exited or phase 2 is reached y_s. The function $\mathbf{1}_{s_0}(s)$ returns 1 if s is the initial state. Starting in a state is the same as entering a state, so it is included. Equation 3 ensures the total probability of switching to phase 2 across all states in all MECs totals 1. Equation 4 makes the total probability of switching to phase 2 in an MEC (left-hand side) equal to the total frequency of using actions within that MEC. This equation is necessary to relate y_s and x_a variables. Equation 5 guarantees the conservation of flow when in phase 2. In other words, the probability of entering a state (left-hand side) must be equal to the probability of leaving a state (right-hand side). Equation 6 ensures the expectation goals. The left-hand side is an expectation of the i^{th} reward across many runs while the right is the constraint [9]. Equation 7 is an added constraint that ensures the satisfaction goals are met regardless of the MEC that the strategy commits to in phase 2 [8].

Additional modifications were made to Equation 7 for our specific application. In our application, $\boldsymbol{sat}$ is the SER_{max}. The left-hand side of the Equation calculates the expected total reward by weighting the rewards against the frequency of performing each action x_a in the respective MEC. There is a constraint for each MEC in the MDP to ensure $\boldsymbol{sat}$ are met for any outcome. The right-hand side of Equation 7 is also weighted against the frequency of actions x_a in the respective MEC. Both the left-hand and right-hand sides consider the frequency of actions x_a tied to idle states. We removed these states as actions taken in idle states do not affect the expectation. Actions tied to active states $Active_i$ for the i^{th} reward are used to reduce the scope of Equation 7.

$$\sum_{a \in C \cap Active_i} x_a \cdot r_i(a) < \sum_{a \in C \cap Active_i} x_a \cdot SER_{max_i} \forall C \in \boldsymbol{MEC}, 1 \leq i \leq k$$

This has the effect of tightening the $\boldsymbol{sat}$ bound in our application.

5.7. PARETO OPTIMALITY

The objective, Equation 1, was suggested in Lemma 4.6 of [9] to ensure Pareto optimality. The reward expectation for objective in this construction is

$$\mathbb{E}\left[lr_{sup}(r_i)\right] = \sum_{a \in A} r_i(a) \cdot x_a.$$

To produce a Pareto optimal solution:

1. Add the Equation 1 to the linear constraints Equations 2-7 to form a linear programming problem.

2. Solve for a positive solution, and produce a reward expectation vector $\vec{v}$.

3. Confirm $\vec{v}$ is Pareto optimal by substituting it for $\boldsymbol{sat}$ and $\boldsymbol{exp}$ and solve again. Ensure new reward expectation is on better than $\vec{v}$.

We added the weights w_i where $\sum_{i=1}^{k} w_i = 1$ to allow an implementation to pick different solutions from the Pareto curve. All solutions are guaranteed to meet SER requirements (i.e., the satisfaction and expectations constraint), but the weights allow any excess reward potential to be directed toward a chosen objective. Effectively, it allows an implementer to add a static priority.

5.8. DECISION RULE

The solution variables to the linear program problem can be used to form a 2-stage randomized decision policy. When the policy is in phase 1, the memory $m = m_1$, when the policy is phase 2 the memory $m = m_2$. Next action behavior is determined by a multivariate random distribution constructed using x_a and y_a [9]:

$$\sigma_n(s, m_1)(a) = \frac{y_a}{\sum_{a' \in Act(s)} y_{a'}},$$

$$\sigma_n(s, m_2)(a) = \frac{x_a}{\sum_{a' \in Act(s)} x_{a'}}.$$

The memory update strategy is stochastic and determined by the variable y_s. The decision policy transitions to phase 2 in state s with probability y_s. Once in phase 2, the policy stays in phase 2 with probability 1 [9].

$$\sigma_u(a, s, m_1)(m_2) = y_s$$

$$\sigma_u(a, s, m_2)(m_2) = 1$$

The initial memory state is determined by function alpha. It is defined by y_{s_0} that is, the probability of transitioning to phase 2 in the initial state s_0 [9].

$$\alpha(a, s, m_1) = (1 - y_{s_0})$$

$$\alpha(a, s, m_2) = y_{s_0}$$

5.9. APPLICATION NOTES AND OPTIMIZATION

Our MDP often has only a single MEC which leaves room for some optimizations. The simplified 2-state state space used for each sub-process (i.e., SDR application) with only *active* and *idle* states can transition between both states unless a state transition probability is zero for some action. As the states are grouped into *idle* or *active* the transition probabilities are grouped by means of weighted sums. Recall the sub-processes are independent of each other. If for all sub-processes, there are no inaccessible states, then no state in the composite state space $\boldsymbol{S}$ is inaccessible $P(\boldsymbol{s}'|\boldsymbol{s}, a) \neq 0$. Thus, there is only a single MEC. With only a single MEC, phase 1 of the decision policy may not be necessary. Therefore the linear programming equation could be simplified. For instance, in the single MEC case, the expectation constraint serves the same purpose as the satisfaction constraint. Of course, many wireless protocols never go to an idle state (i.e. $P(idle|s, a) = 0$ for $s \in S$) which could result in multiple MECs even with the simplified 2-state state space. Additionally, when more detailed state-spaces are used for the each sub-process the possibility of an inaccessible state increases.

6. SIMULATION AND RESULTS

In this section, we conduct simulations of the proposed SDR control system and discuss the results. We present two example scenarios representative of a simple multi-radio SDR system. This simulation used the simplified 2-state MDP model. Parameters were chosen arbitrarily to produce graphically interesting results that highlight different aspects of operation.

