ECE1896

Wireless Networking using Software Defined Radio

DESIGN SPECIFICATIONS

CODY HOUCK AND NATHAN HACHTEN

Background and Project Definition

Software defined radio (SDR) is a communication technology where the analog front-end hardware that is used in many conventional systems is replaced with high-speed digital logic that is fully configurable by the designer through software. This project is an attempt to harness the versatility of SDR in the form of a packet data transmission system similar to the modern day Wi-Fi protocol using only the software platform provided by GNU Radio and Universal Software Radio Peripherals (USRP).

Software defined radio started its development back in the late 80's as a way for United States Air Force to communicate across a wide range of frequencies without the need for specialized equipment. The program SPEAKEasy was the final product of this research, which allowed ground units to quickly change RF bands and also build receivers for various signal formats from the ground up in less than two weeks' time[8]. This provided a substantial advantage as several regions of the world operate in different bands than the United States - due to the lack of regulatory agencies such as the FCC – and could have used foreign modulation schemes that were not currently implemented. This project worked well for small units but overall failed to deliver the real-time capabilities needed in military situation, thus it was scrapped until about a decade later when the DoD commissioned a team with the sole purpose of getting this technology off the ground[8].

The Joint Tactical Radio System (JTRS) is considered to be the first reliable realization of software defined radio as it provided all of the capability originally promised by SPEAKEasy while remaining true to its versatile role where all front-end components are completely software controlled[11]. This program eventually changed its face and gathered a strong backing in the military realm, where it soon became the standard for several branches. JTRS is kept alive by the development of software radios by both the government and commercial providers who wish to take advantage of the configurability and connectivity provided by the platform[8].

Commercial providers involved in supporting the world of wireless devices ideally provide continuous, reliable service that is always available for customers to use[10]. Taking a base station down for maintenance could potentially leave an area dark (no access to a cellular network) which presents a safety hazard and also financial issues to the user. Although this is an unlikely event, it happens. The hardware eventually wears out or is deprecated, unable to keep up with new standards and devices dumping larger volumes of data into the system. Using software defined radio prevents these outages as updating hardware involves only re-flashing an FPGA or instantiating a new version of the software platform, neither of which take more than five minutes if properly tested. This also prevents wasteful replacement of equipment, making this technology a sustainable alternative to the modern wireless network architecture as the devices can be updated via software instead of being scrapped entirely.

The problem in current local area networks (LAN) is very similar to the problem that was encountered by the military back in the late 80's as different standards operate using different carrier frequencies and modulation schemes, not to mention all of the security layers that need to be handled. The ability to adapt to different types of signals without swapping out hardware is the biggest push for the 2.4 [GHz] band as it supports several different standards (Wi-Fi, Bluetooth, etc) which all require their own specialized hardware and algorithms. By using a software defined radio, these incompatible standards can be linked through software which can implement the algorithmic processing and control the specialized hardware without the need of a standard communication process[13]. The software demodulates the signal, turns it into a machine language (binary), and hands it off to the next layer of the system without caring about the signal characteristics.

The problem with this technology is that most of the development is performed inside of larger companies with the hopes that they will be the first to offer its services to their customers. These projects tend to rely on specialized processing units known as FPGA's which require extensive training and usually incur long development time[13]. They fit the definition of software radio, but they lack the full API that opens their development to beginning users or developers that have a classical programming background. The overarching goal of modern software radio supporters is to have an easily-accessible template that conventional software developers can interface with in order to incorporate these systems into modern architectures[13]. These solutions exist in the form of open-source platforms such as GNU Radio and RedHawk, which aim to use general purpose processors as digital signal processing units and offload data to FPGA's only when necessary. This idea represents a convergence of radio frequency engineering and software development, which could eventually give programmers the power to perform intricate signal processing without the need for intensive training.

Previous research has been performed in the area of open-source, general purpose signal processing architectures by the previously mentioned SDR platforms GNU Radio and RedHawk. GNU Radio is a full development suite of signal processing algorithms that can be used by a software developer to create complex modulation schemes, packet architectures, and various other structures common in networks of today[13]. RedHawk is the same type of platform, with the exception that it also directly supports the fact that each algorithm can use its own method of processing the data. RedHawk supports general purpose processing along with the ability to easily offload data to an FPGA when it reaches a certain point in the transmission or reception, making it a more viable solution to the real-time requirements of a standard communication system. GNU Radio lacks in this arena, but provides a more intuitive interface to common algorithms, making it more 'developer friendly' than the RedHawk API.

