Software Defined Radio Mesh Network Testbed Implemented in GNU Radio with Batman-adv

John McCormack, Bradley Trowbridge, Joseph Prine, R. Cody Maden, Ryan Integlia

***Abstract*—Software Defined Radio Networks (SDRNs) use systems of Software Defined Radios (SDRs) to establish networks with flexible physical and link layers. GNU Radio is an open source software tool set for working with SDRs and can be used as a basis for creating SDRNs. Mesh networks are designed to allow for flexible and distributed network architectures that are self forming and function without the need for centralized infrastructure. Batman-adv is a popular, open source, layer 2 mesh network protocol. Our work establishes an SDRN platform by combining GNU Radio with Batman-adv to create a fully open source software defined radio mesh network. The platform can work with any USRP SDR device to quickly prototype and experiment with SDR and Cognitive Radio (CR) Protocols. Due to the flexibility of Batman-adv and GNU Radio, programs acting above Layer 2 can utilize this network without any changes. In order to further increase the cognitive abilities of the platform, we leverage the A.L.F.R.E.D. tool chain within the batman-adv ecosystem to distribute information about frequency changes across the mesh network. This creates a novel method to globally change the frequency of the network in a completely decentralized way.**

***Index Terms*—Software Defined Radio, ad-hoc network, mesh network, Software Defined Radio Network, SDR, SDRN, Batman-adv, USRP, GNU Radio**

I. Introduction

Software defined radios (SDRs) have been around for many years. However, as the cost of SDRs continues to drop, the technology becomes much more accessible. Additionally, open source tools such as GNU Radio make developing for SDRs much easier. GNU Radio provides a feature rich ecosystem that provides a wealth of signal processing blocks. Though hardware is not a direct component in GNU Radio, numerous other developers and projects have integrated hardware functionality into the system either natively or through additional out-of-tree modules.

GNU Radio Companion allows for GUI development of PHY and MAC layer protocols within the GNU Radio environment.The project itself is implemented in a combination of Python and C++ modules. Ettus research, a division of National Instruments, created the Universal Software Radio Peripheral (USRP) which is the SDR we chose for the project and a popular choice among many projects. Ettus also released the Universal Hardware Drivers (UHD) which allow for the use of the USRP with GNU Radio.

Cognitive Radio Networks (CRNs) are networks made up of SDRs that are capable of making intelligent decisions on their own and adjusting parameters such as signal strength. Many Cognitive Radio scenarios are designed around the idea of ad-hoc or mesh networks. In these networks, all of the associated radio components are able to talk to each other either directly or by "hopping" from one node to another until they reach their destination. The Better Approach to Mobile Ad-hoc Networks (BATMAN) project created the Batman-adv protocol. This layer 2 protocol has a fairly large community and is integrated into the Linux Kernel.

Our goal with this project was to create a low cost, open source platform, that could serve as a basis for future projects in Cognitive Radio Network research. Our hope is that by combining the GNU Radio and Batman-adv projects, more collaborative research can be done with a standarized toolset. Our platform will also allow for students who are looking to create application layer products that leverage cognitive radio meshes to get started without having to worry about all the intracasies of these networks.

II. Related Work

A. 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 [6]. This makes the platform very expensive and limits its adaptability for use in other research groups.

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 [7]. 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 [**?**]. 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 [**?**].

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. [10] Similarly, the University of California, Irvine and Boeing Corporation developed a testbed based off of USRP Radios and GNU Radio, but they implement custom MAC layers [9].

The platform that most closely resembles ours is presented in [8]. However, this platform uses 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 [**?**].

B. GNU Radio and 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 [12] and [13] the authors use GNU Radio as a way to verify the succesful use of algorithms for mesh networking. However, they do not use SDRs,instead choosing to use GNU Radio for simulation.