Table 1. Parameters of the SDR Applications

Notation	Parameter Definition	Value
	Modulation	M-PSK
M-ary	Constellation Points	4
Bd	Baud Rate	10e6
B	Bandwidth	20 MHz
FER_{max}	Maximum Frame Error Ratio	0.1
L	Frame Length	8192 Symbols
P_{min}	Sensitivity	-80 dBm
SER_{max}	Maximum Symbol Error Ratio	1.2861e-05

Table 2. Stochastic Parameters

Stochastic Parameter	Scenario 1	Scenario 2
$P_{m \in M}(idle \vert idle, a = Any)$	0.95	0.95
$P_{m \in M}(active \vert active, a = \emptyset)$	0.05	0.05
$P_{m \in M}(active \vert active, a = 1 : K)$	0.3	0.5

6.1. SIMULATION SETUP

In this simulation, we model a simple scenario typical of a smartphone or mobile device. The device features 5 SDR applications and with 4 front-ends. The 4 available front-ends each have a different noise figures as noted in Table 3. The first front-end has a 4 dBm noise figure. The remaining front-ends' noise figures worsen by 3 dB sequentially. These values were chosen to highlight the proposed MDP's ability to balance interference across all SDR applications. For simplicity, each SDR application has identical radio and stochastic parameters which can be seen in Table 1 and Table 2 respectively. The applications behave according to the 2-state Markov model from previous sections. Each scenario is tested for four million time epochs. For this simulation, we assumed no dependency between time slots. Each time slot, therefore, represents a complete transmission or reception event.

Table 3. Simulation Parameters

Notation	Parameter Definition	Value
M	No. of Applications	5
K	No. of Front-Ends	4
NF	Noise Figure	4,7,10,13 dBm

Each application is modeled as a simple 4-PSK modulation radio on a Gaussian white noise channel. The sensitivity is set at -80 dBm, with a channel bandwidth of 20 MHz and a frame length of 8192 symbols. These parameters where derived from the simpler modes in the IEEE 802.11 specification [1]. The SER_{max} was calculated based on equations from previous sections.

The 2-state Markov model transition parameters were chosen to model an arbitrary radio standard are listed in Table 2. It is assumed that assignment actions taken while an SDR application is idle will have no influence on the probability that the application will move into the active state. This is reflected in the idle-to-idle transition probability. Further, it is assumed that assignment actions taken while an SDR application is active will have the most influence on its behavior. In our arbitrarily radio standard, if no assignment is made (i.e. $a = \emptyset$) while an SDR application is active, it is assumed that the application will register a failure and return to the idle state with high probability. Likewise, when an assignment is made while the application is in an active state, it is assumed the application will remain in the active state with a high probability. For simplicity, all radio assignments are assumed to have the same influence on behavior despite their different noise figures. The difference between the two scenarios is the active-to-active transition probability.

For both simulation scenarios, we test three control algorithms. While the SDR applications are identical, they are numbered in a descending order of priority. The first algorithm is simple static assignment. This means that each application is connected to a front-end for the duration of the simulation. Assignments are made once in order of priority until the front-ends are exhausted. Additionally, the quality of the front-end assigned is

determined by application priority. The static assignment algorithm results in the first application being assigned the best front-end and the last application not being assigned a front-end. The second algorithm is a static priority system similar to the one in [5]. Assignments are made at each decision epoch according to which applications are active and application priority. The quality of the front-end assigned is determined by application priority. The final algorithm is the proposed MDP which makes assignments stochastically based on long-term expected rewards.

6.2. RESULTS AND DISCUSSION

In both scenarios, we examine three test results. Most importantly, the SER of each SDR application is measured against the SER_{max}. This is the mean SER measurement per time slot. For each application, we also measure the mean SNR per time slot as well as the satisfaction ratio. It is important to note that the mean SNR per slot has some influence over the SER, but it a weak relationship. The satisfaction ratio is the number of served time slots divided by the number of requested time slots. Testing showed satisfaction ratio had the highest influence over SER.

6.3. SCENARIO 1

The SER results of the first scenario can be seen in Figure 5. The static assignment algorithm showed increasing SER with decreasing priority of application. The first application was highly over-served with the lowest noise front-end while SDR_5 was not served at all. Despite having a dedicated front-end, SDR_4 slightly exceeded the SER_{max} threshold due to the radio parameters chosen. This shows that SDR applications can exceed their SER_{max} thresholds even with a dedicated front-end. Figure 6 shows SNR performance per time slot. As expected SNR is proportional to application priority. Blindly mixing and matching SDR front-ends and back-ends can lead to substandard performance.

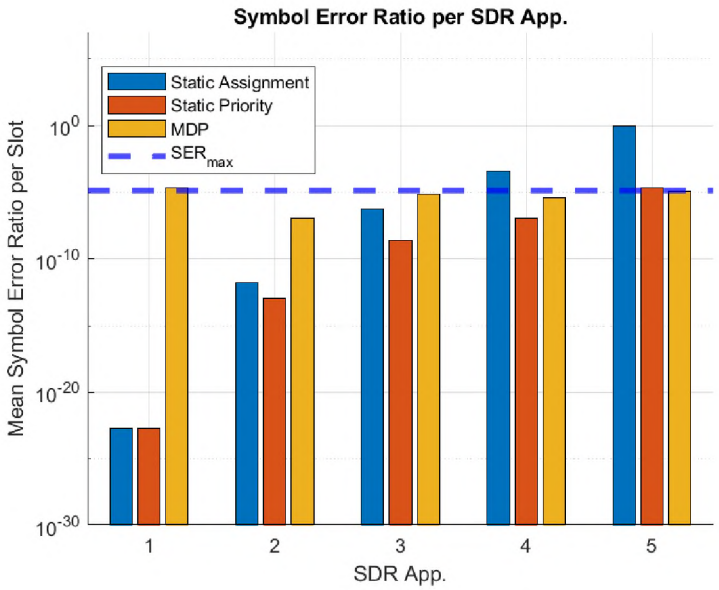


Figure 5. Mean SER per slot for simulation 1.

