ARL-TN-0851 ● Oct 2017

US Army Research Laboratory

Cognitive Radio Application for Evaluating Coexistence with Cognitive Radars: A Software User's Guide

by Jeff Poston, Anthony Martone, Kelly Sherbondy, and Michael Buehrer

NOTICES

Disclaimers

ARL-TN-0851 ● OCT 2017

US Army Research Laboratory

Cognitive Radio Application for Evaluating Coexistence with Cognitive Radars: A Software User's Guide

by Jeff Poston and Michael Buehrer
Virginia Polytechnic Institute and State University, Blacksburg, VA

Anthony Martone and Kelly Sherbondy
Sensors and Electron Devices Directorate, ARL

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED (From - To)**
October 2017	Technical Note	5 June 2017 to 11 August 2017

4. TITLE AND SUBTITLE	**5a. CONTRACT NUMBER**
Cognitive Radio Application for Evaluating Coexistence with Cognitive Radars: A Software User's Guide	
	5b. GRANT NUMBER W911NF-17-2-0021
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	**5d. PROJECT NUMBER**
Jeff Poston, Anthony Martone, Kelly Sherbondy, and Michael Buehrer	W911NF-17-2-0021
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	**8. PERFORMING ORGANIZATION REPORT NUMBER**
US Army Research Laboratory ATTN: RDRL-SER-U 2800 Powder Mill Road Adelphi, MD 20783-1138	ARL-TN-0851
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This application demonstrates several cognitive radio behaviors and is intended as a research tool with which to study the interaction of the cognitive radio with both conventional wireless systems as well as other types of cognitive RF systems (e.g., cognitive radar). The radio hardware for this application is the Ettus Research Universal Software Radio Peripheral (USRP), and this application was tested on the USRP Model N210 configured with an SBX or WBX daughtercard. This technical report begins with a system-level overview in Section 1. Then, the remaining sections explain the configuration and usage of the application.

15. SUBJECT TERMS

cognitive radio, Universal Software Radio Peripheral, USRP, radar, coexistence, user's guide

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Kelly Sherbondy
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	UU	20	**19b. TELEPHONE NUMBER (Include area code)**
Unclassified	Unclassified	Unclassified			(301) 394-2533

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction and Theory of Operation

The system model has 2 kinds of cognitive radio nodes: a base station and one or more mobiles. By default, all cognitive radios begin in *rendezvous mode* where they broadcast beacon messages on a sequence of frequency channels and attempt to discover one another's presence. Once this discovery happens the base station commands mobiles to remain on the channel where this discovery process was successful and switch to *communications mode* for the purpose of exchanging data. If at some point this channel is no longer viable for communications (e.g., due to interference on the channel) then the base station commands mobiles to revert to rendezvous mode and the discovery process begins anew. For reasons to be explained shortly, the commands on the downlink from the base station to mobile can be assumed to be received reliably even though the uplink from mobile to base station may not be reliable. In addition, the application supports special test modes that lock the cognitive radio into communications-only mode or rendezvous without regard to the interference conditions.

The base station and mobiles communicate by means of frequency division duplex (FDD) between the base station and mobiles with orthogonal frequency division multiplexing (OFDM) as the physical layer modulation. For the link layer, the base station and mobiles generate data and control frames according to one or more cognitive radio spectrum-access behaviors and emulation of higher-layer network protocols. In the scenario that motivated the development of this research tool, other cognitive RF systems (e.g., cognitive radar) operate in the same radio band and geographical region. Given that the base stations are at fixed positions and often elevated and operating with relatively high power compared with mobiles, it is straightforward for cognitive RF systems to detect the base station's transmissions and avoid activity that would harm this downlink. By contrast, the mobile positions are generally unknown a priori and path-loss variations at different points in the mobile trajectories create uncertainty about detecting the mobile transmission. Thus, a cognitive RF system may incorrectly conclude the frequencies for the uplink are vacant and inadvertently cause harmful interference to the uplink. For this reason the rendezvous process of this application operates only on the selection of uplink frequency channel.

