FLORIDA POLYTECHNIC UNIVERSITY

MASTER'S THESIS

ARCAM-NET: A Software Defined Radio Network Testbed

Author:
John MCCORMACK

Committee Chair:
Dr. Ryan INTEGLIA

A thesis submitted in fulfillment of the requirements
for the degree of Masters of Engineering

in the

College of Engineering

Committee Members:
We **accept and approve** the thesis of the student named above.

_____   _____

Dr. Ryan Integlia   Date

_____   _____

Dr. Anas Salah Eddin   Date

_____   _____

Dr. Harish Chintakunta   Date

April 20, 2016

Declaration of Authorship

I, John MᴄCᴏʀᴍᴀᴄᴋ, declare that this thesis titled, "ARCAM-NET: A Software Defined Radio Network Testbed" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.

- Where I have consulted the published work of others, this is always clearly attributed.

- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

- I have acknowledged all main sources of help.

Signed:

_____

Date:

_____

Advisor Approval

_____   _____

Dr. Ryan Integlia   Date

FLORIDA POLYTECHNIC UNIVERSITY

Abstract

Dr. Ryan Integlia

College of Engineering

Masters of Engineering

ARCAM-NET: A Software Defined Radio Network Testbed

by John MCCORMACK

ARCAM-Net is a Software Defined Radio Network (SDRN) testbed and platform. ARCAM-Net was born out of a need for a platform to develop SDRN applications. At the start of the project, no readily available system could be identified that would allow for user space applications to be run on an SDRN. Thus work began on the creation of ARCAM-Net. ARCAM-Net creates a fully functioning data network capable of acting as a multi-hop mesh network. Nearly every component of ARCAM-Net, except the Software Defined Radios (SDRs) themselves, is released using an open source license. The goal of ARCAM-Net is to establish a low cost platform that can be quickly implemented by anyone interested in experimenting with SDRNs. The system provides a simple user interface for interacting with the network. User space applications can be run on the network as if it was a traditional network.This document presents the network itself and also acts as a manual for working with ARCAM's first implementation.

This work represents a first step in a longer term project to create a fully functioning Cognitive Radio Ad-Hoc Network (CRAHN). The work demonstrates the potential for using GNU Radio in conjunction with batman-adv and other Open-Mesh solutions. As the work continues forward, we hope the testbed can serve as a collaboration point between GNU Radio and Open-Mesh.

Both represent next generation, open source, wireless solutions and could likely benefit from collaboration between the two projects. This work can be used as an excellent starting point for anyone looking to get an SDRN up and running quickly to begin prototyping other sections of the tool chain. We plan on releasing the code as well as a handful of tools to help other researchers get started. This work can serve as the first step in creating an equivalent Cognitive Radio Network platform and testbed.

To my parents, Joe and Kathy McCormack for supporting me in all my endeavors.

Acknowledgements

Contents

List of Figures

List of Abbreviations

ADC	Analog to Digital Converter
ALFRED	Almighty Lightweight Fact Remote Exchange Daemon
BATMAN	Better Approach To Mobile Ad-hoc Networking
CGRAN	Comprehensive GNU Radio Archive Network
COTS	Commerical Off The Shelf
CR	Cognitive Radio
CRN	Cognitive Radio Network
CRAHN	Cognitive Radio Ad-Hoc Network
DAC	Digital to Analog Converter
DIY	Do It Yourself
DSP	Digital Signal Processing
DSS	Dynamic Spread Spectrum
FCC	Federal Communications Commission
FOSS	Free Open Source Software
FPGA	Field Programmable Gate Array
GMSK	Gaussian Minimum-Shift Keying
GRC	GNU Radio Companion
IRC	Internet Relay Chat
ISM	Industrial Scientific Medical
ISP	Ineternet Service Provider
LAN	Local Area Network
LNA	Low Noise Amplifier
MAC	Medium Access Control
MANET	Mobile Ad-hoc NETwork
MPR	Multi Point Relays
OFDM	Orthogonal Frequency Division Multiplexing
OLSR	Optimized Link State Routing
OOT	Out Of Tree
OSI	Open Systems Interconnection
PA	Power Amplifier
PHY	PHYsical Layer
PMT	Poly Morphic Tree
PU	Primary User
RF	Radio Frequency
RFNoC	Radio Frequency Network on a Chip
SCP	Secure CoPy
SDR	Software Defined Radio
SDRN	Software Defined Radio Network
SSH	Secure SHell

SU	**S**econdary **U**ser
TAP	Network **TAP**
TCP	**T**ransmission **C**ontrol **P**rotocol
TUN	**TUN**nel
UDP	**U**ser **D**atagram **P**rotocol
UHD	**U**SRP **H**ardware **D**river
USB	**U**niversal **S**erial **B**us
USRP	**U**niversal **S**oftware **R**adio **P**eripheral
VHDL	**V**HSIC **H**ardware **D**efinition **L**anguage
VHSIC	**V**ery **H**igh **S**peed **I**ntegrated **C**ircuits
VOLK	**V**ector **O**ptimized **L**inux **K**ernel
WAN	**W**ide **A**rea **N**etwork
WLAN	**W**ireless **L**ocal **A**rea **N**etwork

Chapter 1

Introduction

The Advanced Radio Communication Ad-Hoc Mesh Network, or ARCAM-Net, is a platform formed through the union of several other projects and concepts [1]. The main concepts embodied within ARCAM-Net are mesh networks and software defined radio networks. The current iteration of ARCAM-Net is developed using GNU Radio and Batman-adv. As work progresses on the platform, the software defined radio network can be transitioned to a cognitive radio network by leveraging the capabilities found within GNU Radio.

1.1   Mesh Networks

In a traditional Wide Area Network (WAN), a user is able to connect to the internet through an Internet Service Provide (ISP). The user will almost always pay the ISP in exchange for the ability to connect to the rest of the internet. This is considered a centralized way to connect to the internet, where the users all connect through a few central points in the ISP's infrastructure. An alternative to this type of networking is multi-hop, ad-hoc, mesh networking. Mesh networks are decentralized networks [8].

In an ad-hoc network, each radio is able to communicate directly to any other radio within its transmission range. There is no need to connect to a central router [8]. Ad-hoc networks are defined at the physical layer (PHY), or layer 1 in the Open Systems Interconnect (OSI) model. Mesh routing takes place as part of layer 2 or 3 of the OSI model depending on the routing protocol chosen. A mesh network builds upon an adhoc network by allowing radios to retransmit any packets they receive. This allows two radios to communicate over a larger distance by leveraging other radios located in between the sender and receiver[9].

In simple mesh networking protocols, any packet sent may flood through the network to every other radio. However, with more advanced protocols an algorithm is used to ensure that a packet follows a direct path from sender and receiver and only uses a hop if necessary. The distributed nature of a mesh network creates many unique features. The decentralized nature of a mesh network prevents issues related to single points of failure. If a node goes down, the network can reconfigure and find a new path to the target [9].

1.2   Software Defined Radio Networks

Software Defined Radios (SDRs) are radio communication systems that utilize software to process radio frequency information in place of traditional hardware [10]. A radio frequency front end is able to capture and transmit signals, while the actual processing of the signal is taken care of by a digital system like general purpose processor on a traditional computer. This allows for a single piece of hardware to replace the need for multiple types of radios [11].

A typical cell phone can have a bluetooth, wifi, gps, and cellular radio all in a very small package. In the future, these systems could be replaced by a single SDR [11]. SDRs are capable of using both digital and analog transmission protocols. They can use general purpose processors, digital signal processors [11], or FPGAs [12] to process the RF information. Analog to Digital Converters (ADCs) are used to receive data from the antenna while Digital to Analog Converters (DACs) are used to transmit the processed signals .

As the name suggests, a Software Defined Radio Network (SDRN) is a network made up of SDRs. The networks can operate on a nearly infinite combination of center frequencies, amplitudes, bandwidths, and protocols [13]. The flexibility of an SDRN is limited by the physical hardware capabilities of the SDR, the computation speed of the processing unit, and regulations from governing bodies like the Federal Communications Commission (FCC). Still, the flexibility of reconfigurable radios leads to opportunities for advancing communications infrastructure beyond traditional protocols.

1.3   Batman-adv

The Better Approach to Mobile Ad-Hoc Networking Advanced (Batman-adv) protocol is a layer 2 mesh routing protocol developed by the Open Mesh foundation [14]. The protocol uses a Transmission Quality (TQ) metric to find a tradeoff between a low hop count and stable links. Every node on the network broadcasts an originator message to its neighbors. Each neighboring node listens for these broadcasted messages and uses the number received to create the TQ metric. Each node also broadcasts the names of its neighbors. This allows for the creation of a multihop network where packets can be forwarded through other routers until they reach their destination [15].

1.4   GNU Radio

GNU Radio is an open source tool chain for digital signal processing [16] . The GNU Radio Companion is a graphical programming tool for developing DSP systems. GNU Radio can be used for a variety of topics, but it has many tools for working with software defined radios [17]. GNU Radio is free to use, and

all of the source code is available for download under the GNU Public License [18].

1.5   Trends Towards Cognitive Radio Environments

Cognitive Radio Networks (CRNs) are systems of SDRs that are capable of utilizing artificial intelligence and machine learning to dynamically alter transmission patterns in real time [19]. These decisions can be made by a central server that oversees all nodes on the network, but modern systems strive to make each node capable of independent decisions.

Cognitive Radio Ad-hoc Networks (CRAHNs) combine software defined radios to form mesh networks [20]. Cognitive features of the radios can allow them to change transmission parameters in accordance with link quality to ensure packets are routed properly in the mesh network. The CRs could make small changes, like increasing or decreasing their gains, in order to continue to transmit to moving nodes. They could also make larger changes, like switching entire protocols, to adapt to the needs of the network in near real time.

Beyond ensuring good throughput in a network, CRNs also solve a major issue facing the wireless world. Frequency spectrum is a finite resource, and the available bandwidth is being quickly used up. CRNs can utilize frequency hopping to share frequency resources with traditional wireless communication systems [21]. A CRN can begin its operation on a specified band. If a traditional transmitter, typically called the primary user (PU), begins to operate on that frequency then the CRN will "hop" by switching to a different frequency and continuing operation.

1.6   ARCAM-Net

In order to begin work on SDRNs and CRAHNs, a testbed and research platform needed to be established. This thesis serves to describe the Advanced Radio Communication Ad-hoc Mesh Network, or ARCAM-Net, test bed and platform. The goal of this project is to create a open source, low cost, research platform that can serve as the basis for future work at Florida Polytechnic University and other organizations due to the open source nature of the project. By combining well established open source tools, ARCAM-Net could become a great tool for other research groups to begin working with SDRs. All software used by ARCAM-Net is freely available. The only costs come from the hardware.

ARCAM-Net's architecture will be thoroughly discussed in this document, but the two major components are GNU Radio and Batman-adv. GNU Radio is an open source toolchain for creating digital signal processing tools to be used with SDRs. Batman-adv is a layer 2 mesh networking protocol. Both projects are community driven, but exist as separate entities. I hope that with

ARCAM-Net the two projects can begin collaborating to help create next generation wireless network resources.

ARCAM-Net is designed to be a platform. Therefore all of the code will be released on github. I encourage interested users to fork the repository and begin experimenting on their own.

Chapter 2

Literature Review

The following section provides an overview of the current work being done in this field. The first section provides an overview about Software Defined Radios. Next, tools for working with SDRs are presented and discussed. Then, various networking concepts are described with a focus on mesh networks. Then, current work on software defined radio testbeds are examined and compared to ARCAM-Net. Finally, the state of the practice and the state of the art are examined.

2.1   SDR Fundamentals

A software defined radio, or SDR, is a radio system which performs signal processing in software instead of relying on dedicated hardware components to process signals. In a traditional hardware radio, each component is designed and built to handle a single task. If changes are needed, the hardware components must be replaced. Software defined radios are flexible and can be changed by altering code instead of components. A single SDR can be configured to act as numerous hardware radios. [16] [22]

An SDR itself is the hardware component which then offloads the data to a processing unit such as an FPGA, microprocessor, or general purpose computer [16] [22]. There are many different SDRs available on the market but they share many common components. Every SDR works in slightly different ways, but they share many common components.

An "Ideal" SDR only has a few stages. The first is the RF amplification stage. These stages increase the gain of the signal as it enters and leaves the radio. A Low Noise Amplifier (LNA) is used on the receiver and a Power Amplifier (PA) is used on the output. The PA dramatically extends the range of the SDR and the LNA increases the signal to noise ratio of the receiver [10].

In the "digital conversion" stage, an analog to digital (ADC) and digital to analog (DAC) converter are used to change from the analog to digital domain [10][23]. The ADC receives the analog signal and converts it to a digital code. The DAC takes digital material and converts it to an analog signal to be transmitted.

The final stage is the "processing" stage. In this stage the digital signal is sent to and from the software processor. This can come in many different forms including microcontrollers, FPGAs, and general purpose computers [11][23].

In an ideal SDR, the DAC and ADC can receive and transmit any needed signal. In practice, a real SDR requires a "frequency conversion" stage [11] [23]. This is an analog preprocessing stage that allows the useable frequency range of an SDR to be increased. Without this stage, the SDR's useable frequency range would be limited by the ADC and DAC to a much smaller range.