Table 4. Satisfaction Ratio : Simulation 1

Method	SDR_1	SDR_2	SDR_3	SDR_4	SDR_5
Static Assignment	1	1	1	1	0
Static Priority	1	1	1	1	0.99998
MDP	0.99998	1	0.99999	1	0.99999

The static priority system fared substantially better than static assignment algorithm. Like the static assignment algorithm, higher priority apps had better SER. However unlike the previous results, all applications were well served. SDR SDR_5 slightly exceeded the SER_{max} threshold, but this difference was within the margin of error of the simulation. Table 4 shows SDR_5 was slightly under-served likely leading to higher SER value. Figure 8 shows extremely uniform SNR. With this algorithm, lower priority applications will get access to higher quality front-ends. This helps to even out performance. The major drawback to this approach is that it is simply a best effort algorithm with no performance guarantee. Though the per time slot SNR seems to indicate good performance, SDR_5 nearly fell above the threshold.

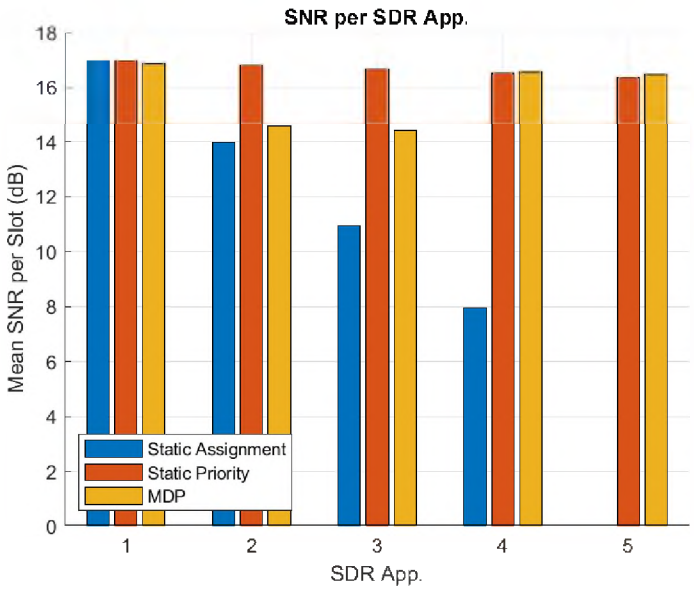


Figure 6. Mean SNR per slot for simulation 1.

The proposed MDP algorithm resulted in extremely uniform SER performance. Most applications fell below the SER_{max} threshold with the exception of SDR_1. This difference was within the margin of error of the simulation. Overall SER was higher than the static priority algorithm; however, no static priority was set for the MDP. This is what likely resulted such uniform performance. SER performance of SDR_1, SDR_3, and SDR_5 were slightly higher than the rest. This corresponds to their slightly reduced satisfaction ratio seen in Table 4. SNR values were very close to that of the static priority algorithm with the exception of SDR_3 and SDR_4. SER performance was expected to be slightly better given that the solution was approximate Pareto optimal. However, a slight drop in performance is likely acceptable given the SER_{max} performance guarantee.

Table 5. Satisfaction Ratio : Simulation 2

Method	SDR_1	SDR_2	SDR_3	SDR_4	SDR_5
Static Assignment	1	1	1	1	0
Static Priority	1	1	1	1	0.99994
MDP	0.99999	0.99998	0.99997	0.99998	0.99999

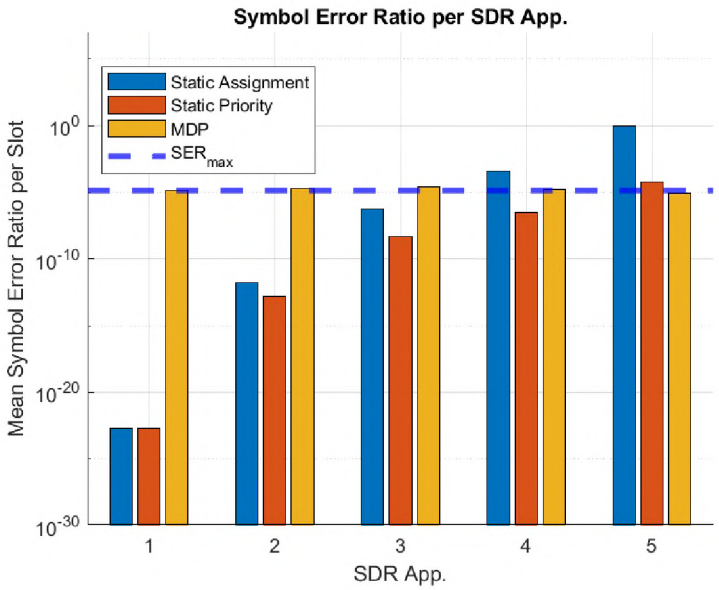


Figure 7. Mean SER per slot for simulation 2.

6.4. SCENARIO 2

In the second scenario, the active-to-active probability was increased resulting in increased demand for front-end time slots. Figure 7 shows slightly exaggerated results when compared to the first scenario. The static assignment algorithm's results were identical. The static priority algorithm's results are very similar, but this time SDR_5 exceeded SER_{max} by a larger margin. The proposed MDP algorithm had SER scores that were all below SER_{max}, but with less room to spare. Table 5 demonstrates the MDP algorithm distributing serviced disruptions relatively evenly across the SDR applications unlike the static priority algorithm. SNR performance is shown in 8. Interestingly, the static priority algorithm's SNR scores decay slightly with application priority. This trend was not apparent in the previous scenario. Given more SDR applications, or fewer front-ends this trend would likely be exaggerated. Like the previous scenario, the proposed MDP algorithm shows relatively uniform SNR performance. This time 4 out of the 5 applications showed a reduced SNR.