2. Overview of Application Software

The target operating environment for this application is Ubuntu 16.04 plus additional libraries and drivers needed for signal processing and radio hardware interfacing. To facilitate reproducible research the entire environment, including the application software, comes in the form of a virtual machine created with VirtualBox software. All of this required software is open source. If any of this software is redistributed then the terms of the open-source licenses require that copies of the license text be included with the distribution. This cognitive radio software package includes a directory `legal_notices` containing text files for the relevant licenses.

The application itself, `cogradio_app`, is written in C++ because C++ is the language of the Ettus-provided application programming interface known as the USRP Hardware Driver. The signal-processing portions of the application, however, are almost entirely in pure C.

In broad terms, there 4 kinds of C++ class in this application:

- Communication Protocols
- Managers
- Monitors
- Utilities

The communications protocols consist of the physical layer that provides an OFDM modem and related signal processing, a link layer, and emulated higher layer protocols. This link layer organizes sequences of incoming or outgoing data into frames. The frame syntax includes source and destination addressing, designation of frame type (e.g., data, control message, or beacon), payload, and a cyclic redundancy code integrity check of the frame contents. These frames could be organized into network layer packets (e.g., IP datagrams) but, at present, the application only emulates network and higher-protocol layer activity and collects statistics about the link layer frames.

The manager class control resources and configuration. The `AppManager` controls interaction between the host computer and the application, reads the user-provided configuration file, and initializes other application class objects. The `FrequencyManager` partitions the permitted operating frequency ranges into discrete frequency channels. The `StateManager` maintains the application's internal representation of operating configuration that can be changed autonomously by the application. The `ActionManager` evaluates application performance and, subject to the cognitive radio's adaptation policy, changes the operating configuration (i.e., application state) to optimize performance objectives.

The `RewardMonitor` transforms low-level communications statistics into a composite performance measure of the value (i.e., utility) the application currently provides. It is this measure that guides the `ActionManager` in the decision-making process. There are also utility classes `ErrorHandler` and `Logger` that produce error messages and log files, respectively. This application collects a variety of statistics about its operating status and performance. The user-provided configuration file specifies which of these are saved to log files and the name of the log files. The log files are plain-text files that are self-documenting in the sense that they begin with a preamble that explains the syntax and semantics of the remainder of the file. All logs contain a common, application-wide timestamp for each log entry reporting an event, state change, or performance statistic. Thus, entries in separate logs from the same experiment can be correlated by their timestamp. The source of the timestamp is the host computer executing the `cogradio_app` program. Figure 1 shows a plot produced from log files by one of the example post-processing utilities included with this cognitive radio software package. Specifically, the program `plot_example_basestation_logs.m` within the directory `log_postprocessing_examples` generates this plot. The program runs in either Octave or MATLAB. (Figure 1's plot is for illustrative purposes only and should not be construed as a representative performance characterization.)