SDR has numerous benefits compared to Traditional Radio. A single SDR can replace a multitude of radios, saving on space and cost. The operating frequency of an SDR is also flexible. This indicates that a radio can be moved from country to country and only the software and antenna needs to be changed to operate on new frequencies [11] [24]. As new standards, protocols, and technologies are developed, an SDR can download new software to adjust its operating parameters. Compare this to existing Wi-fi routers, where a person needs to purchase a whole new router as new standards are created [10] [24].

2.2 GNU Radio and MATLAB

There are several different tools available for creating SDR software. A few different versions are presented below.

2.2.1 GNU Radio

GNU Radio is a free, open source, digital signal processing toolset aimed at allowing for quick and easy development of software defined radio programs. GNU Radio is licensed under the GNU General Public License (GPL) version 3 [24]. GNU Radio itself is made up of a combination of python and C++ modules. It relies heavily on the Vector-Optimized Library of Kernels (VOLK) [25] and the Boost C++ libraries [26].

GNU Radio programs can be developed in python or using the GNU Radio Companion [27]. GNU Radio Companion (GRC) is a graphical tool for creating signal flow graphs and can also be used to generate python code from the flow graphs. GRC is useful for creating working SDR software with minimal programming experience. It also allows for the development of parallel processing without having to understand the details of keeping everything thread safe [27].

Custom GRC blocks can be written in C++. Flowgraphs developed in GRC can also be converted into custom blocks by first converting them to hierarchical blocks. GRC can also create interactive GUIs by leveraging Qt or Wx [27].

GNU Radio and GNU Radio Companion can be extended by using out of tree (OOT) modules [28]. These are a set of modules that are not part of the core feature set of GNU Radio, but have been identified as important by the community. The official OOT repository is hosted by the Comprehensive GNU Radio Archive Network (CGRAN) [29]. GNU Radio also provides the PyBOMBS tool to create an "app store" style interface for downloading GNU Radio tools [28].

GNU Radio has a large community of active users. A mailing list and internet relay chat (IRC) channel are available for asking questions. Source code is updated often and anyone can contribute to the process [2]. Though hardware is not specifically part of GNU Radio, numerous vendors such as Ettus Research and Nuand offer support libraries for working with GNU Radio [30].

2.2.2 MATLAB

MATLAB and Simulink are another set of tools used for working with software defined radios. MATLAB and Simulink are commercial options developed by Mathworks. They are not free or open soure. The two pieces of software provide similar functionality to GNU Radio, but give you the ability to leverage Mathwork's support staff and knowledge base [31]. For this project, MATLAB was evaluated but ultimately not used. The decision was made to use as many free and open-source software (FOSS) components as possible.

2.2.3 RFNoc

Ettus Research recently released the RF Network on Chip (RFNoC) extension to GNU Radio. RFNoc allows for the creation of FPGA code blocks. Unlike normal GRC blocks, these FPGA blocks compile into code that can then be run on the FPGA located on an SDR. This can dramatically increase the speed of the block. Users can create additional RFNoC blocks in VHDL and Verilog. Unfortunately, RFNoC is only compatible with X300 and E300 series SDRs, so the software could not be used for ARCAM-Net [32].

2.3 Network Concepts

A Network consists of two or more computers that are connected together in a way that allows them to share resources, transfer files, and communicate electronically. The connection between them could be physical or wireless. Traditional computer networks include local area networks (LAN) and wireless local area networks (WLAN) [33] [34].

2.3.1 Mesh Network

Mesh networks, sometimes referred to as Mobile Ad-hoc NETworks(MANETs), are a type of network in which routers connect directly to nearby routers and connect indirectly to far away routers by "hopping" through the nearby routers until the packets reach their destination. Mesh networks can be created independent of traditional internet service providers (ISPs) as anyone with a router is able to join and create their own node on the network [35].

Instead of using traditional and expensive network infrastructure, mesh networks rely on sheer number of nodes to create reliable and distributed networks. If a single node fails, the network can reconfigure and use other nodes to route packets to their destination. Expanding the network is as simple as adding more nodes to the edges of the network [35].

2.3.2 Mesh Network Testbeds

The CONFINE platform uses batman-adv as the routing protocol for their mesh network testbed. However, their testbed does not utilize GNU Radio or SDRs. [36] batman-adv was also a key component of WiBed, a project to create a Commercial off-the-shelf (COTS) mesh test bed using low cost wireless routers [37] [38]. Though they are not specific to SDR platforms, these still show the usefulness of batman-adv as a component of a testbed.

2.3.3 GNU Radio Mesh Networks

There has been work done in the past on establishing mesh networks using GNU Radio, however most of the research has had a different focus than ours. In [39] and [40] the authors use GNU Radio as a way to verify the successful use of algorithms for mesh networking. However, they do not use SDRs, instead choosing to use GNU Radio for simulation.

The researchers in [41] created a simple multihop test bed using three USRP radios to relay data from one computer to another. A fourth USRP acts as a primary user and attempts to block the signal. However, their work focuses on using reinforcement learning to allow for frequency hopping instead of focusing on a mesh routing protocol that would be able to scale to more radios.

Much of the existing work done using USRPs and GNU Radio for SDR Mesh Networks revolves around implementing different parts of the Open Systems Interconnection (OSI) protocol stack from the ground up. In some papers the authors focus on the physical or mac layer [42]. There has also been work in developing new higher layer protocols for cognitive radio mesh networks such as work done to replace Transmission Control Protocol (TCP) with a more robust protocol [43]. These systems will usually react to frequency changes but some also change their topology based on power use [44].

2.3.4 What is a network protocol

A protocol is essentially a set of rules that govern the communication between two devices. A network protocol is used to define the way multiple computers can electronically send messages to one another. By standardizing protocols, computers and devices from various manufacturers can still communicate effectively [45] [34].

OSI Model

The Open Systems Interconnection (OSI) Model is an abstraction that breaks network communication into seven layers [45] [46]. Each layer has a standardized way of communicating with the layers above and below it. This indicates that at any given layer, a protocol can be exchanged for another without making any change to the other layers[45].

The first layer is the physical layer. This layers deals with the transmission and reception of unstructured raw bit streams over physical mediums. Common Layer 1 protocols include ethernet and 802.11 [47] [46]. Layer 2 is the data link layer. This layer passes structured data to Layer 1, and passes recognized packets on to layer 3 [45]. A common layer 2 protocol is Medium Access Control (MAC) [47] [46].

Layer 3, or the network layer, deals with routing information through the network. A common layer 3 protocol is Internet Protocol version 4 (IPv4) [47] [48]. Layer 4 is the transport layer. Layer 4 can be used to ensure that messages are delivered error free and in sequence. Common layer 4 protocols are Universal Datagram Protocol (UDP) and Transportation Control Protocol (TCP). The next three layers are Session, Presentation, and Application layers [45] [46]. ARCAM-Net does not make any changes to these layers so they are less pertinent to the discussion.

Layer 5, or session layer, allows for the establishment of sessions between processes running on different stations. Layer 6, or presentation layer formats the data to be presented to the application layer. The final layer, application layer, passes the data off to the user [45].

2.3.5 Common Mesh Network Protocols

There are a variety of network protocols focused on the creation of Mesh Networks. Unicast routing protocols are one family of mesh network protocols. One form of these protocols is the practice unicast routing protocol [8]. In these networks a "table" is created by continuously evaluating routes to all reachable nodes. The protocol will try to keep the table as up to date and accurate as possible [8]. When a network topology changes, the changes must be propagated throughout the network. OLSR borrows from this concept [9] [49].

Another unicast method is called reactive unicast routing.These protocols are sometimes called "on-demand" unicast routing protocols [8]. This type of routing finds routes only when desired by a source node. Only then does it initiate a routing discovery process. This can sometimes cause disconnect in mobile networks, but reduces the overhead associated with a proactive routing protocol [9].

Multicast routing protocols improve on unicast methods by making networks more robust to mobile nodes that can change their location in a network. There are three main categories of multicast networks [8]. The first is the simplest, in which packets "flood" the network. Every node that receives a network passes the data on to every node it is connected to. This causes an exponential growth in the number of packets on a network for every packet sent. The second is a proactive approach which pre-computes paths to all nodes and then routinely updates and distributes the routing table [8].

In the final method, paths are created on demand. First, a source searches the mesh for the end destination. Then it computes the shortest path and sends on this path [8]. Batman-adv is a hybrid model that borrows ideas from both multicast and unicast protocols. It is described as a "biologically inspired" network protocol [9].

Batman-adv

Batman-adv is a layer 2 mesh routing protocol developed by Open-Mesh. By running on layer 2, batman-adv becomes layer 3 agnostic. This allows most applications to utilize batman-adv without making any changes to the code. Batman-adv has been integrated into the linux kernel and can be found packaged with many popular distributions of linux [49].

Batman-adv is a proactive routing protocol aimed at using minimal resources in order to work with embedded hardware. Batman-adv uses the principle that better links will provide fast, reliable connections. Every node periodically sends out a broadcast message to inform its neighbors of its existence. The neighbors then relay this information along to the rest of the network. The link quality metric is established by detecting where these broadcast packets fail [49].

OLSR

The Optimised Link State Routing (OLSR) protocol uses a "link-state" algorithm to proactively find effcient paths between nodes. The network works by using Multi-Point Relays (MPRs) that increase network throughput by creating efficient network schemes. OLSR does this by selecting a subset of neighbouring nodes to relay data instead of every node acting as a relay. This minimies

the number of packets required to establish a routing table. Network information is shared between MPRs. Every MPR has a complete routing [49]. Batman-adv was created by Open Mesh to fix problems they saw with OLSR, namely creating an algorithm better suited to embedded system hardware [14].

802.11s

802.11s is an extension of the 802.11 standard to include capabilities for multi-hop mesh networking [50] [51]. 802.11s is a new standard that has been rewritten over the past few years. As such, it is hard to find a good comparison between 802.11s and batman-adv as many comparisons are based on older, draft versions of 802.11s.

802.11s combines AODV (Adhoc On-Demand Distance Vectoring) with a proactive protocol to create a Hybrid Wireless Mesh Protocol (HWMP) [52]. AODV keeps packet overhead small by only finding routes when a route is needed. However, the tradeoff is higher computational complexity. AODV also takes more time to build a routing table than other protocols like OLSR and Batman-adv. To improve upon the performance of AODV, 802.11s uses the proactive route generation to create a predefined path to all mesh nodes. If the node moves, or can no longer be found, then 802.11s will use AODV to find a new path [51].

The open80211s project is an open source implementation of 802.11s. Currently, it only supports the AODV protocol so benchmarks cannot fully characterize the effectiveness of this standard [51]. Testing has revealed that 802.11s has better node recovery time and lower packet loss than batman-adv. However batman-adv has much higher throughput and lower power requirements [51] [52]. 802.11s is also only compatible with 802.11 hardware. In order to implement it on an SDR, the radio would need to be configured as an 802.11 radio. Batman-adv is agnostic to layer 1 and can be configured to run on any standard such as ethernet [51]. 802.11s is also designed to work with around 32 nodes [53].

2.4   Software Defined Radio Testbeds

There are several well-known Software Defined Radio testbeds in use at different Universities. One major platform is the WARP platform from Rice University. This platform is made up of many custom components including the radio hardware itself [54]. This makes the platform very expensive and limits its adaptability for use in other research facilities.

Another platform is the Hydra platform developed at UT Austin. This platform uses GNU Radio to define PHY Layer parameters and the Click Modular Router to implement Layer 2 protocols [55]. The Hydra platform also uses USRP radios as the hardware frontend. However, Click is an older software

which, according to their website, has not had an updated release since 2011 [56]. GNU Radio has since added the Polymorphic Tree (PMT) and Message types to allow for more Layer 2 development to be done right inside GNU Radio [57].

The ADROIT project was another platform developed in conjunction with DARPA. This project relied heavily on Click and GNU Radio for much of its functionality. [58] Similarly, the University of California, Irvine and Boeing Corporation developed a testbed based on USRP Radios and GNU Radio, but they implement custom MAC layers [59].

A platform similar to ARCAM-Net is presented in [60]. However, this platform uses the Optimized Link State Routing Protocol (OLSR) which has been shown to perform poorly when compared to batman-adv. OLSR is also not truly decentralized as only certain nodes relay network information [49] Rutgers University also has a platform called the ORBIT test bed. This platform is a massive mix of over 100 nodes. However, not all nodes are equipped with SDRs. Also, ORBIT is meant to be used by other researchers, not replicated. Therefore, instructions for building the platfrom yourself are not provided. They also do not provide a working software backend, the user is expected to test their own waveforms and networking protocols [61].

2.5   State of the Practice

2.5.1   RTL-SDR

The RTL-SDR is an SDR tool discovered by the DIY and hacking community. Its original purpose is to be used as a digital TV tuner. However, it was discovered that this system could also be used to general SDR purposes. There is now a large community dedicated to using this tiny SDR to receive various different signals. Prior to the creation of the open source drivers for the RTL-SDR, the most popular devices for SDR came from USRP. The USRP devices are fantastic products, but cost at least $1,000 and can cost quite a bit more with additional features. The RTL-SDR is based on the Realtek RTL2832U chip. This device can often be purchased for between $20 and $30 [62]. The range for the RTL-SDR is typically 64 MHz to 1700 MHz, however this varies depending on which tuner the manufacturer paired with it. The authors in [62] paired the RTL-SDR with a mixer in order to lower the range all the way to DC. For this, they used the NE6062AN chip.