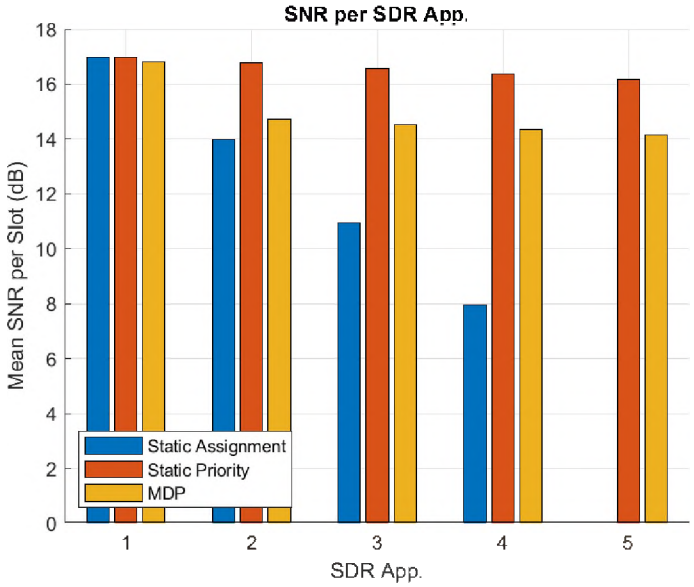


Figure 8. Mean SNR per slot for simulation 2.

7. CONCLUSION

In this article we presented a MDP based control strategy for multi-radio, multi-protocol SDR systems. This system provided a means to time-share a limited number front-ends among several concurrent SDR applications. Importantly, it guaranteed SNR be maintained for each SDR application's unique requirements. We established a relationship between TDM time-slots and SNR. Through this relationship, we showed that by maintaining satisfaction ratio, we can maintain a long term SNR conducive with an SDR application's requirements. We modeled this using a weakly coupled MDP and solved it using a 2-memory state randomized strategy solution method and decision strategy [8].

We demonstrated the complete implementation with a simulation that showed excellent results. All SDRs maintained a long-run average SER lower their respective requirements. We observed from the results that satisfaction ratio had the greatest influence on SER performance. The traditional static priority system showed excellent SNR results exceeding those of the proposed MDP; however, it had a tendency to under serve the lowest

priority application. This lead to lowest priority SDR application exceeding acceptable SER levels in the second simulation. By contrast, the proposed MDP evenly distributed serviced disruptions resulting in all applications operating within the SER threshold.

Future work can improve many aspects of the proposed solution. It could be improved with MDP techniques that require fewer *a priori* statistics such as Q-learning. In the simulation we assumed that each time slot was independent; however, this is not likely to be the case in a practical application. It would worth exploring extensions of the state space that consider radio events that span multiple time slots. This may also lead to more direct control of FER.

ACKNOWLEDGMENTS

The authors would like to express their gratitude to Dr. Kurt Kosbar (Missouri University of Science and Technology) for his feedback and guidance.

REFERENCES

[1] 'IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,' 2016, doi:10.1109/IEEESTD.2016.7786995.

[2] 'Draft ETSI EN 303 648 V1.1.2: Reconfigurable Radio Systems (RRS); Radio Equipment (RE) reconfiguration architecture,' 2020.

[3] Abdelaziz, M., Anttila, L., Tarver, C., Li, K., Cavallaro, J. R., and Valkama, M., 'Low-Complexity Subband Digital Predistortion for Spurious Emission Suppression in Noncontiguous Spectrum Access,' IEEE Transactions on Microwave Theory and Techniques, 2016, **64**(11), pp. 3501–3517, ISSN 0018-9480, doi:10.1109/TMTT.2016.2602208.

[4] Ahtiainen, A., Berg, H., and Westmeijer, J., 'Architecting Software Radio,' in 'Proceeding of the SDR '07' Technical Conference,' 2007 .

[5] Ahtiainen, A., van Berkel, K., Kampen, D. V., Moreira, O., Piipponen, A., and Zetterman, T., 'Multiradio Scheduling and Resource Sharing on a Software Defined Radio Computing Platform,' in 'Proceedings of the SDR '08 Technical Conference,' 2008 .

[6] Bertsekas, D. P., *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont, Massachusetts, 4th edition, 2012, ISBN 978-1-886529-44-1.

[7] Boujemâa, H., Said, M. B., and Siala, M., 'Throughput performance of ARQ and HARQ I schemes over a two-states Markov channel model,' in 'Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems,' ISBN 9789972611001, 2005 doi:10.1109/ICECS.2005.4633423.

[8] Brázdil, T., Brožek, V., Chatterjee, K., Forejt, V., and Kučera, A., 'Unifying Two Views on Multiple Mean-Payoff Objectives in Markov Decision Processes,' in 'Proceedings - Symposium on Logic in Computer Science,' volume 13, ISBN 9780769544120, ISSN 10436871, 2017 pp. 33–42, doi:10.1109/LICS.2011.10.

[9] Chatterjee, K., 'Markov decision processes with multiple long-run average objectives,' Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2007, **4855 LNCS**, pp. 473–484, ISSN 03029743, doi:10.2168/LMCS-10.

[10] Chatterjee, K. and Henzinger, M., 'Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification,' Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, 2011, pp. 1318–1336, doi:10.1137/1.9781611973082.101.

[11] Choi, Y., Kim, H., Han, S. W., and Han, Y., 'Joint resource allocation for parallel multi-radio access in heterogeneous wireless networks,' IEEE Transactions on Wireless Communications, 2010, **9**(11), pp. 3324–3329, ISSN 15361276, doi:10.1109/TWC.2010.11.100045.

[12] Dolgov, D. A. and Durfee, E. H., 'Optimal resource allocation and policy formulation in loosely-coupled Markov decision processes,' Proceedings of the 14th International Conference on Automated Planning and Scheduling, ICAPS 2004, 2004, pp. 315–324.

[13] Feinberg, E. A. and Shwartz, A., 'Constrained discounted dynamic programming,' Mathematics of Operations Research, 1996, **21**(4), pp. 922–945, ISSN 0364765X, doi:10.1287/moor.21.4.922.