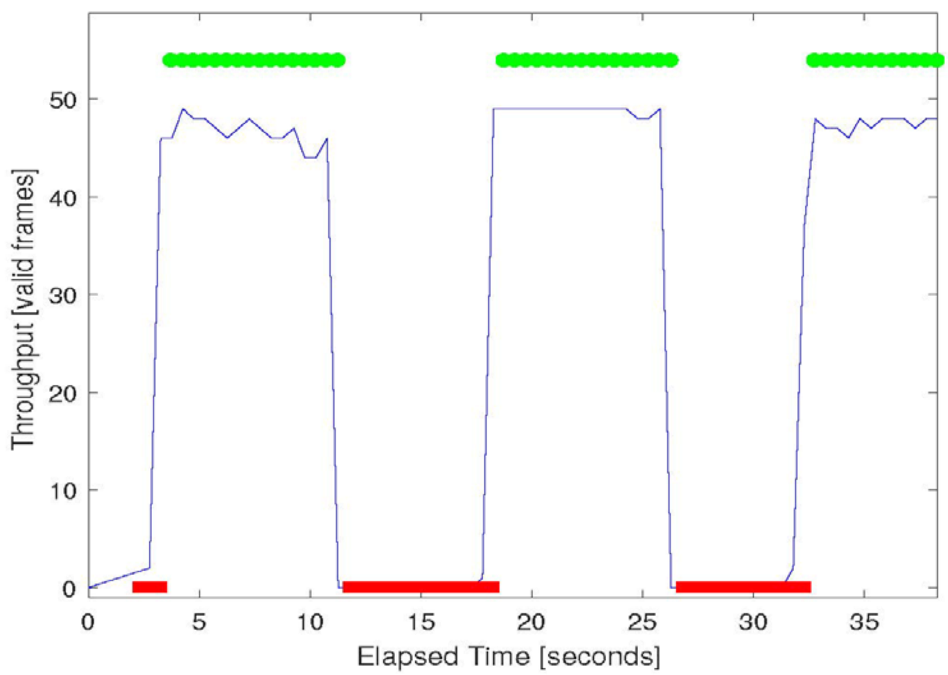

Fig. 1 Example plot from postprocessing logs from a cognitive radio operating as base station: blue line—number of frames successfully sent from mobile to base station per time unit (a metric of utility of operating in a particular configuration); red marks—points in time when cognitive radio operated in rendezvous mode; and green circles—radio operation in communications mode

To review the application's source code, the Doxygen utility program can generate automatically the documentation for the application's classes, member functions, and structures as well as other definitions. A sample Doxygen configuration file is provided in (`./doc/doxygen/Doxyfile`); thus, to generate this documentation one only needs to enter the doxygen directory and at the command line enter

```
doxygen ./Doxyfile
```

This Doxygen configuration generates Hypertext Markup Language files (i.e., documentation viewable in a web browser) in the `\./doc/doxygen\_html/` subdirectory. When reviewing the source code it is important to keep in mind the code was written in anticipation of capabilities that might be useful for research, rather than being written to satisfy a design specification. Similarly, there is some redundancy of class variables and methods that could be reduced by class–subclass inheritance if a design specification did emerge.

3. Application Configuration

The user begins by logging into the virtual machine's "demo" account (login password available separately from the author of this guide) and confirming proper configuration of hardware and networking prior to starting the application. The remaining instructions in this section assume the reader has a basic familiarity with the Linux command line. Some steps require sudo (i.e., root) permissions to permit certain network operations. In Ubuntu, when a user enters a command that requires these permissions the user is prompted with the message to enter the "[sudo] password." In Ubuntu this is just the password for the user account ("demo").

The VirtualBox virtual machine can operate with different emulated hardware configurations.

The recommended configuration is having at least 4 central processing units, at least 4 GB of random-access memory, and bridged Ethernet with static IP addresses for the network connection to the USRPs. As noted in Section 1, a specific radio's configuration presumes the existence of an overall radio network, thus requiring at least 2 USRPS: one operating as a base station and the other as a mobile. Prior to starting the application, the user should confirm network connectivity between the virtual machine(s) and the USRPs by, for example, executing the `ping` command.

The user begins the application by opening a terminal and invoking the program from the command line. Here is a summary of command line-usage:

```
USAGE:
```

```
cogradio_app [OPTIONS]

This program permits options to be specified either in a short
form (a single hyphen followed by a single letter) or in a long
form (double hyphen followed by full option name). Unless noted
otherwise, these options require an argument. For string
arguments double quotes are accepted but not required.

OPTIONS:

    -c, --config-file      Configuration file name
                           (default: cogradio_app.cfg)
    -d, --debug-setting    For development testing only
                           (default: empty setting)
    -h, --help             This help message; needs no argument
    -v, --verbosity        Set application verbosity level:
                             0:none,...,4:high (default: 3)
```

The configuration file read by the program provides the required operating parameters. Furthermore, the configuration file also documents the settings for a particular cognitive radio experiment. As the example listing in the Appendix shows, the configuration files have a C-style syntax. For example, `phy_freq_base_tx_default = 2.425E9`; is an assignment for a floating-point parameter, and `app_log_file = "test.log"`; is an assignment for a string parameter. Most of the configuration options should be straightforward to understand from their names and the accompanying comments in the example configuration file. The order of the settings does not matter, and it is straightforward to generate the configuration file automatically from another program.

In the `demo` subdirectory there are several example configuration files along with shell scripts that invoke the `cogradio_app` with required parameters, including an appropriate example configuration file.

INTENTIONALLY LEFT BLANK.