Starting with Release 2013b, MATLAB/Simulink now have a support package that targets RTL-SDR devices. In Simulink, the package contains a single block called "RTL-SDR Receiver." This block allows the user to tune the center frequency, change the tuner gain, set the sampling rate, and alter the frequency correction factor. The block can then output the complex envelope (IQ) of the received signal in both floating point and integer formats[63]. Due to the open

nature and low cost of the RTL-SDR, the authors in [64] propose using this as a tool set for teaching DSP and Communications principles to students.

When the cost of the system is far less than that of a textbook, it is easy to understand why this could become a valuable learning tool for many students. UC Berkley has already begun to use the RTL-SDR as one of the project assignments in their digital signal processing course. There have been efforts to use the RTL-SDR with the popular Raspberry Pi computing platform. However, at least with the B+ model there is not enough power available to process the signal. Instead, it has to be used as a TCP server that is then able to forward the data onto a more powerful computer[65]. Currently, it does not appear as though anyone has tested this with the Raspberry Pi 2 microcomputer. Other work has been done to estimate the cost savings of using a USRP in conjunction with several RTL-SDRs to replace existing DSP lab infrastructure [66].

The most unfortunate downside to using the RTL-SDR is that it is unable to transmit. However, researchers in [67] proposed a system in which a USRP SDR is used as a master device that broadcasts out to a series of slave nodes that can only listen for information. In their system they used a tool called GStreamer to pass video data into GNURadio. This information was then broadcast to multiple computers in a room running an RTL-SDR with GNU Radio. The nodes were all able to view the video stream in near real time.

2.5.2 USRP

ARCAM-Net uses the Universal Software Radio Peripheral (USRP) created by Ettus Research, a division of National Instruments[68]. Ettus also released the Universal Hardware Drivers (UHD) [3] which allow for the use of the USRP with GNU Radio. USRP radios are supported by both Mathworks [31] and GNU Radio [30]. Ettus research provides the open source USRP Hardware Drivers (UHD) to allow the devices to interface into various tools [3].

The USRP line of radios are able to transmit and receive data. The B200 and B210 operate over USB 3.0 [69]. The E310 radios are standalone systems with their own onboard processor built into the FPGA [70]. The newer E310's also come with an onboard battery, allowing them to operate in a mobile fashion [71]. The USRP line of radios is used heavily in academia. The radios form the basis of numerous testbeds [59] [55]. They are also used to validate data from simulations [42].

2.6 Industry State of the Practice

General Dynamics produces the S-Band software defined radio. This radio is used in the Space Communications and Navigation Testbed on the International Space Station. This SDR provides experimenters an opportunity to

develop and demonstrate experimental waveforms in space [72].

SDRN and CRN devices are ideal for emergency response and military operations. Various countries operate using different standards for radio transmission. SDRs and CRs allow for better coordination, as the radios can be configured to operate using a form not commonly used by a given organization. In emergency response, the parameters of the radio can also be changed to provide additional bandwidth and spectrum for the duration of the operation [73].

Harris also develops commercially available SDRs. Harris began work in 1998 using RISC based processors for their Software Defined Payloads (SDP) [12]. Since then, they have switched to FPGAs to offer greater flexibility. Harris's SDPs are deployed in numerous Space Telecommunications Radio Systems (STRSes). By using an SDR with an FPGA, Harris is able to remotely update and improve the capability of satellite communication devices [12].

2.6.1 Software Defined Networks

Software Defined Networking (SDN) is a new paradigm that moves a network's control logic to a central location and away from the underlying hardware [74]. This allows for greater flexibility in how the network is designed. The control platform can be configured to move data more predictably and with finer control over the data. In traditional IP networks, the control and data planes are tightly coupled [74]. Each router or switch will have both capabilities on board will need to be configured individually [75]. This leads to fast routing, but only if each router is configured properly. A single router performing in an undesirable way may be near impossible to detect as the administrators will need to scan each router to find the cause.

Software Defined Networking separates the control and data plane from one another [74]. The hardware devices then become simple packet forwarding elements. Forwarding decisions become flow based instead of destination based. Instead of routing tables, flow tables define how groups of packets should be treated when they reach a given piece of hardware [75]. The control logic is removed from the hardware devices and instead an SDN controller is used [74]. This controller is a software platform that runs on a server and provides the essential resources and abstractions needed to program the forwarding devices. Software applications running on top of this network controller allow for the administrators to interact with the data plane [76].

2.6.2 SDRN vs SDN

SDRNs and SDNs have a similar purpose, but take very different approaches. They both seek to increase flexibility in communication, but at different layers in the OSI model. SDRNs focus on flexibility in layer 1 and 2 [19]. SDN's focus on flexibility in layers 3 and 4 [74]. Software Defined Networking is not

a part of ARCAM-Net. However, as OSI layer abstraction is maintained, SDN platforms that are compatible with the hardware running the SDR could be used [77].

2.7 State of the Art

2.7.1 Cognitive Radio Networks

Cognitive Radio Networks represent a technological evolution from Software Defined Radio Networks [78]. Cognitive Radio Networks (CRNs) are networks made up of SDRs that are capable of sensing their environment, making decisions, and changing transmission parameters [19].In a traditional Software Defined Radio Network the waveforms and transmission protocols are flexible. However, they are predefined by the software loaded onto the computer or radio. To change operating parameters, a different program is run or the user manually alters transmission patterns. In a Cognitive Radio Network, changes can be made intelligently through the use of machine learning and artificial intelligence.

A major benefit of Cognitive Radio Networks is that they can use Dynamic Spectrum Allocation (DSA) [21]. Current wireless transmission is performed on static, licensed bands. The spectrum must be purchased through the Federal Communications Commission (FCC) and the user can then only operate on that particular frequency. With Dynamic Spectrum Allocation, radios are able to find bands of frequency that are not currently being utilized and perform their transmissions there. Once a licensed user begins transmitting on that frequency, the radios can shift to a new band and continue operation. Dynamic Spectrum Allocation helps to alleviate the problems of spectrum sharing and drastically increases the usability of each frequency band [78].

As with SDRs, Cognitive Radios are able to alter operating parameters such as transmission power, frequency, modulation type. Unlike SDRs, any of these parameters can be changed through the use of Artificial Intelligence and Machine Learning. The general cognitive functions of CRNs occur in what is called a "cognitive cycle." [73] This cycle is made up of four components: Sensing, Analysis, Reasoning, and Adaptation. In spectrum sensing and analysis, the radios are constantly scanning over a given frequency band, looking for openings. In the reasoning phase, the radio looks at the data for each frequency band and quantifies the "ideal" band to begin transmitting on. In the adaptation phase the radio switches to the new operating parameters [73].

Each radio may be capable of performing these actions on their own, or they may leverage a spectrum broker. The spectrum broker is a centralized cognitive radio that may have wider band sensing capabilities [73]. This central broker will then send commands to the other cognitive radios to dictate which frequency they should continue to transmit on. This decentralized approach is

less ideal, as it limits the scalability of the network. However, it can often be less computationally complex to deploy [73].

Cognitive Radio decision making can be made at nearly any level of the OSI model, having different effects on the overall network operation. Layer 1 decision making can be used to enable Dynamic Spectrum Sharing (DSS) [79]. As wireless technology grows, more and more spectrum is being allocated. However, the world is quickly running out of useable spectrum. Cognitive Radios and DSS allow for cooperative sharing of available frequency [79].

Primary Users (PUs), or those that have purchased a license to use the band, can operate on a given frequency [80]. If there are any gaps in their transmission, then a secondary user (SU) can quickly transmit and receive on the frequency and then switch to a new frequency before the PU begins transmitting again [80].

Layer 1 decision making can also be used to avoid congestion. If a Cognitive Radio detects there are a lot of users operating on a given frequency, then it can switch to a different channel to continue transmitting.[19] Cognitive decision making is not limited to Layer 1. Layer 2 cognition could be used to reroute the paths used to transport packets. This could be done to improve throughput or reduce energy use [81]. Layer 3 cognition could adjust signal gains if packets are being dropped too frequently [82]. Layer 4 could detect when packets are going directly to a single node, and adjust the nodes frequency so they have their own independent channel of operation [83]. In general, Cognitive Radios improve upon SDRs in that they can react to and predict changes in order to refine the operation of the network.

2.7.2   Cognitive Radio Ad-Hoc Networks

Cognitive Radio Ad-Hoc Networks (CRAHNs) are essentially mesh networks made up of intelligent software defined radios [20]. They are a specialized subset of Cognitive Radio Networks [73]. CRAHNs seek to combine the flexibility of Cognitve Radios with the distributed network routing of mesh networks. CRAHNs allow for dynamic route topology due to the ability of ad-hoc networks to connect to any and all peers within their broadcast range. The features of the radio, such as frequency and amplitude, can be adjusted to force the topology of the user's choosing. This can also be done dynamically using artificial intelligence and decision making [20].

Routing in multi-hop CRNs or CRAHNs has to deal with a number of challenges. The dynamic changing of the available spectrum can alter the topology of the network as different frequencies react to the environment in different ways. The routing metrics for CRAHNs will need to be able to adjust each time a hop is performed in order to rebuild the topology [78]. The routing protocol will also have to be able to handle the transmission delay that will occur as the nodes are changing frequency [78]. During the change, the

network will be unreachable. The routing protocol will need to be able to begin transmission again just as the network comes back online or risk dropping packets.

2.7.3 Cognitive Networks

Just as SDRNs can also be CRNs, SDNs can have cognitive features. These are simply referred to as Cognitive Networks [84]. In these networks, the central control plane can reroute packets to distribute a load more evenly throughout a network [85]. If one router is receiving much more data than the rest of the network, artificial intelligence can be used to begin to off load some of these packets to other routers. Many of the algorithms used to find empty spectrum in Cognitive Radio Networks can be applied to find more efficient routes within software defined networks [84] [85].

2.7.4 Biologically Inspired Protocols

These protocols are based on algorithms that are designed to mimic naturally occurring phenomenon. This leads to novel and efficient algorithms well suited for the distributed nature of a mesh network.

Swarm Protocols

Swarm Intelligence is a class of algorithms that are based around the social behaviors of insects. They are highly applicable to mesh networks as they are designed for distributed and decentralized problem solving. Adaptive Swarm-based Distributed Routing (Adaptive-SDR) is one type of swarm routing protocol that can be used for SDR applications. In this protocol, nodes are divided into colonies. Each node learns how to get to the other colonies, and how to get to nodes within their own colonies. When a packet traverses the network, each node forwards the data on the path to the correct colony first. Then nodes within the colony forward on to the correct node [86]. This protocol is similar to the one used by Batman-adv [9].

Hive Protocols

Hive protocols emulate the behavior of bees that forage and return to a central hive [87]. In this network structure, the network is divided into several foraging zones. To find a path to a node, a bee (in the form of a packet) will seek out a destination node. Once the packet reaches the destination, another one will be sent back to the source on the same route. When it reaches the source this packet will then communicate the distance, direction, and quality of the path between the two nodes. If multiple bees are sent out, they will then be able to communicate the optimum route to take for future packets [88].

Ant Colony Optimization Protocols

Ant Colony Optimization (ACO) Protocols are modeled off of ants cooperating to find food for their colony [89]. In this protocol, a set of "Ants" are sent out in the form of discovery packets. The first set try to find the source node within the network. The other set try to find the destination node. As the ants walk, they deposit pheromones within the network to mark the nodes they have traversed. As more ants visit a given node, the quantity of pheromones at that location will begin to increase. After a while, the optimal path between a source and destination node can be determined by following the path with the largest amount of pheromones [89].

2.7.5 Decision Making and Biologically Inspired Decision Making

At a very simple level, decision making can just be automatic gain correction (AGC). A cognitive radio with AGC may adjust its gain to its highest level until it begins to receive data. At the highest gain level, it may detect clipping and begin to lower its gain until an undistorted signal, with a good signal to noise ratio, is achieved. [72]

Genetic Algorithms can be applied to MIMO cognitive radio systems. In these CR environments, a genetic algorithm is used to optimize bandwidth, band efficiency, transmission power, data rate, and bit error rate [90]. The genetic algorithm works by using a fitness score to indicate the quality of the established link. Combinations of parameters that generate high fitness scores are allowed to "breed" into new generations of parameters. Eventually, an optimal set of parameters is discovered and the Cognitive Radio continues to transmit and receive at these settings [90].

Biologically-inspired algorithms are those that mimic nature in their execution. Biologically-inspired spectrum sharing is a major component of Cognitive Radio Networks. One such algorithm is called "BIOlogically-Inspired Spectrum Sharing" or the BIOSS algorithm [91] [92]. This algorithm uses a task allocation model based on insect colonies. The various parameters that an SDR can monitor are given response thresholds. If this threshold is passed then the radio will react and change to a new parameter. For example, if a large enough primary user began to use the spectrum the cognitive radio is transmitting on then it will switch to a new frequency [91] [92].

Another biologically inspired algorithm is used to form consensus-based spectrum sensing [93]. This algorithm is based off of flocks of birds and group decision making of other types of animals. In this scenario, each node senses the spectrum around itself. When a given node begins to sense a primary user, it lets the nodes adjacent to it know [93]. These nodes then propagate it further. Overtime, the entire network will "steer" towards a new frequency the same way a school of fish may collectively steer away from a predator.