[14] Gittins, J. C., 'Bandit Processes and Dynamic Allocation Indices,' Journal of the Royal Statistical Society: Series B (Methodological), 1979, **41**(2), pp. 148–164, doi:10.1111/j.2517-6161.1979.tb01068.x.

[15] Goldsmith, A., *Wireless Communications*, Cambridge University Press, New York, NY, USA, 1 edition, 2005, ISBN 978-0-521-83716-3.

[16] Jalloul, T., Ajib, W., Yeste-Ojeda, O. A., Landry, R., and Thibeault, C., 'DME/DME navigation using a single low-cost SDR and sequential operation,' in 'AIAA/IEEE Digital Avionics Systems Conference - Proceedings,' Institute of Electrical and Electronics Engineers Inc., ISBN 9781479950010, ISSN 21557209, 2014 pp. 3C21–3C29, doi:10.1109/DASC.2014.6979451.

[17] Kester, W., editor, *The Data Conversion Handbook*, Newnes, Burlington, MA, USA, 2005, ISBN 0-7506-7841-0.

[18] Kiayani, A., Anttila, L., Kosunen, M., Stadius, K., Ryynanen, J., and Valkama, M., 'Modeling and Joint Mitigation of TX and RX Nonlinearity-Induced Receiver Desensitization,' IEEE Transactions on Microwave Theory and Techniques, 2017, **65**(7), pp. 2427–2442, ISSN 0018-9480, doi:10.1109/TMTT.2017.2654222.

[19] Kiminki, S., Saari, V., Pärssinen, A., Hirvisalo, V., Immonen, A., Ryynänen, J., and Zetterman, T., 'Design and Performance Trade-offs in Parallelized RF SDR Architecture,' in 'Proceedings of the 6th International ICST Conference on Cognitive Radio Oriented Wireless Networks and Communications,' IEEE, ISBN 978-1-936968-19-0, 2011 doi:10.4108/icst.crowncom.2011.245888.

[20] Kist, M., Rochol, J., Dasilva, L. A., and Both, C. B., 'HyDRA: A hypervisor for software defined radios to enable radio virtualization in mobile networks,' in '2017 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2017,' Institute of Electrical and Electronics Engineers Inc., ISBN 9781538627846, 2017 pp. 960–961, doi:10.1109/INFCOMW.2017.8116510.

[21] Kist, M., Rochol, J., DaSilva, L. A., and Both, C. B., 'SDR Virtualization in Future Mobile Networks: Enabling Multi-Programmable Air-Interfaces,' in '2018 IEEE International Conference on Communications (ICC),' IEEE, ISBN 978-1-5386-3180-5, 2018 pp. 1–6, doi:10.1109/ICC.2018.8422643.

[22] Lee, T. H., 'Throughput efficiency of some arq strategies under markov error models,' Electronics Letters, 1989, **25**(20), pp. 1347–1349, ISSN 00135194, doi:10.1049/el:19890900.

[23] Mbengue, A. and Chang, Y., 'Spatial and Time Domain Model for LAA/Wi-Fi Coexistence,' IEEE Communications Letters, 2018, **22**(9), pp. 1798–1801, ISSN 15582558, doi:10.1109/LCOMM.2018.2854556.

[24] Mitola, J. and Maguire, G., 'Cognitive radio: making software radios more personal,' IEEE Personal Communications, 1999, **6**(4), pp. 13–18, ISSN 10709916, doi:10.1109/98.788210.

[25] Ngangue Ndih, E. D., Cherkaoui, S., and Dayoub, I., 'Analytic Modeling of the Coexistence of IEEE 802.15.4 and IEEE 802.11 in Saturation Conditions,' IEEE Communications Letters, 2015, **19**(11), pp. 1981–1984, ISSN 10897798, doi:10.1109/LCOMM.2015.2449298.

[26] Price, N. D., Zawodniok, M. J., and Guardiola, I. G., 'Transceivers as a Resource: Scheduling Time and Bandwidth in Software-Defined Radio,' IEEE Access, 2020, **8**, pp. 132603–132613, ISSN 21693536, doi:10.1109/ACCESS.2020.3011051.

[27] Puterman, M. L., *Markov Decision Process: Discrete Stochasti Dynamic Programming*, John Wiley & Sons, Inc., Hoboken, New Jersey, 2005, ISBN 0-471-72782-2.

[28] van Berkel, K., Burchard, A., van Kampen, D., Kourzanov, P., Moreira, O., Piipponen, A., Raiskila, K., Slotte, S., van Splunter, M., and Zetterman, T., 'A Multi-radio SDR Technology Demonstrator,' in 'Proceedings of the SDR '09' Technical Conference,' 2009 .

[29] Wang, Z., Guan, L., and Farrell, R., 'Undersampling Observation-Based Compact Digital Predistortion for Single-Chain Multiband and Wideband Direct-to-RF Transmitter,' IEEE Transactions on Microwave Theory and Techniques, 2017, **65**(12), pp. 5274–5283, ISSN 0018-9480, doi:10.1109/TMTT.2017.2758366.

[30] Whittle, P., 'Restless Bandits : Activity Allocation in a Changing World,' Journal of Applied Probability, 1988, **25**, pp. 287–298.

[31] Yang, X., Chaillot, D., Roblin, P., Liou, W.-R., Lee, J., Park, H.-D., Strahler, J., and Ismail, M., 'Poly-Harmonic Modeling and Predistortion Linearization for Software-Defined Radio Upconverters,' IEEE Transactions on Microwave Theory and Techniques, 2010, **58**(8), pp. 2125–2133, ISSN 0018-9480, doi:10.1109/TMTT.2010.2053068.