Sustainability is an important part of any modern design as it is critical to all facets of the process. The design must be reliable, thus capable to work for an extended amount of time without replacement, pollution, or disrupting the public's daily life. It must also be economically optimal as the bottom line of a project is crucial to getting the required parts while under the constraints of a budget. Software defined radio provides a sustainable alternative to the modern methods of digital communication as it can be upgraded without a complete overhaul of equipment, thus reducing the environmental impact of deprecated electronics. It also reduces the overall cost as one central unit that provides all the front-end processing services can be periodically upgraded to support newer technology, removing the need to replace the device after every change. Connecting this radio to the current infrastructure allows the software to be upgraded remotely, reducing the number of field engineers required to maintain the network. And lastly it mitigates the chance of the public losing communication abilities. Software has the unique ability to diagnose its condition and the condition of its components, therefore degradation in network capabilities can be detected before an incident occurs or immediately after. This prevents outages that can cause a communication blackout, therefore providing a layer of safety to the system as well.

The description of software defined radio suggests that it is the ideal candidate to take over modern communication front-ends, so why hasn't it? This problem arises in its current stage of development; it is a far from trivial process to integrate all parts of a previously analog system into a digital framework that supports current network protocols. This project will show how a system can be realized using an open-source software platform along with publicly available software radio technology and how this type of system can be dynamically reconfigured to fit a certain need.

Preliminary Work and Design

This project aims to provide a basic networking system that can be used to pass digital data from one user to another using a simplified version of Point-to-Point connections. The data will be read into the system through an application, passed to the networking layers, encoded, and sent over the air through the physical layer. The OSI Model of a network protocol is a standard way to view a communication system as it provides the various layers that are needed to successfully pass data from a user down to a transmission system and back up to another user[4][5]. Figure 1 shows the different layers of the model that will be referenced in future designs.

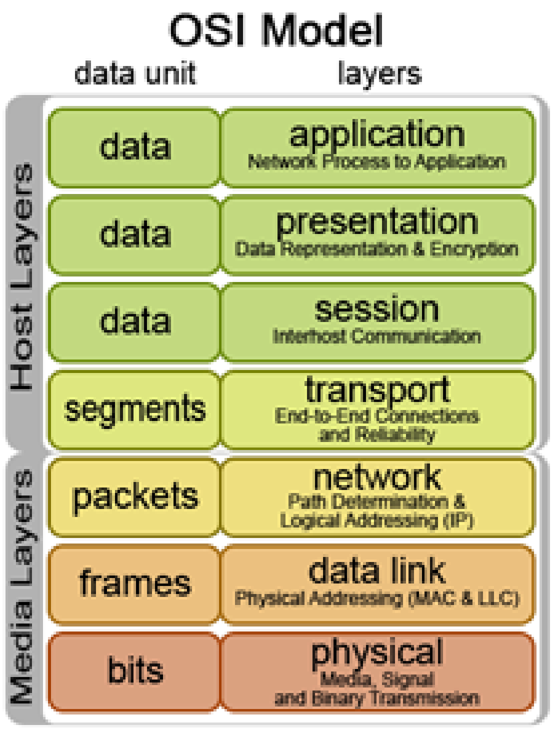
OSI Model
data unit
layers
Host Layers
Media Layers
data
application
Network Process to Application
data
presentation
Data Representation & Encryption
data
session
Interhost Communication
segments
transport
End-to-End Connections and Reliability
packets
network
Path Determination & Logical Addressing (IP)
frames
data link
Physical Addressing (MAC & LLC)
bits
physical
Media, Signal and Binary Transmission

Figure 1 **–** OSI Model of transmission protocol[9]

Each layer presents its own challenges, therefore separating them where possible is a necessary step in the successful development of any standard. The essential functions of each layer can be are defined by this model and are what the project will follow in order to accomplish end-to-end transmission of data. Table 1 outlines the different layers and their function, along with the design options available.