The Ant Colony Optimization (ACO) algorithm has also been applied to dynamic spectrum access in cognitive radio networks [94] Each node explores possible bands that it can transmit in and then uses a simulated pheromone to indicate good locations to transmit on. These pheromones have a decay time to ensure that they are constantly updated. Once a band has a large number of nodes releasing pheromones on it, then the other nodes will converge on this frequency. This process can be repeated continuously to move away from primary nodes as they appear [94].

Chapter 3

Methodology

The Design of the test bed can be broken down into several subsections. The lowest layer is made up of a USRP Software Defined Radio, connected to a computer over USB. On the computer, a Flask web server is started which connects this radio to a GNU Radio Flowgraph. The Flask server then connects the GNU Radio Flowgraph to the batman-adv mesh routing protocol. The Flask web server also configures a web interface that leverages SocketIO to allow for control and monitoring of the GNU Radio flowgraph. Finally, A.L.F.R.E.D. is configured by the web server as a means of sharing data among the nodes on the network.

This configuration is shown in Figure 3.1

3.1   SDR

For ARCAM-Net, we utilized a combination of Ettus Research USRP B200 and USRP B210 SDRs. These radios are able to communicate from 70 MHz to 6 GHz and are well supported in GNU Radio using the open-source USRP Hardware Driver (UHD) provided by Ettus [3]. Their relatively low cost makes them ideal for building out larger testbeds. These serve as the radio transceiver for the current version of our platform. However, thanks to the UHD support in GNU Radio, any other USRP device will be compatible with the rest of the system, with little to no changes made to the development environment [95].

3.2   GNU Radio Flowgraph

GNU Radio utilizes programs called "Flowgraphs" to allow for graphical programming of SDR software. To implement the physical and link layers on the SDR, we utilize the Out of Tree (OOT) module gr-mac created by John Malsbury [5]. This flowgraph is an implementation of a Gaussian Minimum-Shift Keying (GMSK) or Orthogonal Frequency-Division Multiplexing (OFDM) transceiver with a mac layer protocol called "simple mac". There are two main blocks in the flowgraph. The first sets up the GMSK or OFDM radio. This hierarchical block is built by running a separate flowgraph which contains the
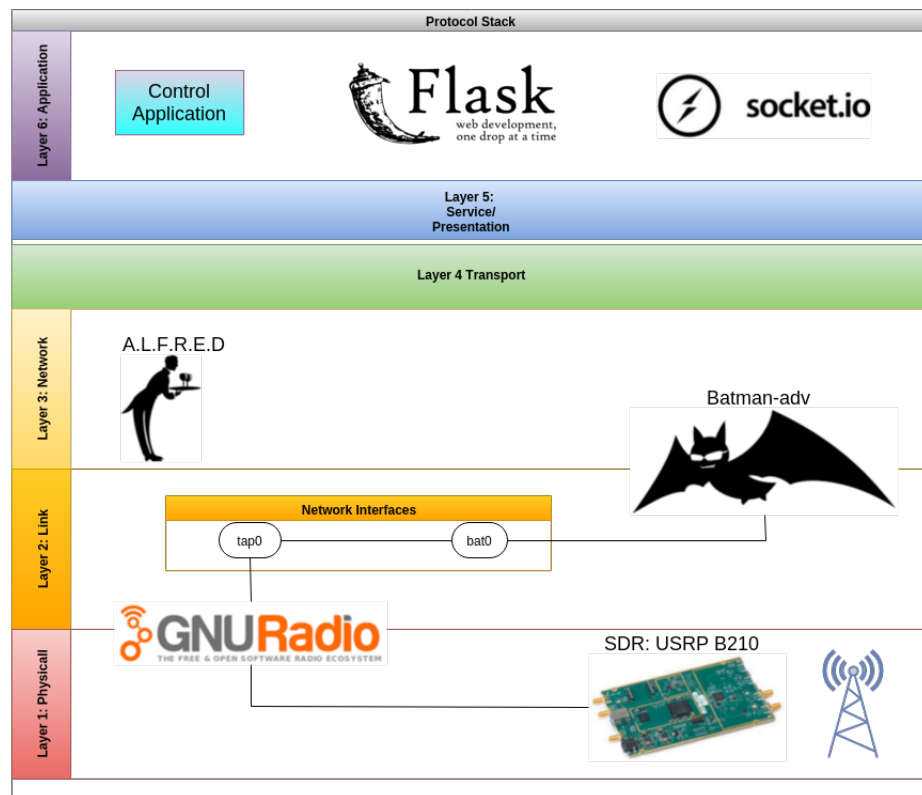
FIGURE 3.1: An overview of the components of the system, including the OSI Layers they interact with. [1] [2] [3] [4] [5] [6] [7]

UHD blocks to interface into the USRP as well as the modulation and demodulation blocks for the waveform. One of the more important aspects of the two radio blocks, is that they convert from streaming data to message data.

Most features of GNU Radio work on streaming data where there is constant data transmission. However, packets are not sent continuously, therefore separate logic is needed to convert streams to messages. These messages are passed into and out of the GMSK and OFDM hierarchical blocks, so the remainder of the flowgraph deals with passing messages only [95].

We use the GMSK block to convert from streaming data to message data and then connect this block to a tunnel (TUN) or network tap (TAP) interface block. TUN/TAP devices are virtual network kernel devices supported entirely in software. TUNs are used to simulate layer 3 devices and TAPs simulate layer 2 [96]. Either of these could be selected to suit the user's purpose, but as batman-adv is a layer 2 protocol, we will use the TAP protocol.

3.3 Batman-adv

Batman-adv was chosen based on its large community and documented success as a mesh routing protocol [49]. It is already included as part of the Linux kernel, and additional software can be downloaded from most distributions repositories [4]. Configuring batman-adv to work on the SDR involves running the program batctl and selecting the recently generated TAP interface created by GNU Radio. The Maximum Transmission Unit (MTU) of the TAP interface must also be changed to 1532 from 1500 in order to incorporate the additional header batman-adv uses when sending data. Using only batman-adv and GNU Radio, we are able to create a Software Defined Radio based mesh network. The remainder of the test bed was implemented to leverage features unique to GNU Radio and batman-adv to create a method of sharing frequency and other data.

3.4 Flask Web Server and Socket.IO

Flask is a lightweight, open source, web framework for the Python programming language [7]. Flask was used to act as a broker between GNU Radio and any other user space applications or control systems we wished to implement. The Flask server runs the GNU Radio flowgraph in a background thread, while simultaneously configuring the TAP interface, setting up batman-adv, and starting A.L.F.R.E.D. as a background process.

Socket.IO is a JavaScript library that enables real-time bidirectional event-based communication [6]. SocketIO was chosen as a means of relaying data between the Flask server and other components of the system due to its speed, flexibility, and ability to broadcast messages to any connected client. Socket.IO also integrates into Flask [97] and can be used in stock Python with a client
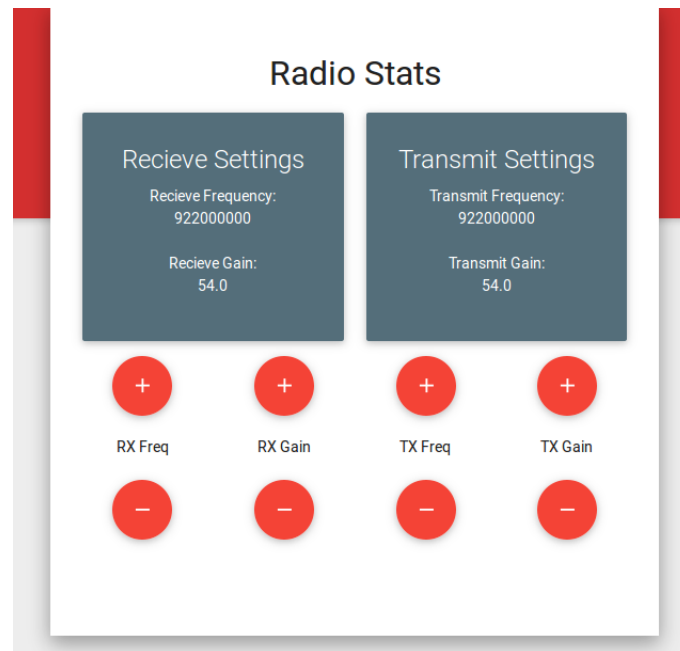
FIGURE 3.2: The web interface that lets the user initiate the network to hop to a new frequency. [1]

library [98]. In Flask, we create wrappers to all the necessary GNU Radio parameters so that external tools can relay data to and from GNU Radio over web sockets.

We also use Flask to host a single webpage that displays various settings about the radio, and allows for the user to change parameters. The interface is shown in Figure 3.2. Since our platform does not yet include logic for automatic detection of primary users, we simulate this by allowing a person to click a button to change to a new frequency. This frequency will then be sent to the Flask server using web sockets.

3.5 A.L.F.R.E.D.

The "Almighty Lightweight Fact Remote Exchange Daemon," or A.L.F.R.E.D., is a system for distributing data to all nodes on a mesh network [5]. Whenever a node writes data to a channel on A.L.F.R.E.D., that data is passed between each node so that all members of the network receive the data. Typical uses for A.L.F.R.E.D. include keeping track of sensor data or building a visual map of the network.

An additional feature of A.L.F.R.E.D. is its ability to pass a command to the command line whenever new data is added [5]. When the transmission frequency of the USRP is changed on the Flask server, Flask sends this information along with a UTC timestamp to A.L.F.R.E.D. before changing frequencies.

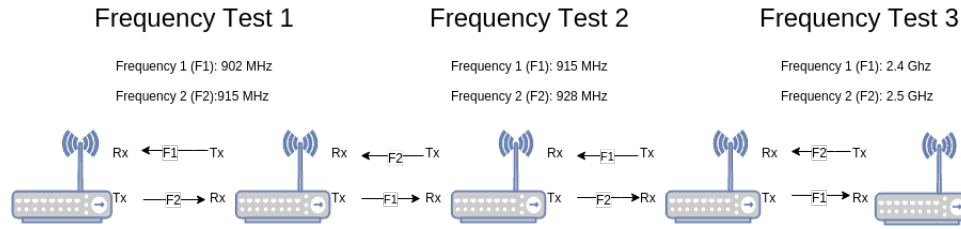FIGURE 3.3: The configuration used for the first set of tests. [1]

A small delay is created so that we can be sure the information was sent to the other nodes before the node changes its broadcast frequency.

When the other nodes receive the updated data table, A.L.F.R.E.D.'s callback function will run. This is a short program that parses the A.L.F.R.E.D. data table and looks for the most recent data it received. The callback function then sends the new frequency to Flask using Socket.IO which causes Flask to change the frequency in the GNU Radio flowgraph.

3.6 Fabrication and Build Process

The network was setup and run on a series of Lenovo S30 Computers. All computers features an Intel Xeon E5 processor and 32 GB of RAM. Each computer was running Ubuntu 14.04 LTS. The install process for GNU Radio, Batman-adv, and the rest of the technology stack is detailed in the appendix. Each computer was connected to one Ettus B200/B210 SDR using USB 3.0.

All of the computers were connected together on a 10/100/1000 Ethernet router. This allowed for the control of the entire mesh to be handled by one computer. The popular terminal multiplexer Tmux was used in conjunction with SSH to send commands to all of the nodes in parallel.

An additional Windows 10 laptop computer was used to control a Tektronix RSA306 wireless spectrum analyzer. This computer had an Intel i7 processor, and 8 GB of RAM. It is important to note that the RSA 306 requires a Windows machine, and therefore cannot be run on the same computer as the ones being used to run ARCAM-Net. This spectrum analyzer was used to confirm the operation of the radios and that frequency changes were occurring [99].

3.7 Test Procedures

In order to characterize the platform we ran three sets of tests presented below. The first test characterizes data hopping from one node to the next. The second test demonstrates batman-adv's ability to switch routes based on the quality
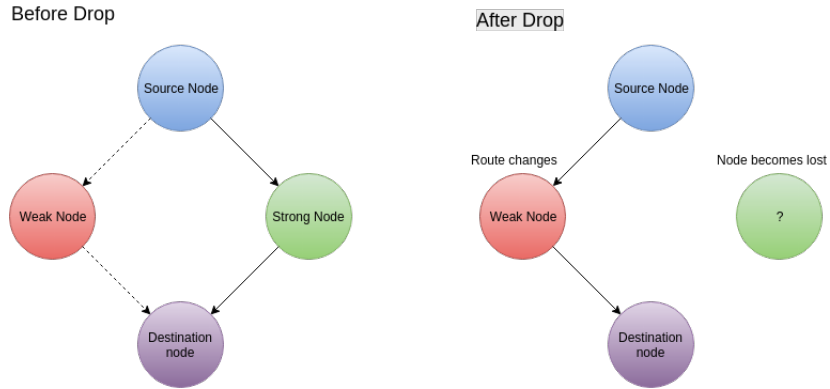
FIGURE 3.4: The configuration used for the second set of tests.
[1]

of each node. The final tests were used to examine A.L.F.R.E.D.'s ability to be used for exchanging frequency information from node to node.