[32] Zetterman, T., Piipponen, A., Raiskila, K., and Slotte, S., 'Multi-radio coexistence and collaboration on an SDR platform,' Analog Integrated Circuits and Signal Processing, 2011, **69**(2-3), p. 329, ISSN 0925-1030, 1573-1979, doi:10.1007/s10470-011-9713-7.

SECTION

2. SUMMARY AND CONCLUSIONS

In this dissertation, novel SDR front-end multiplexing and scheduling schemes are developed to increase the SDR hosting capacity of multi-radio SDR systems. First, a novel MILP technique is developed that combines existing TDM and FDM techniques to maximize the utility of available SDR front-ends over a finite horizon. Next, SDR front-end multiplexing techniques are applied in a case-study on aircraft telemetry and avionic systems. Emphasis is on improving reliability in the event of hardware failures. Finally, an improved real-time scheduling technique based in MDP is presented that addresses many of the concerns and deficiencies of previous techniques. Specifically this technique considers the long-term performance impact from dynamic resource allocation and the individual radio performance requirements of each hosted SDR. It can test the feasibility of a SDR workload and, as a result, provide guarantees for minimum performance. Feasibility testing like this may one day be used to appease concerns of wireless regulatory agencies and standards institutes.

2.1. CONCLUSIONS

In the first paper, a MILP finite horizon scheduler was developed to share SDR front-ends. In simulation, the proposed technique demonstrated service capacity improvements of up to 10% depending on several variables. A MILP scheduler was demonstrated both with and without FDM capabilities. In the tested scenarios, the addition of FDM did not greatly improve the system's hosting capacity except in highly favorable conditions.

Next, the SDR front-end sharing techniques were applied in a case study to aircraft telemetry and avionics. The proposed SDR architecture has the benefit of improved reliability in the event of multiple hardware failures. A simplified reliability model for the system was given. A simulation of the proposed SDR system was conducted featuring six different radio systems. A hybrid approach using both a fixed priority scheduler and a linear programming model was used. The respective performance of each SDR based radio system is given as a function of the total number of front-ends. As the number of SDR front-ends is reduced, performance degrades gradually for each radio system respectively. This result is considerable improvement to conventional radios systems in which each hardware failure would result in the complete loss of the respective radio system.

In the third paper, a real-time MDP based scheduler is built for shared front-end SDR systems. Unlike all other existing literature, this process can be used to estimate and control added equivalent interference from the multiplexing process. The TDM process adds a quantifiable amount of perceived interference to SDRs in instances when the number of transmissions and receptions exceed the number of available front-ends. The solution process features a feasibility test that can be used to evaluate SDR workloads. This process is best used in a scenario where the mixture of SDRs may often change. The workload can be instantly checked to either confirm or deny that each SDR within the mixture will meet its respective SER requirement.

2.2. FUTURE WORK

The presented MDP control scheme represents the most viable front-end sharing system for use in the real world; although, it is still lacking in certain respects. The presented MDP scheme requires a set of stochastic parameters for each SDR. These parameters may be difficult to collect and may change with time. A partially observable Markov decision process or a variation of the presented technique featuring Q-learning be may more practical. However, these methods may make performance guarantees more difficult to ensure. None

of the presented sharing techniques have provisions for multi-radio techniques such as beam forming or MIMO. Many common wireless protocols such as Wi-Fi and LTE make heavy use of multi-radio techniques. It may also be possible to opportunistically apply multi-radio techniques to SDRs to bolster their respective signal quality even when they do not request it explicitly.

REFERENCES

[1] 'IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,' 2016, doi:10.1109/IEEESTD.2016.7786995.

[2] 'Draft ETSI EN 303 648 V1.1.2: Reconfigurable Radio Systems (RRS); Radio Equipment (RE) reconfiguration architecture,' 2020.

[3] 'REDHAWK,' https://redhawksdr.org/, 2020.

[4] Abdelaziz, M., Anttila, L., Tarver, C., Li, K., Cavallaro, J. R., and Valkama, M., 'Low-Complexity Subband Digital Predistortion for Spurious Emission Suppression in Noncontiguous Spectrum Access,' IEEE Transactions on Microwave Theory and Techniques, 2016, **64**(11), pp. 3501–3517, ISSN 0018-9480, doi:10.1109/TMTT.2016.2602208.

[5] Ahtiainen, A., Berg, H., and Westmeijer, J., 'Architecting Software Radio,' in 'Proceeding of the SDR '07' Technical Conference,' 2007 .

[6] Ahtiainen, A., van Berkel, K., Kampen, D. V., Moreira, O., Piipponen, A., and Zetterman, T., 'Multiradio Scheduling and Resource Sharing on a Software Defined Radio Computing Platform,' in 'Proceedings of the SDR '08 Technical Conference,' 2008 .

[7] Amrhar, A., Kisomi, A. A., Zhang, E., Zambrano, J., Thibeault, C., and Landry, R., 'Multi-Mode reconfigurable Software Defined Radio architecture for avionic radios,' in '2017 Integrated Communications Navigation and Surveillance (ICNS) Conference,' 2017 pp. 2D1–1–2D1–10, doi:10.1109/ICNSURV.2017.8011903.

[8] Bertsekas, D. P., *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont, Massachusetts, 4th edition, 2012, ISBN 978-1-886529-44-1.

[9] Boujemâa, H., Said, M. B., and Siala, M., 'Throughput performance of ARQ and HARQ I schemes over a two-states Markov channel model,' in 'Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems,' ISBN 9789972611001, 2005 doi:10.1109/ICECS.2005.4633423.

[10] Brázdil, T., Brožek, V., Chatterjee, K., Forejt, V., and Kučera, A., 'Unifying Two Views on Multiple Mean-Payoff Objectives in Markov Decision Processes,' in 'Proceedings - Symposium on Logic in Computer Science,' volume 13, ISBN 9780769544120, ISSN 10436871, 2017 pp. 33–42, doi:10.1109/LICS.2011.10.