Appendix. Configuration File Listing

This appendix appears in its original form, without editorial change.

```
# cogradio_app.cfg
#
# This is an example configuration file read by the cognitive radio
# application in order to adjust operating parameters to user-defined
# settings.  Omitted settings revert to default values as explained
# in the comments below that accompany each parameter's description.
#
# In order to start the cognitive radio application and specify a file
# (e.g., "cogradio_app.cfg") as the source of configuration information
# use this program's -c option:
#
#   ./cogradio_app -c ./cogradio_app.cfg
#
# In the configuration file lines like these beginning with "#" are
# treated as comments to be ignored by the program's parser.
# Parameter settings have a C style syntax.
# For example, for an integer parameter the syntax is:
#   a = 123;
# For a floating point parameter the syntax is:
#   b = 1.23E6;
# For a string parameter the syntax is:
#   c = "name";
#
# The remainder of this file shows parameters grouped by feature for
# ease of reading, but when the application processes this file the
# actual order of the parameters is unimportant.
#
#####################################################################


#####################################################################
#   Application operation parameters
#####################################################################

# app_log_action_file
# File name for logging of cognitive radio's actions; if omitted there
# is no logging.
# default: (no logging)
app_log_action_file = "cogradio_action.log";

# app_log_file
# File name for overall application status logging; if omitted there
# is no logging, but some status information could be captured by
# redirecting console output to a file.
# default: (no logging)
app_log_file = "cogradio_app.log";

# app_log_reward_file
# File name for logging of reward or utility observed from a particular
# operating state or configuration;  if omitted there is no logging.
# default: (no logging)
app_log_reward_file = "cogradio_reward.log";

# app_log_state_file
# File for logging of cognitive radio's operating state and related
# state variables that influence operating behavior; if omitted there
# is no logging.
# default: (no logging)
app_log_state_file = "cogradio_state.log";
```

```
# app_run_time
# Program run duration in seconds; a value < 0 means run forever
# default: 100.0
app_run_time = 100.0;


######################################################################
#   Cognitive radio behavior parameters
######################################################################

# cr_adaptation_behavior
# The behavior for how the cognitive radio responds to interference
# The options are:
#
#   CR_ADAPTATION_LOCKED_IN_COMMUNICATIONS
#   Remain locked to the default frequencies and conduct communications
#   even during interference.
#
#   CR_ADAPTATION_LOCKED_IN_RENDEZVOUS
#   Continue operating in rendezvous whether or not the base station
#   and mobile detect one another and whether or not interference
#   exists on any rendezvous attempt
#
#   CR_ADAPTATION_MOBILE_RENDEZVOUS_ONCE
#   This policy only adapts the uplink from mobile to base station.
#   Adapt at start of application, if needed, but thereafter remain
#   in the selected operating configuration even if new interference
#   appears in the selected operating configuration.
#
#   CR_ADAPTATION_MOBILE_RENDEZVOUS_REPEATABLE
#   This policy only adapts the uplink from mobile to base station.
#   Adapt at start of application, if needed, and again any time new
#   interference observed in the current operating configuration.
#
# default: CR_ADAPTATION_MOBILE_RENDEZVOUS_ONCE
cr_adaptation_behavior = "CR_ADAPTATION_MOBILE_RENDEZVOUS_ONCE";

# cr_operates_as_mobile
# This Boolean flag controls whether the application operates as a
# mobile (i.e., cr_operates_as_mobile = 1) or as base station
#  (i.e., cr_operates_as_mobile = 0).
# default: 0 (operate as a base station)
cr_operates_as_mobile = 0;


######################################################################
#  Media access control (MAC) layer parameters
######################################################################

# mac_mobile_id_set
# The list of link layer addresses of all mobile nodes participating
# in this radio network.    All mobile node addresses must be in the
# range from 1 to 254, inclusive. The address 0 is reserved for
# the base station and 255 for a broadcast address.  It is acceptable
# to add addresses of non-existent nodes for the purpose of emulating
# resource  allocation in a radio network that has more planned nodes
# than the actual physical nodes being tested.  For example, for a set
# of two mobile nodes with corresponding link layer node addresses of
```