3.7.1 Network Benchmarks

The first test was used to investigate the overhead each node adds to the network. To examine how adding hops affects the network, the USRPS were arranged so that each node was only able to directly connect to one or two neighboring nodes. This forced the nodes to need to use the multihop features of batman-adv in order to connect to the rest of the network. We used a total of 5 nodes as shown in Figure 3.3. We staggered the transmit and receive frequencies of the nodes to ensure that nodes could only talk to their immediate neighbors, forcing the network into the previously indicated configuration. The staggering of frequencies was needed to ensure each node would be unable to communicate to nodes other than its neighbors.

We then tested the setup at three different sets of frequencies, all within the Industrial, Scientific, and Medical (ISM) Band. We used different sets of ping tests in order to determine the number of dropped packets and the time it took to send the packets. We ran a standard ping test, one with reduced packet sizes, and one with increased time to live (TTL) settings. For a control group, two USRPs were connected together without batman-adv running.

3.7.2 Route Changes

In a typical mesh environment, there will usually be more than one route from a source to a destination [20]. In a traditional network, batman-adv switches routes based on the quality of each available link. The test was designed to determine if the same features would work in an SDRN. We initially set up four SDRs: a source, a destination, and two nodes to connect them. The first node is given a significantly larger gain in order to confirm that batman-adv

will recognize that this is a more ideal path to the destination. Then, once the route is placed into the routing table, we lower the gain to 0 in order to force a transition to the other node. We then confirm that batman-adv is still able to find the new route. This setup is shown in Figure 3.4.

3.7.3 Frequency Distribution

In the final test, we tested to evaluate if A.L.F.R.E.D. would properly relay frequency changes over the mesh to other nodes. The user would increase or decrease the frequency using the web interface in order to simulate a cognitive radio making a decision to change to a new frequency. If A.L.F.R.E.D. was able to exchange the information properly, then the routing table would still show all connected nodes. We also used a Tektronix RSA306 Spectrum Analyzer in order to conclude that the transmission was occurring on the new frequency. If A.L.F.R.E.D. was not able to relay the information to all nodes, then some would change to the new frequency while others remain. This would be reflected in the output from the spectrum analyzer.

Chapter 4

Findings

This Chapter presents the results of the tests discussed in 3. Also included is a discussion of applications that were successfully run over the network. Finally, an analysis of the results, and a discussion of error is presented.

4.1   Results

4.1.1   Network Benchmarks

The results of the Network Benchmark tests from section 4 part A are summarized in Figure 4.3. In all cases, the single point to point communication, without the batman-adv protocol running, resulted in a much lower packet loss. This served as our control group. However, in the two sets of lower frequency ratings, the packet loss remained below 50%. Also, the increase in time as hops were added has a roughly linear change. This indicates the overhead of adding more hops is not unmanageable. A full listing of tests run for the 902/915 MHz, and 2.4/2.5 GHz cases are provided in Figure 4.1 and Figure 4.2 respectively. These tables show that running at the higher frequencies causes the SDRN to drop a lot more packets, especially when moving through the full four hops.

4.1.2   Route Changes

The route changing feature of batman-adv showed success with our test setup from section 4 part B. Initially, batctl reported two possible links, one with a transmit quality of 55, and the other with a transmit quality of 18. As we decreased the gain of the intermediate node, the transmit quality reported by batctl also decreased. Eventually, batman-adv switched and began using the other node. At this point, it no longer saw the original node, and reported a transmit quality of 75 on the alternate one. The initial setup can be seen in Figure 4.4. After the change, the routing table appeared as it does in Figure 4.5. This feature works in the SDR system, and can continue to be used without significant changes.

| Packet Loss | | STD Ping | PS | | TTL | | AVG |
Hops	Batman	AVG 3 Tests	24	32	128	255	
1	No	1%	1%	1%	2%	1%	1%
1	Yes	16%	12%	9%	12%	7%	11%
2	Yes	26%	18%	19%	21%	12%	19%
3	Yes	30%	22%	17%	35%	28%	26%
4	Yes	42%	41%	36%	48%	60%	45%

| Time (ms) | | STD Ping | PS | | TTL | | AVG |
Hops	Batman	AVG 3 Tests	24	32	128	255	
1	No	15.25	11.89	12.39	14.57	14.72	13.76
1	Yes	17.63	14.81	14.91	19.13	13.38	15.97
2	Yes	35.36	30.08	29.8	34.62	35.14	33
3	Yes	52.44	44.67	48.49	54.01	54.72	50.87
4	Yes	69.51	58.81	59.54	67.17	76.45	66.3

FIGURE 4.1: The data received from operating at 902/915 MHz.

| Packet Loss | | STD Ping | PS | | TTL | | AVG |
Hops	Batman	AVG 3 Tests	24	32	128	255	
1	No	1%	0%	0%	1%	1%	1%
1	Yes	12%	6%	5%	7%	11%	8%
2	Yes	18%	17%	16%	18%	12%	16%
3	Yes	28%	23%	17%	16%	22%	21%
4	Yes	55%	58%	66%	70%	72%	64%

| Time | | STD Ping | PS | | TTL | | AVG |
Hops	Batman	AVG 3 Tests	24	32	128	255	
1	No	14.98	12.42	12.44	15.07	15.55	13.87
1	Yes	18.44	16.11	15.31	17.23	18.51	16.79
2	Yes	35.02	29.05	29.57	33.39	34.46	31.62
3	Yes	53.53	45.01	46.5	56.05	55.72	50.82
4	Yes	68.27	57.76	61.02	65.72	65.49	62.5

FIGURE 4.2: The data received from operating at 2.4/2.5 GHz.

Packet Loss		Frequency		
Hops	Batman	902/915 MHz	915/928 MHz	2.4/2.5 GHz
1	No	1%	3%	1%
1	Yes	12%	12%	8%
2	Yes	21%	23%	16%
3	Yes	27%	30%	21%
4	Yes	44%	32%	64%

Time (ms)		Frequency		
Hops	Batman	902/915 MHz	915/928 MHz	2.4/2.5 GHz
1	No	13.76	15.38	13.87
1	Yes	15.97	18.21	16.79
2	Yes	33	32.07	31.62
3	Yes	50.87	50.63	50.82
4	Yes	66.3	63.5	62.5

FIGURE 4.3: The Averages from all three tests.

```
[B.A.T.M.A.N. adv 2016.0, MainIF/MAC: tun0/3e:f1:55:1d:9f:e1 (bat0 BATMAN_IV)]
  Originator      last-seen (#/255)        Nexthop [outgoingIF]:   Potential nexthops ...
b2:29:cf:60:12:f5    2.056s   ( 55) 26:b3:c6:bd:44:68 [      tun0]: 96:1b:4a:28:73:f8 ( 18) 26:b3:c6:bd:44:68 ( 55)
```

FIGURE 4.4: The initial condition from the test presented in Section 4 B, where there are two possible routes the packet can take. The output is from running batctl. The transmit quality (TQ) is listed under the column labeld (#/255). A higher number represents a better quality connection.

```
[B.A.T.M.A.N. adv 2016.0, MainIF/MAC: tun0/3e:f1:55:1d:9f:e1 (bat0 BATMAN_IV)]
  Originator      last-seen (#/255)        Nexthop [outgoingIF]:   Potential nexthops ...
b2:29:cf:60:12:f5    0.908s   ( 75) 96:1b:4a:28:73:f8 [      tun0]: b2:29:cf:60:12:f5 (  0) 96:1b:4a:28:73:f8 ( 75)
```

FIGURE 4.5: After the gain is reduced in the test presented in Section 4 B, the packets are now routing through a different node.
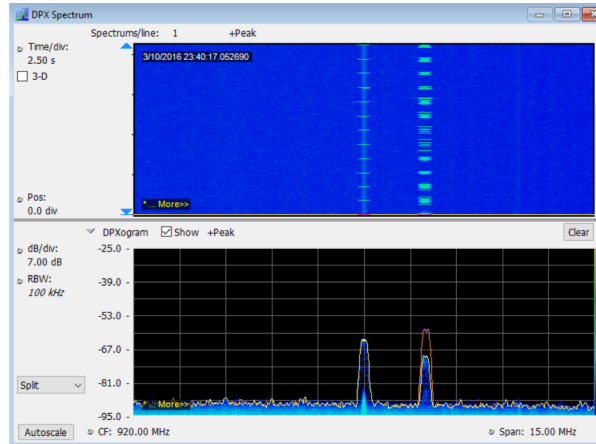
FIGURE 4.6: The result of using ALFRED to shift frequencies. One node is left behind as the others move to the new channel. Image collected using the Tektronix RSA306 Spectrum Analyzer

The transmit quality (TQ) metric is used by batman-adv to choose which route it will take next [100]. This is an integer number from 0 to 255 where 0 represents a dead node. As a packet traverses a batman-adv network, these numbers are updated to continuously find the best path for each future packet to take [100].

4.1.3 Frequency Changes

Using A.L.F.R.E.D. to distribute frequency hopping showed mixed results. Using the setup from section 4 part C, we were able to get the nodes to change frequency in unison, but not reliably. A.L.F.R.E.D. itself is designed for a traditional Wi-Fi environment, and therefore does not have an expectation that the other nodes will become completely unreachable due to changes in operating frequency [5]. We were finding that some nodes would switch before A.L.F.R.E.D. had propagated the data table to the other nodes. This would leave one node with an out-of-date table, meaning it would not make the frequency change. In the current iteration of the project, there is no way for these orphaned nodes to find the rest of the network again. Figure 4.6 shows a situation in which four out of five nodes were able to make the jump, with one node remaining at the original frequency.

4.2 Applications

Several application level programs were run over the network. Each application was run in both a single and multihop setting. Qualitative measurements were recorded as it was difficult to make quantitative measurements about the effectiveness of the network. The first test was run using Secure Shell (SSH).

On one computer, a user would use ssh to connect to a remote computer and run a few commands to confirm operation. Next, Secure Copy (SCP) was used to transmit a file from one computer to another. A simple chat application was built in both Python and Erlang to demonstrate the ability to communicate over the network.

4.3 Data Analysis

In an ideal situation, the test would show minimal packet loss despite introducing batman-adv and multihopping into the network. Additional tests will need to be run to discover what causes the 1 hop with batman-adv to drop many more packets than 1 hop without batman-adv. It is possible the packets are dropped as routing table information is shared, but this has not been confirmed yet.

It is important to notice that packet loss and round trip time of packets increases linerally with each hop. This is good, as it indicates each hop adds a finite amount of delay and packet loss. It would be unlikely that the delay would be constant when adding more hops, so a linear time increase is good.

Another interesting issue with this test was that we had to stagger the operating frequencies to force the straight line topology we were looking for. Batman-adv worked very well at finding alternative paths to each node in the network. However, if the tests were left running for long periods of time it was possible the network would have been able to reconfigure into a topology other than a straight line chain. Therefore, it was necessary to manually ensure that the network would stay in the desired configuration in order to maintain a valid test.

The route changing feature of batman-adv worked fine with the SDR hardware. Initially, there was uncertainty if batman-adv would be able to compute a proper metric for the link quality of each node. This link quality metric is what batman-adv uses to determine the best "next hop" to reach a different node. However, our tests showed the routing table could be properly built and updated as link quality decreased.

Our tests on A.L.F.R.E.D. showed that it was possible to exchange frequency information over the network. However, the tests revealed that this was a very unreliable method. There is no way to ensure that A.L.F.R.E.D. had shared changes to its data table to all other nodes on the network. This meant that some nodes would change their frequency while others would remain at the older frequency. In some CRAHNs, the nodes act as secondary users and transmit on a channel when it is unoccupied. If this was being used to avoid a primary user, a long delay would not be acceptable as it would interfere with the PU's use of the spectrum. Therefore, a new implementation of this feature will need to be produced.

4.4   Assumptions and Discussion of Error

One assumption is that the straight line configuration shown in Figure 3.3 reflects how the network would behave in a full scale environment. Here we are using this configuration to determine the overhead associated with each additional hop. However, in a more robust network configuration it is likely there will be more than one path to any given node. These additional paths may be necessary to lower the overall packet loss of the network. It may be more appropriate to test this type of configuration in simulation rather than in the real world.

We also did not test these setups in an anechoic chamber. Therefore, it is not possible to know how reflections influenced our data set. Reflections of RF waveforms can cause both constructive and destructive interference. This indicates that the reflections could alter the signal quality at any given node.

The experiments were performed over a 24 hour period. Therefore, some tests were performed while the lab was actively being used by other researchers and some were performed while the room was empty. This again could have impacted the results. The reflectivity of the room is altered with people present. Also, the overall level of RF noise will be higher with people present as they will likely be using cell phones and computers with wireless capabilities.

A uniform operating pattern in each node and in each antenna is assumed. Without the necessary equipment, it is impossible to truly characterize the antennas or transmission patterns of the SDRs. This is not likely to be a source of error based upon the information in the data sheets, but it is something to be aware of.

Chapter 5

Discussion

This chapter focuses on discussing what was discovered from the creation and testing of ARCAM-Net. The limitations of the network are provided for reference. Applications that were run on the network are discussed as well.