[11] Chandrashekar, S., Maeder, A., Sartori, C., Höhne, T., Vejlgaard, B., and Chandramouli, D., '5G multi-RAT multi-connectivity architecture,' in '2016 IEEE International Conference on Communications Workshops, ICC 2016,' Institute of Electrical and Electronics Engineers Inc., ISBN 9781509004485, 2016 pp. 180–186, doi: 10.1109/ICCW.2016.7503785.

[12] Chatterjee, K., 'Markov decision processes with multiple long-run average objectives,' Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2007, **4855 LNCS**, pp. 473–484, ISSN 03029743, doi:10.2168/LMCS-10.

[13] Chatterjee, K. and Henzinger, M., 'Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification,' Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, 2011, pp. 1318–1336, doi:10.1137/1.9781611973082.101.

[14] Choi, Y., Kim, H., Han, S. W., and Han, Y., 'Joint resource allocation for parallel multi-radio access in heterogeneous wireless networks,' IEEE Transactions on Wireless Communications, 2010, **9**(11), pp. 3324–3329, ISSN 15361276, doi:10.1109/TWC.2010.11.100045.

[15] Cooklev, T., Normoyle, R., and Clendenen, D., 'The VITA 49 Analog RF-Digital Interface,' IEEE Circuits and Systems Magazine, 2012, **12**(4), pp. 21–32, ISSN 1531-636X, doi:10.1109/MCAS.2012.2221520.

[16] Dolgov, D. A. and Durfee, E. H., 'Optimal resource allocation and policy formulation in loosely-coupled Markov decision processes,' Proceedings of the 14th International Conference on Automated Planning and Scheduling, ICAPS 2004, 2004, pp. 315–324.

[17] Don, M. and Ilg, M., 'Advances in a Low-Cost Software-Defined Telemetry System,' in 'International Telemetering Conference Proceedings,' International Foundation for Telemetering, 2017 .

[18] Feinberg, E. A. and Shwartz, A., 'Constrained discounted dynamic programming,' Mathematics of Operations Research, 1996, **21**(4), pp. 922–945, ISSN 0364765X, doi:10.1287/moor.21.4.922.

[19] Gittins, J. C., 'Bandit Processes and Dynamic Allocation Indices,' Journal of the Royal Statistical Society: Series B (Methodological), 1979, **41**(2), pp. 148–164, doi: 10.1111/j.2517-6161.1979.tb01068.x.

[20] Goldsmith, A., *Wireless Communications*, Cambridge University Press, New York, NY, USA, 1 edition, 2005, ISBN 978-0-521-83716-3.

[21] Hillier, F. S. and Lieberman, G. J., *Introduction to Operations Research*, McGraw-Hill, New York, NY, USA, 10 edition, 2015, ISBN 978-0-07-352345-3.

[22] Jalloul, T., Ajib, W., Yeste-Ojeda, O. A., Landry, R., and Thibeault, C., 'DME/DME navigation using a single low-cost SDR and sequential operation,' in 'AIAA/IEEE Digital Avionics Systems Conference - Proceedings,' Institute of Electrical and Electronics Engineers Inc., ISBN 9781479950010, ISSN 21557209, 2014 pp. 3C21–3C29, doi:10.1109/DASC.2014.6979451.

[23] Kester, W., editor, *The Data Conversion Handbook*, Newnes, Burlington, MA, USA, 2005, ISBN 0-7506-7841-0.

[24] Kiayani, A., Anttila, L., Kosunen, M., Stadius, K., Ryynanen, J., and Valkama, M., 'Modeling and Joint Mitigation of TX and RX Nonlinearity-Induced Receiver Desensitization,' IEEE Transactions on Microwave Theory and Techniques, 2017, **65**(7), pp. 2427–2442, ISSN 0018-9480, doi:10.1109/TMTT.2017.2654222.

[25] Kiminki, S., Saari, V., Pärssinen, A., Hirvisalo, V., Immonen, A., Ryynänen, J., and Zetterman, T., 'Design and Performance Trade-offs in Parallelized RF SDR Architecture,' in 'Proceedings of the 6th International ICST Conference on Cognitive Radio Oriented Wireless Networks and Communications,' IEEE, ISBN 978-1-936968-19-0, 2011 doi:10.4108/icst.crowncom.2011.245888.

[26] Kist, M., Rochol, J., Dasilva, L. A., and Both, C. B., 'HyDRA: A hypervisor for software defined radios to enable radio virtualization in mobile networks,' in '2017 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2017,' Institute of Electrical and Electronics Engineers Inc., ISBN 9781538627846, 2017 pp. 960–961, doi:10.1109/INFCOMW.2017.8116510.

[27] Kist, M., Rochol, J., DaSilva, L. A., and Both, C. B., 'SDR Virtualization in Future Mobile Networks: Enabling Multi-Programmable Air-Interfaces,' in '2018 IEEE International Conference on Communications (ICC),' IEEE, ISBN 978-1-5386-3180-5, 2018 pp. 1–6, doi:10.1109/ICC.2018.8422643.

[28] Lee, T. H., 'Throughput efficiency of some arq strategies under markov error models,' Electronics Letters, 1989, **25**(20), pp. 1347–1349, ISSN 00135194, doi:10.1049/el:19890900.

[29] Machado, R. G. and Wyglinski, A. M., 'Software-Defined Radio: Bridging the Analog-Digital Divide,' Proceedings of the IEEE, 2015, **103**(3), pp. 409–423, ISSN 0018-9219, doi:10.1109/JPROC.2015.2399173.

[30] Margulies, A. and Mitola, J., 'Software defined radios: a technical challenge and a migration strategy,' in '1988 IEEE 5th International Symposium on Spread Spectrum Techniques and Applications - Proceedings. Spread Technology to Africa (Cat. No.98TH8333),' volume 2, IEEE, ISBN 0-7803-4281-X, 1998 pp. 551–556, doi:10.1109/ISSSTA.1998.723845.