The researchers in [2] created a simple multihop test bed using three USRP radios to relay data from one computer to another. A forth 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 can 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 OSI protocol stack from the ground up. In some papers the authors focus on the physical or mac layer [3]. There has also been work in developing new higher layer protocols for cognitive radio mesh networks such as work done to replace TCP with a more robust protocol [4]. These systems will usually react to frequency changes but some also change their topology based on power use [5]. All of this research is extremely important, does not create a complete platform to serve as the backbone for future work.

C. Mesh Network Testbeds

The CONFINE platform uses Batman-adv as the routing protocol for their mesh network testbed. However,this testbed does not utilize GNU Radio or SDRS. [18] Batman-adv was also a key component of WiBed, a project to create a COTS mesh test bed using low cost wirelessrouters [19] [20]. Though they are not specific to SDR platforms, these still show the usefulness of Batman-adv as a component of a testbed.

III. Design

The Design of the test bed can be broken down into the following parts:

- USRP Software Defined Radio
- GNU Radio Flowgraph
- Batman-adv
- Flask Web Server
- SocketIO Web Sockets
- A.L.F.R.E.D.

A. USRP Software Defined Radio

For our project, 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. Their relatively low cost makes them ideal for building out larger testbeds. These serve as the radio transcievers 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.

B. GNU Radio Flowgraph

GNU Radio utilizes programs called “Flowgraphs” to allow for graphical programming of 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. This flowgraph is a very simplistic, but effective, implementation of a GMSK or OFDM transciever 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 heirarchical block is built by running a separate flowgraph which contains the 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 continuously data transmitting in that portion of the flowgraph. However, packets are not sent continuously so separate logic is needed to convert streams to messages. These messages are passed into and out of the GMSK and OFDM heirarchical blocks, so the remainder of the flowgraph deals with passing messages only.

The “simple mac” block is written in C++ and handles decoding the data. This block implements an Automatic Repeat Request (ARQ) protocol in addition to encoding packets being sent to the radio, and decoding packets coming back from the radio. It assigns also assigns a local address for the SDR. The remainder of the flowgraph establishes a TCP server and either a tunnel (TUN) or network tap (TAP) interface. 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. Either of these could be selected to suit the users purpose, but as batman-adv is a layer 2 protocol, we will use the TAP protocol. This flowgraph also implements an 802.3 Tracker to build out a radio to address map of the network.

This flowgraph was largely used in the form provided from its Github repository. The main changes we needed to make were small, but important to making the whole system function. First, we removed all GUI Components so that we could run the environment in a background thread. Next, we altered the “destination address” parameter to be broadcast instead of looking for only one specific destination. Finally, we made most of the important variables into parameters and selected “thread safe setters and getters” so that we could access them from another program.

C. Batman-adv

Batman-adv was chosen based on its large community and documented success as a mesh routing protocol. It is already included as part of the Linux Kernel, and additional software can be downloaded from most distributions repositories. Configuring batman 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. With just 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 in an effort to make cognitive radio testing much simplier.

D. Flask Web Server and Socket.IO

Flask is a lightweight, open source, web framework for the Python programming language. 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. The only input needed from the user is an IP address for the TAP port, but this could later be replaced by running a DHCP server on the mesh network.

Socket.IO is a javascript library that enables real-time bidirectional event-based communication. 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 easily into Flask and can be used in stock python with a client library. 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 verious settings about the radio, and allows for the user to change parameters. 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. Flask recieves the request, and then uses A.L.F.R.E.D. to manage the next step.

E. 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. A.L.F.R.E.D. is very simple to use, but also exceptionally powerful. Whenver a node writes data to a channel on A.L.F.R.E.D., that data is passed from node to node to all other members of the network. Typical uses for A.L.F.R.E.D. include keeping track of sensor data to allow for a visual map of an environment to be made.

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. 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. 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 recieved 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 recieved. 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.

IV. Results

Now we talk about the data collection

V. Limitations and Future Work

We then talk about the assumptions and such

VI. Conclusions

Nearly done

VII. Acknowledgments

We would like to thank the academy

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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. –.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] (2015) Confine project. [Online]. Available: http://confine-project.eu/

[14] 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.

[15] 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.