```
# 1 and 2 then the parameter setting would be:
#   mac_mobile_id_set = [1, 2];
# default: (no list)
mac_mobile_id_list = [1, 2];

# mac_node_id
# The unique link layer address of this base station or mobile; address
# values can be in the range of 1 to 254, inclusive.  This address
# must appear within the mac_mobile_id_list too.
# default: 0
mac_node_id = 0;


#######################################################################
#   Physical layer parameters
#######################################################################

# phy_freq_base_tx_max
# The maximum permitted transmit frequency of the base station.
# Note that this must be consistent with any physical limit on the
# minimum required Frequency Division Duplex (FDD) separation
# between the base station and mobile transmissions.
# default: 839.9E6; MHz
phy_freq_base_tx_max = 839.9E6;

# phy_freq_base_tx_min
# The minimum permitted transmit frequency of the base station
# Note that this must be consistent with any physical limit on the
# minimum required Frequency Division Duplex (FDD) separation
# between the base station and mobile transmissions.
# default: 829.5 MHz
phy_freq_base_tx_min = 829.5E6;

# phy_freq_base_tx_requested
# The preferred transmit frequency of the base station
# Note that this must be consistent with any physical limit on the
# minimum required Frequency Division Duplex (FDD) separation
# between the base station and mobile transmissions.
# default: 830.0 MHz
phy_freq_base_tx_requested = 830.0E6;


# phy_freq_mobile_tx_max
# The maximum permitted transmit frequency of the mobile
# Note that this must be consistent with any physical limit on the
# minimum required Frequency Division Duplex (FDD) separation
# between the base station and mobile transmissions.
# default: 794.9 MHz
phy_freq_mobile_tx_max = 794.9E6;

# phy_freq_mobile_tx_min
# The minimum permitted transmit frequency of the mobile
# Note that this must be consistent with any physical limit on the
# minimum required Frequency Division Duplex (FDD) separation
# between the base station and mobile transmissions.
# default: 784.5 MHz
phy_freq_mobile_tx_min = 784.5E6;

# phy_freq_mobile_tx_requested
```

```
# The preferred transmit frequency of the mobile
# Note that this must be consistent with any physical limit on the
# minimum required Frequency Division Duplex (FDD) separation
# between the base station and mobile transmissions.
# default: 785.0 MHz
phy_freq_mobile_tx_requested = 785.0E6;


# phy_modem_sample_rate
# Sample rate between the radio and the modem(s) within this
application;
# the rate is specified in samples per second (sps).
# Note that this rate does not account for any radio-internal resampling
# operations between the analog-to-digitial converter (ADC), digitial-
# to-analog converter (DAC), digital upconversion stage or digital
# downconverter stage of the radio.
# default 1.0e6 (1 Msps)
phy_modem_sample_rate = 1.0e6;

# phy_rf_gain_rx
# RF analog stage receive gain in dB
# default: 31.5
phy_rf_gain_rx = 31.5;

# phy_rf_gain_tx
# RF analog stage transmit gain in dB
# default: 1.0
phy_rf_gain_tx = 1.0;

# phy_usrp_ip_address
# Network address of the USRP network port connected to the host
# running this application.  Typically, this is a static IP address,
# not a DNS name.  If an empty string is given then the application
will
# attempt to operate with the first USRP it finds; however, that process
# is not as reliable as operating with a specified IP address.
# default: "";
phy_usrp_ip_address = "192.168.10.2";

# phy_usrp_daughterboard_model
# The string name of the daughtercard installed in the USRP.  The
# supported daughtercards include:
#   "CBX"   "CBX-120"   "SBX"   "SBX-120"   "WBX"   "WBX-120"
#   "UBX"   "UBX-120"
# default: "SBX"
phy_usrp_daughterboard_model = "SBX";

# phy_usrp_device_model
# The string name of the the type of USRP.  The supported models
include:
#   "N210"  "X300"  "X310"
# default: "N210"
phy_usrp_device_model = "N210";


# End of configuration parameters; leave a blank line after this one.
```

List of Symbols, Abbreviations, and Acronyms

FDD	frequency division duplex
IP	Internet Protocol
RF	radio frequency
OFDM	orthogonal frequency division multiplexing
USRP	Universal Software Radio Peripheral

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIR ARL IMAL HRA RECORDS MGMT RDRL DCL TECH LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
3 (PDF)	DIR ARL RDRL SER-U A MARTONE K GALLAGHER K SHERBONDY
2 (PDF)	VIRGINIA POLYTECH INST M BUEHRER J POSTON

INTENTIONALLY LEFT BLANK.