Layer	***Function***	***Input Type***	***Output Type***	***Options***
Application	Generates the data that will be sent to user	Data	Data	GUI Interface, Web Browser, etc.
Presentation	Formats data from application	Data	Data	-
Session	Connects above layers to corresponding port	Data	Chunks of Data	-
Transport	Connection and reliability manager	Chunks of Data	Packets	Transmission Control Protocol (TCP) or User Datagram Protocol (UDP)
Network	Addressing and path determination	Packets	Frames	Internet Protocol (IP) or TUN
Data Link	Physical addressing (MAC)	Frames	Bits	Address Resolution Protocol (ARP), TAP, or custom access code method
Physical	Data transmission of bits	Bits	RF	OFDM, QAM, Forward-Error Correction, Encoding, Bit Rate, Bandwidth, etc

Table 1 **–** Description of each layer and the available options

The design criteria are directly related to the breakdown of each layer as certain methods work better with others, such as the TCP/IP connection and TUN/TAP[4]. The criteria for this project are thus naturally divided into two distinct sections; networking and RF transmission/reception. The networking layers provide the user the ability of inputting data into the system in whatever formats are supported, setting up a connection with another user, and packetizing this data for transmission by the physical layer. The physical layer takes the incoming bytes, encodes and modulates them, and then transmits them over the air to the other radio front-end, where the front-end then performs the inverse of these operations and passes the data back to the networking layers[6]. To identify the design criteria of the entire system it is useful to break the process into these two areas and define them separately.

Network Criteria

There are three main criteria to consider when evaluating the feasibility of design alternatives. They are elegance (or ease of use), time/skill required, and performance. Elegance is a measure of how easily do existing applications make use of the network, if at all. Can instant messaging applications, email clients, and web browsers perform normally with no special knowledge of the underlying network architecture or configuration? If not, what steps need to be taken to get these applications working? Or will the network only work with a specialized software communication program that we design?

The time and skill required to complete a design is important because the goal is to be basically finished by March 6, and to be completely finished by about April 7. This gives only a few weeks of design, development, and testing. There may be a design alternative that provides an elegant and high-performance solution, but if it would take a team of five engineers six months to complete, it would be unreasonable.

The final design criteria to be considered is performance. This one is challenging to apply to designs before testing because no quantifiable results are available. There will be two non-numerical facets to the performance criteria that will make it useful for evaluating potential designs. The first is does the design ensure reliable delivery of data. The answer to this is yes or no. The second facet of performance is a risk factor. What is the risk that this approach involves a great deal of overhead that could slow the system down?

These three design criteria will be applied to three design alternatives. These designs focus only on the networking side of things, and each design could be used with any of the proposed physical layer RF design alternatives. These designs aim to provide networking functionality to a user to be able to send information from a computer connected to radio A to a computer connected to radio B.

Design alternative one is to use Linux virtual network kernel devices called TUN and TAP, referred to as TUN/TAP when speaking about them together. TUN, for network tunnel, simulates a layer 3 Network layer device and operates with IP packets, providing routing. TAP, for network tap, simulates a layer 2 Data Link layer device and operates with ethernet frames, providing a network bridge. [15]

TUN/TAP is an easy solution for sending data from user space, for example a web browser, to GNU Radio, which is also in user space, but giving the applications the appearance that they are sending/receiving data from Network layer and Data Link layer devices. Data would come from a server in the form of IP packets and enter into the TUN/TAP device. The TUN/TAP device would then unload an ethernet frame into GNU Radio. GNU Radio would then take care of the RF transmission and the Physical layer.[15]

The TUN/TAP design is an elegant solution because it would allow for the creation of a network connection that, from the view of user programs, looks like an ethernet connection. The time and skill required is reasonable to be finished within the time frame of the course, but there is a risk that interfacing it with GNU Radio could be problematic. Performance would be satisfactory because there is no extra overhead, and protocols like TCP, which ensure reliable data delivery, are available.[15]

Design alternative two is two use TCP/IP to a port on localhost that GNU Radio has access to. Unlike using TUN/TAP, this design does not perform TCP/IP over the entire connection. Once the data gets to GNU Radio it is the responsibility of the radio to ensure that all data is transmitted.[4]

This solution lacks elegance because it requires an extra piece of software to interface communications between users and GNU Radio. For existing user applications to make use of the network, they must be able to pass data to the GNU Radio software interface.

The time and skill required for this solution varies depending on how much functionality is desired. In order to get a very simple communication path setup, the time and skill required is minimal. To get existing user applications to treat use the network without special knowledge would require more time and expertise, but it is likely still within the scope of this course's timeframe.

The performance of this design is poor. It does not ensure any kind of reliable delivery, because TCP/IP halts once the data reaches GNU Radio. Also, there is likely overhead when the data is packaged to be sent, and this would slow down the data transmission.