5.1 Constraints and Limitations

The results from our experiments show that the network is functioning as a multi hop SDRN. In addition to the experiments performed, we were also able to use Secure Shell (SSH) and Secure Copy (SCP) over multiple hops of the SDRN. However, it is clear that more work needs to be done. In a deployed network, packet loss as high as is seen in this network is not optimal. Therefore, it is important to examine ways to mitigate packet loss and increase throughput.

For example, machine learning or artificial intelligence algorithms could be used to adjust transmission parameters as issues are detected. It is likely that a change in frequency or amplitude could mitigate some of the packet loss. For example, if the loss is due to crowding near one node, frequency hopping could be employed to shift to better operating conditions. Batctl's link quality metric, as shown in Figures 4.4 and 4.5 could help with finding weak nodes and making decisions. This would be the beginning of ARCAM-Net's transition from a SDRN to a Cognitive Radio Network (CRN).

Furthermore, it would be beneficial to either improve upon A.L.F.R.E.D. or implement certain features in a new way in order to handle the frequency changing. If we have each node wait for an acknowledgment from its immediate neighbors before changing frequency, that node could then change its frequency knowing that the data will propagate to the rest of the network.

Batctl is able to report the immediate next hop neighbors, so the program could use this information to only wait for acknowledgment from neighbors instead of waiting for the entire network to be ready to change. In order for the current A.L.F.R.E.D. setup to function, a delay was needed to give the network time to respond. Therefore, an asynchronous acknowledge would likely speed up the frequency change.

5.2 SDR Network

Our tests show that ARCAM-Net is a fully functioning SDR Network. We have successfully run ping tests, sent UDP packets, used SSH, used SCP, and even ran distributed Erlang programs over the SDRN. Thanks to the way the OSI model abstracts each layer, all programs that are designed to operate on Layer 3 or higher should see ARCAM-Net as a normal network. Though in its current state it is slower than a traditional WiFi network, there is nothing preventing any normal program from using the SDRN in the way it would use a Wi-Fi or LAN.

5.3 Cognitive Networks

ARCAM-Net is not a cognitive network. However, it is certainly adaptable to be used for cognitive network testing. Developing cognitive algorithms was beyond the scope of this project. Therefore, the concept was broken down into two parts: The decision making itself, and the propagation of this decision to the rest of the network. A.L.F.R.E.D. was chosen to implement the decision propagation and the web interface is meant to be used as a tool for simulating the cognitive decision making. The web interface enables the user to increase or decrease the frequency of operation. This was used to simulate the nodes themselves having cognitive decision making abilities. Using SocketIO, the node can transmit the frequency the Flask server. The Flask server will then update GNU Radio with the new frequency. This way, the logical decision making can be abstracted from the standard operation of the SDR.

In our testing we determined A.L.F.R.E.D. is not a good fit for transmitting frequency data. But we can use whatever system eventually replaces A.L.F.R.E.D. as part of the cognitive network toolchain. When a node makes a decision to change frequency, it can use this tool to distribute the information to the rest of the network.

5.4 Applications

All of our application layer tests showed that programs could be run over the network. SSH was able to run commands from computer to computer. SCP was used to successfully send data to and from different computers over the network. The python and erlang chat applications worked successfully and were able to transmit messages. Erlang itself is a programming language aimed at working in distributed systems and presents a unique opportunity for future work.

Chapter 6

Conclusion

6.1 Summary

ARCAM-Net allows for researchers to create a fully functioning SDRN in a short period of time. In it's current form, ARCAM-Net's auotmated processes could scale to up to 255 simultaneous nodes. More nodes could be added with only a few changes to the start scripts. ARCAM-Net is designed to bridge the Open-Mesh and GNU Radio projects while providing researchers with the tools they need to design, experiment, and test.

6.2 Conclusion

ARCAM-Net is a Software Defined Radio Network (SDRN) testbed and platform. ARCAM-Net was born out of a need for a platform to develop SDRN applications. At the start of the project, no readily available system could be identified that would allow for user space applications to be run on an SDRN. Thus work began on the creation of ARCAM-Net. ARCAM-Net creates a fully functioning data network capable of acting as a multi-hop mesh network.

Both Batman-adv and GNU Radio represent next generation, open source, wireless solutions and each organization could likely benefit from collaboration between the two projects. This work can be used as an excellent starting point for anyone looking to get an SDRN up and running quickly to begin prototyping other sections of the tool chain. We plan on releasing the code as well as a handful of tools to help other researchers get started. This work can serve as the first step in creating an equivalent Cognitive Radio Network platform and testbed.

This work represents a first step in a longer term project to create a fully functioning SDRN. The work demonstrates the potential for using GNU Radio in conjunction with batman-adv and other Open-Mesh solutions. As the work continues forward, we intend for the testbed to serve as a collaboration point between GNU Radio and Open-Mesh.

We were able to show that packets can successfully traverse the network. Packet loss increased linearly with each additional node the packet traversed.

Initially, packet loss was near 95%, but was brought down with changes to the flowgraphs. We were also able to prove batman-adv's multihop and rerouting features worked properly in the Software Defined Radio Network. The attempts to use A.L.F.R.E.D. as a means of sharing frequency data proved to have issues. The information could be passed from node to node, but would not reliably reach all nodes on the network.

Several applications were run over the network. Secure Shell was used to run commands remotely on another computer. Secure Copy was also used to transfer files from one machine to another. Both of these commands would work with more than one hop in between the nodes. A chat application was developed in Python using UDP packets. Nodes could broadcast text messages to the other machines on the network using the broadcast address. Finally a short, distributed program was developed in Erlang. This program was able to send commands to a process on a different machine, over the Software Defined Radio Network.

Several tools were also generated for use with the network. The Flask Server automatically setup the GNU Radio Flowgraph and then configured the batman-adv network on that node. Additonally, a script was generated to allow for all nodes on the network to be setup with just a single command. Finally, an install script was generated that could create either a full install or a virtual machine. This would install all dependencies and then configure GNU Radio and Batman-adv.

6.3   Recommendation

ARCAM-Net should reused and extended upon by future research efforts. It is important to continue to document and project and ensure easy access to the documentation to outside parties. As with any platform, the value is in numerous groups working together on a standard set of tools.

Moving forward, the GMSK blocks can be replaced with different protocols. Having the ability to switch between different protocols, such as OFDM, would further show the flexibility of ARCAM-Net. The same is true of batman-adv. Integrating different protocols or including an SDN tool such as Open-Flow would make ARCAM-Net more extensible and allow control to continue to the upper layers. Finally, allowing the network to be remotely accessible would allow for other researchers to utilize the platform even if they lack equipment of their own.

6.4   Future Work

There are many areas of this project that can serve as starting points for new projects. First, A.L.F.R.E.D. needs to be replaced by something more robust. A.L.F.R.E.D. could continue to be used to store distributed data that will not

change often, but should not be used to store frequency data. Instead, a new tool will need to be create that can safely propagate data to other nodes, and wait for an acknolwedgement of the received information. This tool will be useful in creating the next major revision to ARCAM-Net.

As stated previously, ARCAM-Net is a SDRN, not a CRN or CRAHN. In order to move from an SDRN to a cognitive radio environment, it will be necessary to begin to implement artifical intelligence and decision making in some form. The decision making could be split Layer 1 and Layer 3 decision making. On Layer 1, DSP would be employed to look for PU's, noise and distortion on incoming packets, overcrowded networks, and other issues. Once these problems are deteced the system will then switch to a new frequency. On Layer 3 the nodes will look for excessive dropped packets, poor batman-adv link quality, orphaned nodes, and other factors. The system will then adjust gains and frequencies until this layer has been optimized.

While running experiments, batctl was used to monitor the quality of each node. Though batctl works fine for a few nodes, as the network scales it would make finding issues and bottle necks more difficult. Another great way to improve upon the current project would be to implement a more interactive environment for monitoring the network. A central "control" server could be created to monitor all of the nodes on the network and display information in a 2D or 3D environment. This would make network monitoring more organized, as batctl produces different data on each node and therefore must be monitored on many computers simultaneously.

Bibliography

[1] J. McCormack, B. Trowbridge, J. Prine, R. C. Maden, and R. Integlia, "Arcam-net: A software defined radio mesh network testbed implemented in gnu radio with batman-adv," in *IFIP Wireless and Mobile Networking Conference*, July 2016, p. PENDING REVIEW.

[2] (2016) Welcome to gnu radio. [Online]. Available: http://gnuradio.org/redmine/projects/gnuradio/wiki

[3] (2016) Uhd (usrp hardware driver). [Online]. Available: https://www.ettus.com/sdr-software/detail/usrp-hardware-driver

[4] (2016) Open-mesh. [Online]. Available: https://www.open-mesh.org/projects/open-mesh/wiki

[5] (2014) A.l.f.r.e.d - almighty lightweight fact remote exchange daemon. [Online]. Available: https://www.open-mesh.org/projects/alfred/wiki

[6] (2016) Socket.io. [Online]. Available: http://socket.io/

[7] (2016) Flask (a micro framework). [Online]. Available: http://flask.pocoo.org/

[8] L. Junhai, Y. Danxia, X. Liu, and F. Mingyu, "A survey of multicast routing protocols for mobile ad-hoc networks," *IEEE Communications Surveys Tutorials*, vol. 11, no. 1, pp. 78–91, First 2009.

[9] A. Sharma and N. Rajagopalan, "A comparative study of b.a.t.m.a.n. and olsr routing protocols for manets," *International Journal of Advanced Trends in Computer Sceince and Engineering*, vol. 2, no. 5, pp. 13–17, tenth 2013.

[10] J. Mitola, "Software radio architecture: a mathematical perspective," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 514–538, Apr 1999.

[11] ——, "The software radio architecture," *IEEE Communications Magazine*, vol. 33, no. 5, pp. 26–38, May 1995.

[12] A. W. Mast, "Reconfigurable software defined payload architecture that reduces cost and risk for various missions," in *Aerospace Conference, 2011 IEEE*, March 2011, pp. 1–5.

[13] D. F. Macedo, D. Guedes, L. F. M. Vieira, M. A. M. Vieira, and M. Nogueira, "Programmable networks from software-defined radio to software-defined networking," *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 1102–1125, Secondquarter 2015.

[14] O. Mesh. (2014) B.a.t.m.a.n. advanced. [Online]. Available: https://www.open-mesh.org/projects/batman-adv/wiki/Wiki

[15] D. Seither, A. Konig, and M. Hollick, "Routing performance of wireless mesh networks: A practical evaluation of batman advanced," in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, Oct 2011, pp. 897–904.

[16] L. C. Tran, D. T. Nguyen, F. Safaei, and P. J. Vial, "An experimental study of ofdm in software defined radio systems using gnu platform and usrp2 devices," in *2014 International Conference on Advanced Technologies for Communications (ATC 2014)*, Oct 2014, pp. 657–662.

[17] E. G. Sierra and G. A. R. Arroyave, "Low cost sdr spectrum analyzer and analog radio receiver using gnu radio, raspberry pi2 and sdr-rtl dongle," in *2015 7th IEEE Latin-American Conference on Communications (LATIN-COM)*, Nov 2015, pp. 1–6.

[18] P. Alvarez, N. Pratas, A. Rodrigues, N. R. Prasad, and R. Prasad, "Energy detection and eigenvalue based detection: An experimental study using gnu radio," in *Wireless Personal Multimedia Communications (WPMC), 2011 14th International Symposium on*, Oct 2011, pp. 1–5.

[19] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks*, vol. 51, no. 4, pp. 921 – 960, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128606002751

[20] I. F. Akyildiz, W.-Y. Lee, and K. R. Chowdhury, "Crahns: Cognitive radio ad hoc networks," *Ad Hoc Networks*, vol. 7, no. 5, pp. 810 – 836, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S157087050900002X

[21] K. K. Chauhan and A. K. S. Sanger, "Survey of security threats and attacks in cognitive radio networks," in *Electronics and Communication Systems (ICECS), 2014 International Conference on*, Feb 2014, pp. 1–5.

[22] V. L. Martin Braud, Ben Hilburn. (2014) What is gnu radio and why do i want it. [Online]. Available: http://gnuradio.org/redmine/projects/gnuradio/wiki/WhatIsGR

[23] I. Poole. Software defined radio sdr hardware. [Online]. Available: http://www.radio-electronics.com/info/rf-technology-design/sdr/software-defined-radios-hardware-architecture.php

[24] W. I. Forum. What is software defined radio. [Online]. Available: http://www.wirelessinnovation.org/assets/documents/SoftwareDefinedRadio.pdf

[25] G. Radio. (2014) Volk. [Online]. Available: https://gnuradio.org/redmine/projects/gnuradio/wiki/Volk

[26] ——. (2014) Ubuntu install. [Online]. Available: https://gnuradio.org/ redmine/projects/gnuradio/wiki/UbuntuInstall

[27] ——. (2016) Gnu radio companion. [Online]. Available: https://gnuradio.org/redmine/projects/gnuradio/wiki/UbuntuInstall

[28] ——. (2016) Out of tree modules. [Online]. Available: https://gnuradio.org/redmine/projects/gnuradio/wiki/OutOfTreeModules

[29] CGRAN. (2016) The comprehensive gnu radio archive network. [Online]. Available: http://cgran.org/

[30] G. Radio. (2016) A quick guide to hardware and gnu radio. [Online]. Available: http://gnuradio.org/redmine/projects/gnuradio/ wiki/Hardware

[31] Mathworks. Software-defined radio (sdr). [Online]. Available: http://www.mathworks.com/discovery/sdr.html? requestedDomain=www.mathworks.com

[32] Ettus. Rfnoc (rf network on chip). [Online]. Available: https://www. ettus.com/sdr-software/detail/rf-network-on-chip

[33] O. Spaniol, "The evolution of wireless networking: conformance or conflict with the osi reference model?" in *Personal, Indoor and Mobile Radio Communications, 1994. Wireless Networks - Catching the Mobile Future., 5th IEEE International Symposium on*, vol. 3, Sep 1994, pp. 786–793 vol.3.