[31] Mbengue, A. and Chang, Y., 'Spatial and Time Domain Model for LAA/Wi-Fi Coexistence,' IEEE Communications Letters, 2018, **22**(9), pp. 1798–1801, ISSN 15582558, doi:10.1109/LCOMM.2018.2854556.

[32] Mitola, J., 'The software radio architecture,' IEEE Communications Magazine, 1995, **33**(5), pp. 26–38, ISSN 01636804, doi:10.1109/35.393001.

[33] Mitola, J., *Software Radio Architecture: Object-Oriented Approaches to Wireless Systems Engineering*, John Wiley & Sons, Inc., New York, 2000, ISBN 0-471-21664-X.

[34] Mitola, J. and Maguire, G., 'Cognitive radio: making software radios more personal,' IEEE Personal Communications, 1999, **6**(4), pp. 13–18, ISSN 10709916, doi:10.1109/98.788210.

[35] Ngangue Ndih, E. D., Cherkaoui, S., and Dayoub, I., 'Analytic Modeling of the Coexistence of IEEE 802.15.4 and IEEE 802.11 in Saturation Conditions,' IEEE Communications Letters, 2015, **19**(11), pp. 1981–1984, ISSN 10897798, doi:10.1109/LCOMM.2015.2449298.

[36] Price, N. and Kosbar, K., 'Decoupling Hardware and Software Concerns in Aircraft Telemetry SDR Systems,' 2018.

[37] Price, N. D., Zawodniok, M. J., and Guardiola, I. G., 'Transceivers as a Resource: Scheduling Time and Bandwidth in Software-Defined Radio,' IEEE Access, 2020, **8**, pp. 132603–132613, ISSN 21693536, doi:10.1109/ACCESS.2020.3011051.

[38] Puterman, M. L., *Markov Decision Process: Discrete Stochasti Dynamic Programming*, John Wiley & Sons, Inc., Hoboken, New Jersey, 2005, ISBN 0-471-72782-2.

[39] Reinhart, R. C., Kacpura, T. J., and Handler, L. M., 'TELECOMMUNICATIONS RADIO SYSTEM (STRS) ARCHITECTURE STANDARD : NASA-STD-4009A w/CHANGE 1,' 2018, **4009**, pp. 1–201.

[40] RRS, 'EN 303 146-2 - V1.2.1 - Reconfigurable Radio Systems (RRS); Mobile Device (MD) information models and protocols; Part 2: Reconfigurable Radio Frequency Interface (RRFI),' Technical report, 2016.

[41] RRS, 'EN 303 146-4 - V1.1.2 - Reconfigurable Radio Systems (RRS); Mobile Device (MD) information models and protocols; Part 4: Radio Programming Interface (RPI),' Technical report, 2017.

[42] RRS, 'EN 303 146-1 - V1.3.1 - Reconfigurable Radio Systems (RRS); Mobile Device (MD) information models and protocols; Part 1: Multiradio Interface (MURI),' Technical report, 2018.

[43] Sklivanitis, G., Gannon, A., Batalama, S. N., and Pados, D. A., 'Addressing next-generation wireless challenges with commercial software-defined radio platforms,' IEEE Communications Magazine, 2016, **54**(1), pp. 59–67, ISSN 0163-6804, doi:10.1109/MCOM.2016.7378427.

[44] Trivedi, K. S., *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, John Wiley & Sons, Inc., New York, 2 edition, 2002, ISBN 0-471-33341-7.

[45] van Berkel, K., Burchard, A., van Kampen, D., Kourzanov, P., Moreira, O., Piipponen, A., Raiskila, K., Slotte, S., van Splunter, M., and Zetterman, T., 'A Multi-radio SDR Technology Demonstrator,' in 'Proceedings of the SDR '09' Technical Conference,' 2009 .

[46] Wang, Z., Guan, L., and Farrell, R., 'Undersampling Observation-Based Compact Digital Predistortion for Single-Chain Multiband and Wideband Direct-to-RF Transmitter,' IEEE Transactions on Microwave Theory and Techniques, 2017, **65**(12), pp. 5274–5283, ISSN 0018-9480, doi:10.1109/TMTT.2017.2758366.

[47] Whittle, P., 'Restless Bandits : Activity Allocation in a Changing World,' Journal of Applied Probability, 1988, **25**, pp. 287–298.

[48] Yang, X., Chaillot, D., Roblin, P., Liou, W.-R., Lee, J., Park, H.-D., Strahler, J., and Ismail, M., 'Poly-Harmonic Modeling and Predistortion Linearization for Software-Defined Radio Upconverters,' IEEE Transactions on Microwave Theory and Techniques, 2010, **58**(8), pp. 2125–2133, ISSN 0018-9480, doi:10.1109/TMTT.2010.2053068.

[49] Yeste-Ojeda, O. A., Zambrano, J., and Landry, R., 'Design Of Integrated Mode S Transponder, ADS-B and Distance Measuring Equipment Transceivers,' 2016 .

[50] Zetterman, T., Piipponen, A., Raiskila, K., and Slotte, S., 'Multi-radio coexistence and collaboration on an SDR platform,' Analog Integrated Circuits and Signal Processing, 2011, **69**(2-3), p. 329, ISSN 0925-1030, 1573-1979, doi:10.1007/s10470-011-9713-7.

VITA

Nathan Daniel Price received a BS in both Electrical Engineering and Computer Engineering in July 2012 from the Missouri University of Science and Technology. He taught coursework as a graduate teaching assistant for the computer engineering department in summer 2014. He acted as a graduate teaching assistant on multiple occasions under Dr. Maciej Zawodniok for laboratory coursework. He received his Doctor of Philosophy in Computer Engineering from the Missouri University of Science and Technology in May 2021. His research interests included, software-defined radio systems, computer architecture, embedded systems, and real-time systems.