The third design option is to build a custom suite of protocols that provide the desired functionality. This would involve designing a layer 2 Link layer protocol that provides MAC addressing services. The point-to-point protocol could provide a good example for the custom protocol.[4]

This design alternative has the potential to be very elegant. Since it is built from scratch, it can be made to work especially well with GNU Radio. Existing applications could make use of the network with no special knowledge.

The time and skill required for this solution are very high. Since this involves designing and implementing a protocol from scratch, it would likely take longer than the few weeks before the end of the course.

Performance for the solution depends on the skill of the designer. It could potentially allow for protocols like TCP to ensure reliable delivery and to send data with little overhead.

In evaluating the design alternatives, each criteria is worth between 0 and 5 points. More points means a better design. For time/skill, a 5 means that it is achievable but challenging. A 0 means that it is either too easy, or too difficult.

Design	***Elegance***	***Time/Skill***	***Performance***	***Total***
TUN/TAP	5	4	5	14
TCP/IP to localhost	3	2	3	8
Custom	5	0	5	10

The best design, according to the scoring metric, is to use TUN/TAP.

Physical Layer Criteria

The physical layer has very little to do with the networking of the system as it blindly takes data and sends it over the air via a radio front-end to another radio front-end; hence the only requirement from this layer is that the above layers pass bytes that are already formatted for transmission. For the radio front-end, a software defined radio central unit will be required in order to interface with the RF hardware that is needed to transmit the data. This includes the actual software radio along with the corresponding antennas needed for a particular band in the spectrum. Due to the nature of the project, the radio will have to provide a programmer friendly interface that can be programmed easily, thus the low level components (such as the FPGA or proprietary circuits) will need to be abstracted away for efficient design. The selected SDR platform for the design is GNU Radio as it is intuitive and provides the ability to use the algorithms on a general purpose processor. The Ettus Research USRP software radios were then selected based on this decision as their products fully support GNU Radios API. There are several radio front-ends that can provide this functionality, therefore constraints need to be placed on the design. These constraints are defined below in the list, which is a basic breakdown of the performance metrics the physical layer needs to satisfy. The constraints were chosen such that the system will provide a fast, reliable connection to the corresponding system while remaining inside certain boundaries defined by the Federal Communications Commission (FCC) and practical use by a customer.

- 1 [W] output power / 4 [W] effective isotropic radiated power (EIRP)[14]
- < 20 [MHz] bandwidth to imitate Wi-Fi
- Center frequency in the Wi-Fi band (2.4 - 2.48 [GHz])
- 100 [Kbps] for practical use as a communication system (can browse internet)
- 0% packet loss for perfect reconstruction of original data

Although several other constraints can be placed on any given communication system, these provide a practical and legal basis for further design. The first requirement is defined by the FCC for unlicensed low-power devices and is a legal constraint on the system. This limits the output power of the system and thus directly limits the range of the system and indirectly limits the throughput of the channel[2]. The second is a constraint that will be used out of courtesy as the Wi-Fi bands are a maximum of 20 [MHz] in bandwidth; going over this bandwidth could be considered a conflict and adversely affect users on adjacent channels. The third constraint is a practical concern, as a system that operates below this level would be unusable as a 1 [MB] file would take over 80 [sec] to transmit, which is considered ‘slow’ by comparison to modern systems. The final constraint is that all of the data that is sent by the first user is received by the second user and none is lost during the transmission.

The first constraint can be satisfied by any radio that has a variable gain; if the system exhibits a power level too high for the constraint then it can be reduced in order to compensate[2]. The second constraint is also trivially satisfied, as most off-the-shelf software radios have an effective bandwidth that is less than 20 [MHz], therefore leaking into adjacent bands would involve over-exerting the radio. The final two constraints are where the design presents itself. Speed in a digital communication system is commonly quantified by the bit rate R, which is the number of bits that are transmitted per second[2]. Using only transmission of binary bits through a poorly conditioned channel will result in a large bit error rate (BER) and ultimately cause many packet errors to occur, meaning that this design criteria directly affects the packet loss constraint as well[2]. Therefore the design process involves choosing a modulation scheme that is robust against poor channel conditions such as those in a room with other devices operating in the 2.4 [GHz] band while also increasing the data rate.

Several types of modulation are available in digital communication systems that would satisfy this criterion, thus the design choices were limited to those that are used in modern day wireless networks. The two main design choices for a system that must overcome poor channel conditions are code division multiplexing (which was used in cellular devices starting in 1995) and orthogonal frequency division multiplexing. Code division multiplexing (CDM) uses orthogonal bases in the time domain XOR'ed with a pseudorandom carrier code to modulation the respective input bit stream[3]. The alternative method is orthogonal frequency division multiplexing which also utilizes orthogonality, but in the frequency domain. Each bit (or number of bits) is mapped to an symbol (quadrature amplitude modulated symbol) and a particular subcarrier using the discrete Fourier transform (DFT); the resulting subcarrier signal is then added to the full data symbol[3][7].