[34] D. R. Winkelman. (2013) What is a protocol? [Online]. Available: http://fcit.usf.edu/network/chap2/chap2.htm

[35] M. Eslami, O. Karimi, and T. Khodadadi, "A survey on wireless mesh networks: Architecture, specifications and challenges," in *Control and System Graduate Research Colloquium (ICSGRC), 2014 IEEE 5th*, Aug 2014, pp. 219–222.

[36] (2015) Confine project. [Online]. Available: http://confine-project.eu/

[37] P. Escrich, R. Baig, A. Neumann, A. Fonseca, F. Freitag, and L. Navarro, "Wibed, a platform for commodity wireless testbeds," in *Wireless Days (WD), 2013 IFIP*, Nov 2013, pp. 1–3.

[38] P. Escrich, R. Baig, E. Dimogerontakis, E. Carbo, A. Neumann, A. Fonseca, F. Freitag, and L. Navarro, "Wibed, a platform for commodity wireless testbeds," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on*, Oct 2014, pp. 85–91.

[39] R. Alimi, L. Li, R. Ramjee, H. Viswanathan, and Y. Yang, "ipack: in-network packet mixing for high throughput wireless mesh networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008, pp. –.

[40] L. Li, R. Alimi, R. Ramjee, H. Viswanathan, and Y. Yang, "munet: Harnessing multiuser capacity in wireless mesh networks," in *INFOCOM 2009, IEEE*, April 2009, pp. 2876–2880.

[41] A. Syed, K. Yau, H. Mohamad, N. Ramli, and W. Hashim, "Channel selection in multi-hop cognitive radio network using reinforcement learning: An experimental study," in *Frontiers of Communications, Networks and Applications (ICFCNA 2014 - Malaysia), International Conference on*, Nov 2014, pp. 1–6.

[42] P. Nagaraju, L. Ding, T. Melodia, S. Batalama, D. Pados, and J. Matyjas, "Implementation of a distributed joint routing and dynamic spectrum allocation algorithm on usrp2 radios," in *Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on*, June 2010, pp. 1–2.

[43] H. Khalife, J. Seddar, V. Conan, and J. Leguay, "Validation of a point to multipoint cognitive radio transport protocol over gnu radio testbed," in *Wireless Days (WD), 2013 IFIP*, Nov 2013, pp. 1–6.

[44] S. Kamruzzaman, A. Alghamdi, and S. Mizanur Rahman, "Spectrum and energy aware multipath routing for cognitive radio ad hoc networks," in *Information and Communication Technology Convergence (ICTC), 2014 International Conference on*, Oct 2014, pp. 341–346.

[45] S. N. Orzen, "Interaction understanding in the osi model functionality of networks with case studies," in *Applied Computational Intelligence and Informatics (SACI), 2014 IEEE 9th International Symposium on*, May 2014, pp. 327–330.

[46] Microsoft. (2014) The osi model's seven layers defined and functions explained. [Online]. Available: https://support.microsoft.com/en-us/kb/103884

[47] Y. Li, D. Li, W. Cui, and R. Zhang, "Research based on osi model," in *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, May 2011, pp. 554–557.

[48] J. Thomas. (2014) Seven layers of osi model and functions of seven layers of osi model? [Online]. Available: http://www.omnisecu.com/tcpip/osi-model.php

[49] M. Abolhasan, B. Hagelstein, and J. C. P. Wang, "Real-world performance of current proactive multi-hop mesh protocols," in *Communications, 2009. APCC 2009. 15th Asia-Pacific Conference on*, Oct 2009, pp. 44–47.

[50] "Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1–2793, March 2012.

[51] R. G. Garroppo, S. Giordano, and L. Tavanti, "Experimental evaluation of two open source solutions for wireless mesh routing at layer two," in *Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on*, May 2010, pp. 232–237.

[52] J. Chung, G. González, I. Armuelles, T. Robles, R. Alcarria, and A. Morales, "Characterizing the multimedia service capacity of wireless mesh networks for rural communities," in *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2012, pp. 628–635.

[53] S. Conner, J. Kruys, K. Kim, and J. Zuniga, "Overview of the amendment for wireless local area mesh networking," in *IEEE 802 Plenary*, Nov 2006.

[54] P. Murphy, A. Sabharwal, and B. Aazhang, "Design of warp: A wireless open-access research platform," in *Signal Processing Conference, 2006 14th European*, Sept 2006, pp. 1–5.

[55] K. Mandke, S.-H. Choi, G. Kim, R. Grant, R. Daniels, W. Kim, R. Heath, and S. Nettles, "Early results on hydra: A flexible mac/phy multihop testbed," in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, April 2007, pp. 1896–1900.

[56] (2011) Click modular router. [Online]. Available: http://read.cs.ucla.edu/click/click

[57] (2016) Polymorphic types. [Online]. Available: https://gnuradio.org/doc/doxygen/page_pmt.html

[58] G. Troxel, E. Blossom, S. Boswell, A. Caro, I. Castineyra, A. Colvin, T. Dreier, J. B. Evans, N. Goffee, K. Haigh, T. Hussain, V. Kawadia, D. Lapsley, C. Livadas, A. Medina, J. Mikkelson, G. J. Minden, R. Morris, C. Partridge, V. Raghunathan, R. Ramanathan, C. Santivanez, T. Schmid, D. Sumorok, M. Srivastava, R. S. Vincent, D. Wiggins, A. M. Wyglinski, and S. Zahedi, "Adaptive dynamic radio open-source intelligent team (adroit): Cognitively-controlled collaboration among sdr nodes," in *Networking Technologies for Software Defined Radio Networks, 2006. SDR '06.1st IEEE Workshop on*, Sept 2006, pp. 8–17.

[59] X. Li, W. Hu, H. Yousefi'zadeh, and A. Qureshi, "A case study of a mimo sdr implementation," in *Military Communications Conference, 2008. MILCOM 2008. IEEE*, Nov 2008, pp. 1–7.

[60] J. Seddar, H. Khalife, V. Conan, and J. Leguay, "A dtn stack for cognitive radio ad hoc networks," in *8th Karlsruhe Workshop on Software Radios*.

[61] M. Ott, I. Seskar, R. Siraccusa, and M. Singh, "Orbit testbed software architecture: supporting experiments as a service," in *First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*, Feb 2005, pp. 136–145.

[62] M. Sruthi, M. Abirami, A. Manikkoth, R. Gandhiraj, and K. Soman, "Low cost digital transceiver design for software defined radio using rtl-sdr,"

in *Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013 International Multi-Conference on*, March 2013, pp. 852–855.

[63] A. Sergienko, "Software-defined radio in matlab simulink with rtl-sdr hardware," in *Computer Technologies in Physical and Engineering Applications (ICCTPEA), 2014 International Conference on*, June 2014, pp. 160–161.

[64] B. Uengtrakul and D. Bunnjaweht, "A cost efficient software defined radio receiver for demonstrating concepts in communication and signal processing using python and rtl-sdr," in *Digital Information and Communication Technology and it's Applications (DICTAP), 2014 Fourth International Conference on*, May 2014, pp. 394–399.

[65] R. Danymol, T. Ajitha, and R. Gandhiraj, "Real-time communication system design using rtl-sdr and raspberry pi," in *Advanced Computing and Communication Systems (ICACCS), 2013 International Conference on*, Dec 2013, pp. 1–5.

[66] M. Abirami, V. Hariharan, M. Sruthi, R. Gandhiraj, and K. Soman, "Exploiting gnu radio and usrp: An economical test bed for real time communication systems," in *Computing, Communications and Networking Technologies (ICCCNT),2013 Fourth International Conference on*, July 2013, pp. 1–6.

[67] S. Nimmi, V. Saranya, G. Theertha Das, and R. Gandhiraj, "Real-time video streaming using gstreamer in gnu radio platform," in *Green Computing Communication and Electrical Engineering (ICGCCEE), 2014 International Conference on*, March 2014, pp. 1–6.

[68] (2016) Ettus research. [Online]. Available: https://www.ettus.com/

[69] E. Research. (2016) B200. [Online]. Available: https://www.ettus.com/product/details/UB200-KIT

[70] ——. (2016) E310. [Online]. Available: https://www.ettus.com/product/details/E310-KIT

[71] ——. (2016) E312. [Online]. Available: https://www.ettus.com/product/details/USRP-E312

[72] J. C. Briones and J. M. Nappier, "Sdr input power estimation algorithms," in *Aerospace Conference, 2013 IEEE*, March 2013, pp. 1–9.

[73] B. Wang and K. J. R. Liu, "Advances in cognitive radio networks: A survey," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 1, pp. 5–23, Feb 2011.

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

[75] S. Wang, D. Li, and S. Xia, "The problems and solutions of network update in sdn: A survey," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, April 2015, pp. 474–479.

[76] C. Trois, M. D. D. D. Fabro, L. C. E. de Bona, and M. Martinello, "A survey on sdn programming languages: Towards a taxonomy," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2016.

[77] G. Sun, G. Liu, and Y. Wang, "Sdn architecture for cognitive radio networks," in *Cognitive Cellular Systems (CCS), 2014 1st International Workshop on*, Sept 2014, pp. 1–5.

[78] M. Youssef, M. Ibrahim, M. Abdelatif, L. Chen, and A. V. Vasilakos, "Routing metrics of cognitive radio networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 92–109, First 2014.

[79] S. M. Cheng, P. Y. Chen, and K. C. Chen, "Ecology of cognitive radio ad hoc networks," *IEEE Communications Letters*, vol. 15, no. 7, pp. 764–766, July 2011.

[80] V. Kuppusamy and R. Mahapatra, "Primary user detection in ofdm based mimo cognitive radio," in *Cognitive Radio Oriented Wireless Networks and Communications, 2008. CrownCom 2008. 3rd International Conference on*, May 2008, pp. 1–5.

[81] M. T. Zia, F. F. Qureshi, and S. S. Shah, "Energy efficient cognitive radio mac protocols for adhoc network: A survey," in *Computer Modelling and Simulation (UKSim), 2013 UKSim 15th International Conference on*, April 2013, pp. 140–143.

[82] J. A. Msumba and H. J. Xu, "Spectrum sensing for cognitive radio networks: The need for cross-layer design approach for multimedia applications," in *AFRICON, 2011*, Sept 2011, pp. 1–6.

[83] K. Chowdhury, M. Di Felice, and I. Akyildiz, "Tp-crahn: a transport protocol for cognitive radio ad-hoc networks," in *INFOCOM 2009, IEEE*, April 2009, pp. 2482–2490.

[84] H. Cui, Y. Zhang, C. Ma, W. Lai, N. C. Beaulieu, S. Sobolevsky, and Y. Liu, "Design and realization of cognitive routing resources using big data analysis in sdn," in *2015 IEEE International Congress on Big Data*, June 2015, pp. 424–429.

[85] S. Namal, I. Ahmad, M. Jokinen, A. Gurtov, and M. Ylianttila, "Sdn core for mobility between cognitive radio and 802.11 networks," in *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, Sept 2014, pp. 272–281.

[86] I. Kassabalidis, M. A. El-Sharkawi, R. J. Marks, P. Arabshahi, and A. A. Gray, "Adaptive-sdr: adaptive swarm-based distributed routing," in *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, vol. 1, 2002, pp. 351–354.

[87] K. Kiran, P. D. Shenoy, K. R. Venugopal, and L. M. Patnaik, "Fault tolerant beehive routing in mobile ad-hoc multi-radio network," in *Region 10 Symposium, 2014 IEEE*, April 2014, pp. 116–120.

[88] V. K, K. K, P. D. Shenoy, and V. K. R, "Iobr: Interoperable bee-hive routing in a heterogeneous multi-radio network," in *TENCON 2015 - 2015 IEEE Region 10 Conference*, Nov 2015, pp. 1–5.

[89] R. A. Mahale and S. D. Chavan, "Throughput aware aco based routing protocol for wireless sensor network," in *Wireless Computing and Networking (GCWCN), 2014 IEEE Global Conference on*, Dec 2014, pp. 234–238.

[90] P. V. Kumar and S. Malarvizhi, "Experimental study of genetic algorithm based link adaptation for mimo cognitive radio application," in *Electronics and Communication Systems (ICECS), 2015 2nd International Conference on*, Feb 2015, pp. 1354–1359.

[91] B. Atakan and O. B. Akan, "Biologically-inspired spectrum sharing in cognitive radio networks," in *2007 IEEE Wireless Communications and Networking Conference*, March 2007, pp. 43–48.

[92] G. Li, S. W. Oh, K. C. Teh, and K. H. Li, "Enhanced biologically-inspired spectrum sharing for cognitive radio networks," in *Communication Systems (ICCS), 2010 IEEE International Conference on*, Nov 2010, pp. 767–771.

[93] F. R. Yu, M. Huang, and H. Tang, "Biologically inspired consensus-based spectrum sensing in mobile ad hoc networks with cognitive radios," *IEEE Network*, vol. 24, no. 3, pp. 26–30, May 2010.

[94] X. Mao and H. Ji, "Biologically-inspired distributed spectrum access for cognitive radio network," in *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, Sept 2010, pp. 1–4.

[95] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "Towards an open source ieee 802.11p stack: A full sdr-based transceiver in gnu radio," in *2013 IEEE Vehicular Networking Conference*, Dec 2013, pp. 143–149.

[96] (2015) Using openwrt as an openvpn server with a tap device (with bridging). [Online]. Available: https://wiki.openwrt.org/doc/howto/vpn.server.openvpn.tap

[97] (2016) Flask-socket.io. [Online]. Available: https://flask-socketio.readthedocs.org/en/latest/

[98] (2015) socketio-client. [Online]. Available: https://pypi.python.org/pypi/socketIO-client

[99] Tektronix. (2016) Rsa306 usb spectrum analyzer. [Online]. Available: http://www.tek.com/spectrum-analyzer/rsa306

[100] O. Mesh. (2016) Understand your b.a.t.m.a.n. advanced network. [Online]. Available: https://www.open-mesh.org/projects/batman-adv/wiki/Understand-your-batman-adv-network

Appendix A

Running ARCAM-Net

This appendix will serve as a "How-to" manual for getting started with ARCAM-Net.

A.1 Configuring the Computer

First, Ubuntu 14.04 LTS will need to be installed on each node computer. Installing Ubuntu is beyond the scope of this document. However, there are plenty of easy to follow tutorials available online.

Once installed, the remaining steps can be done manually, or you can use a shell script I created to automate most of the process. This shell script is available on Github. As the name of the repository suggests, you can also use this to setup and configure a virtual machine to run the SDR software using Vagrant and Virtual Box. If you are unable to install an OS on the computers you are using, this is a good alternative. Please be aware that both options involve a lot of downloading and compiling of software. This process can take up to 8 hours so be sure to have plenty of time to let this run. An internet connection is required.

A.1.1 Installing Natively

```
git clone https://github.com/jmccormack200/GnuRadio-Vagrant-Script.git
cd GnuRadio-Vagrant-Script sudo sh bootstrap.sh
```

A.1.2 Installing Through Vagrant

Download and install Vagrant and Virtualbox to your computer. Then run:

```
vagrant up
```

The rest should be configured automatically.

A.1.3 Manual Install

If you prefer to do things yourself just be sure to follow along with the "boot-strap.sh" file below:

LISTING A.1: Installer Shell Script

```
#!/usr/bin/env bash

apt-get update

#Everything below here is mentioned as being a prereq for
#GNURadio except Vim, Vim is just swell. Love Vim.
apt-get --yes --force-yes install python
apt-get --yes --force-yes install vim
apt-get --yes --force-yes install cmake
apt-get --yes --force-yes install python-pip
apt-get --yes --force-yes install libboost-all-dev
apt-get --yes --force-yes install libcppunit-dev
apt-get --yes --force-yes install git
apt-get --yes --force-yes install libfftw3-dev libfftw3-
    doc
apt-get --yes --force-yes install libgsl0ldbl
apt-get --yes --force-yes install swig
apt-get --yes --force-yes install libsdl2-2.0
apt-get --yes --force-yes install python-wxgtk2.8
apt-get --yes --force-yes install build-essential
apt-get --yes --force-yes install mesa-common-dev
apt-get --yes --force-yes install qt-sdk
apt-get --yes --force-yes install libqwt-dev
apt-get --yes --force-yes install python-numpy python-
    scipy python-matplotlib ipython ipython-notebook
    python-pandas python-sympy python-nose
apt-get --yes --force-yes install python-setuptools
apt-get --yes --force-yes install python-gtk2-dev
apt-get --yes --force-yes install doxygen
apt-get --yes --force-yes install doxygen-gui
apt-get --yes --force-yes install graphviz
apt-get --yes --force-yes install git-core
apt-get --yes --force-yes install g++
apt-get --yes --force-yes install python-dev
apt-get --yes --force-yes install pkg-config
apt-get --yes --force-yes install python-cheetah
apt-get --yes --force-yes install python-lxml
apt-get --yes --force-yes install libxi-dev
apt-get --yes --force-yes install python-sip
apt-get --yes --force-yes install libqt4-opengl-dev
apt-get --yes --force-yes install libfontconfig1-dev
```

```
apt−get −−yes −−force−yes install libxrender−dev
apt−get −−yes −−force−yes install python−sip
apt−get −−yes −−force−yes install python−sip−dev
apt−get −−yes −−force−yes install libboost−all−dev
apt−get −−yes −−force−yes install libusb−1.0−0−dev
apt−get −−yes −−force−yes install python−mako
apt−get −−yes −−force−yes install python−docutils
apt−get −−yes −−force−yes install xinitx
apt−get −−yes −−force−yes install batctl
apt−get −−yes −−force−yes install gnome
apt−get −−yes −−force−yes install xinit


sudo pip install pyside

#These were to allow for qt to work, but seem to be
    errors
export PYTHONPATH=$PYTHONPATH:/opt/qt/lib/python2.7/dist−
    package
export PYTHONPATH=$PYTHONPATH:/usr/bin/python


#remove standard packages of UHD if they are there
sudo apt−get −−yes −−force−yes remove libuhd−dev
sudo apt−get −−yes −−force−yes remove libuhd003
sudo apt−get −−yes −−force−yes remove uhd−host

#Install UHD Using Ettus Repos
sudo bash −c 'echo "deb␣http://files.ettus.com/binaries/
    uhd/repo/uhd/ubuntu/'lsb_release␣−cs'␣'lsb_release␣−cs
    '␣main" > /etc/apt/sources.list.d/ettus.list'
sudo apt−get update
sudo apt−get −−yes −−force−yes install −t 'lsb_release −
    cs' uhd


git clone −−recursive http://git.gnuradio.org/git/
    gnuradio.git

#Here we build gnuradio from sources
#This will be the longest process
cd gnuradio
mkdir build
cd build
cmake ..
sudo make
sudo make install


sudo ldconfig
```

```
##Now we will download gr-mac
cd ..
cd ..
git clone https://github.com/balint256/gr-mac.git
cd gr-mac
mkdir build
cd build
cmake ..
sudo make
sudo make install

sudo ldconfig

##Which also needs gr-foo
cd ..
cd ..
git clone https://github.com/bastibl/gr-foo.git
cd gr-foo
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig


#RTL Driver
cd ..
cd ..
git clone git://git.osmocom.org/rtl-sdr.git
cd rtl-sdr
mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON
sudo make
sudo make install
sudo ldconfig

#RTL-GNU Radio Pluging
cd ..
cd ..
git clone git://git.osmocom.org/gr-osmosdr
cd gr-osmosdr/
mkdir build
cd build/
```

```
cmake ../
make
sudo make install
sudo ldconfig


cd ..
cd ..
git clone https://github.com/btrowbridge/alfred-ubuntu.
   git
cd alfred-ubuntu
sudo bash ubuntuAlfredInstall.sh
```

A.2   Getting The Flowgraphs

The plubic version of ARCAM-Net can be found on Github. The public version
will be the most stable release of the network, and will be updated each time
the private repository is used to publish a new paper. To get the flowgraphs
use:

```
   git clone https://github.com/jmccormack200/ARCAM-Net-Public.git
cd ARCAM-Net-Public cd Flowgraphs
```

A.3   Running The Flowgraphs

At the time of this writing, there are four files in the Flowgraph folder of the
repository. The four files are actually two sets of files. The files ending in ".grc"
are the GNU Radio Companion implementations and the files ending in ".py"
are the same files but written in python. The Python files are automatically gen-
erated from the GRC blocks. The files "batNoGui.grc" and "batmanNoGui.py"
represent the files used by ARCAM-Net which do not contain a GUI. This is
needed to run the flowgraph in a background thread while using the Flask
web server. If you just plan on running the flowgraphs I recommend using
"broadcastwithFreqNoMac.grc" or "broadcastwithFreqNoMac.py" as this will
also include a helpful GUI for interacting with the flowgraph.

These flowgraphs are used by ARCAM-Net but are not the entire network.
They are just the scripts used to control the SDR.

To run a GNU Radio script open it using:

```
   sudo gnuradio-companion Name_of_Script
```

Then at the top of the flowgraph click "build" and then "Run". Alternatively, they can be run straight from the command line using:

```
sudo python Name_of_Script
```

Please note that admin privileges are needed in order to create the TUN/-TAP device. If running the python file from the command line, it is possible to use command line arguments to adjust certain parameters prior to running. We found this to be useful as we could start the network a specific gain instead of having to go through and manually change the gains at each node.

A.3.1 Flow Graph to Mesh Network

Running just the flowgraph will not start the system with batman-adv running. The user must manually assign an IP address to the TUN/TAP by running:

```
sudo ifconfig tun0 192.168.200.XXX
```

Where the X's can be replaced with an IP Address of your choice. Make sure to keep the IP's unique on each node. Then, batman-adv must be started on each node. To do this, a helpful script called "raiseBatSignal.sh" can be run to automatically configure batman-adv. This is described below.

A.3.2 Raising the Bat Signal

"raiseBatSignal.sh" can be found in:

```
ARCAM-Net-Public/WebInterface/static/shell/
```

It can be run by typing:

```
sudo sh raiseBatSignal.sh
```

Alternatively, you can run the commands individually from the command line. This shell script ensures the batman-adv module is loaded into the kernel then configures tun0 to be the interface used by batman. If this does not match your setup, adjust tun0 to be whatever interface the SDR is named. The contents of the shell script are below:

LISTING A.2: Shell Script to Start Batman-adv

```
#!/bin/sh
```

```
sudo modprobe batman−adv
sudo batctl if add tun0
sudo ip link set mtu 1532 dev tun0
#sudo ip link set down tun0
sudo ip link set up tun0
#sudo ip link set down bat0
sudo ip link set up bat0
```

A.3.3 TMux Script

TMux is a terminal multiplexer that makes it easy to work with numerous ssh instances across multiple machines. A script is provided that sets up the entire network by running just a single command. TMux needs a bit of adjustment to work with different setups. Luckily, the only changes that need to be made are to open the file and edit the IP addresses, usernames, and passwords of the systems you want to control. This script assumes each node will have the same username and password. If network policies restrict you from doing this, then the script will need heavier modification. This script cant be found in:

```
ARCAMNetPublic/Tools/tmux_start.sh
```

The contents of the file are shown below. To run the command use:

```
sudo bash tmux_start.sh
```

Please note that this script must be run with bash in order to work.

LISTING A.3: Bash Script to start the entire network

```
#!/bin/bash

# Requires sshpasss
# sudo apt−get install sshpass

SESSION=$USER

# Function to connect to all sessins
# format should be:
#               connect user pass ip

# To turn on network type:
#       sudo bash tmux_start.sh

# To Stop type:
#       tmux ls
```

```
#  Then :
#           sudo  tmux  k i l l −s e s s i o n  −t  <name  from  above
    command>

connect ( ) {

          tmux  new−window
          tmux  rename−window  $3
          tmux  s p l i t −window  −v

          tmux  select −pane  −t  0

          ssh="sshpass ␣−p␣$2␣ssh␣−X␣$1@192 . 1 6 8 . 1 . $3"
          tmux  send−keys  "$ssh"  C−m

          tmux  send−keys  "cd␣SDR/SDR/ Flowgraphs"  C−m
          tmux  send−keys  "echo␣$2␣|␣sudo␣−S␣python␣
              broadcastwithFreqNoMac . py␣−−tx−gain␣45␣−−rx−
              gain␣45"  C−m

          tmux  select −pane  −t  1
          tmux  send−keys  "$ssh"  C−m
          tmux  send−keys  "sleep␣10s"  C−m
          tmux  send−keys  "echo␣$2␣|␣sudo␣−S␣sh␣~/SDR/SDR/
              WebInterface/ s t a t i c / s h e l l / raiseBatSignal . sh"  C
              −m
          tmux  send−keys  "echo␣$2␣|␣sudo␣−S␣ i f c o n f i g ␣bat0␣
              192.168.200.$3"  C−m

          tmux  send−keys  "sleep␣10s"  C−m

          tmux  send−keys  "echo␣$2␣|␣sudo␣−S␣ a l f r e d ␣−i␣bat0␣
              −m␣&"  C−m

          tmux  send−keys  "echo␣$2␣|␣sudo␣−S␣ b a t c t l ␣o␣−w"  C−
              m

}

user=′ v t c l a b ′
pass=′ v t c l a b ′


#  IP  L i s t
#  192.168.1.100
#  192.168.1.101
#  192.168.1.102
```

```
#  192.168.1.106
#  192.168.1.107
#  192.168.1.109

#IP = (100, 101, 102, 106, 107, 109)

IP[0]="100"
IP[1]="101"
IP[2]="103"
IP[3]="104"
IP[4]="105"
IP[5]="106"

# Start Session
tmux -2 new-session -d -s $SESSION

for ip in "${IP[@]}"
do
        connect $user $pass $ip
done


# Connect to setup session
tmux -2 attach-session -t $SESSION
```