CDM offers a unique way to layer data and allows for multiple users to share the same channel, but it is inherently difficult to implement. Generating a pseudorandom binary signal at a rate that is high enough to modulate the baseband bit stream is a daunting task for a general purpose processor as it has a distinct limit set by the clock speed. Also, generating mutually orthogonal signals in time for multiple streams can be computationally complex as it involves using the modified Gram-Schmidt method to dynamically construct the bases. OFDM also has its flaws, as it requires perfect frequency synchronization between the transmitter and the receiver which is a variable set by the software to hardware tuning mechanism[12]. Any resulting error between these will cause the data to rotate its phase at the speed defined by the offset, requiring additional computing power to track this rotation.

The other constraint that is considered is the resistance to errors during the transmission process. If the data bits are mapped to the subcarriers directly a spurious, deep fade of one of the subcarriers will directly result in a loss of data and the symbol will have to be repeated entirely. Forward-error-correction (FEC) is a method of encoding the original data in a way that makes each chunk of bits unique. This is ideal in a system that must send packetized data as losing one information bit will result in a complete retransmission of the frame, therefore adversely affecting the speed of the system. There are several methods of FEC that could be used but for the sake of implementability only two have been considered; block codes and convolution codes. A block code adds parity bits to a chunk of data that act as a detect-and-correct mechanism, utilizing a unique characteristic to tell when an error has occurred and correcting that error when possible[2][7]. A convolution code uses a sliding boolean polynomial to encode the data by adding parity bits in a way such that only one bit code generates a certain parity scheme, much like that of the block codes[1][7]. Convolution coding and block coding are similar in realization complexity but have different implementation problems. Convolutional coding is simple to perform on the transmission side but requires a 'Trellis decoder' to recover the data, which is a maximum likelihood decoder used on every bit in the sequence[7]. Although this is a simple task to perform on each individual bit, the system has a higher computational complexity than that of a block code and requires much more detailed decoding algorithms. Block codes on the other hand are a simple matter of appending parity bits to a chunk of bits and checking them at the received side, a process that involves simple Boolean operations on each chunk. Although this method is simple, it comes at the cost of performing worse than a convolution code as its information bits are very closely to each other[2].

The two main designs that have been selected for development were based off of an analysis of the resources available. There are only two groups members designing the software with a fixed budget and the development time is approximately three months, thus it is crucial to get a design that both meets certain criteria and can be accomplished in the allotted time. By outlining the above design alternatives and stating the resources, the final criteria for the physical layer design can be chosen accordingly. The two crucial criteria for the design of the physical are the robustness of the system and the time and skill required for its completion within the time frame.

The robustness criteria stems from a direct need to ensure the data that is handed off to the transmitter by the first user is properly received by the second user. Failure to satisfy this criterion results in the system not allowing communication between two nodes, which is the ultimate purpose of a digital communication system. By reducing the bit error rate the robustness of the system is improved, therefore this will be considered the top priority in the design of the physical layer. The secondary priority is the time and skill level needed to complete the project. Time is the limiting factor of this project as a fully functioning system complete with testing and statistics is required in a little over three months, but skill level is what will ultimately decide how elegant the solution is. A robust system can be created even if some of the design considerations are swapped due to complexity, it is a matter of creatively implementing the software in a way that harnesses the full capability of the radio.

By using the same method as above, the physical layer metric is defined below.

Design	**Robustness**	**Time/Skill Level**	**Total**
OFDM	5	3	8
CDM	3	4	7
Convolution Code	5	2	7
Block Code	3	4	7

By analyzing the above metric, it can be concluded that OFDM is the ideal choice for the modulation scheme. The forward error correction is the choice of the developer as they both exhibit the same metric, thus it is a matter of choice.

Selected Design Approach

The design approach selected for the physical layer is to use OFDM with 16-level quadrature amplitude modulation symbols (16QAM) along with ½ rate block coding, meaning that for every *n* information bits *n* parity bits are added. This design approach can be implemented to achieve high bit rates and is extremely robust when presented with multipath interference in the signals bandwidth. The final design of the system is included in the list below:

- Bandwidth < 20 [MHz]
- OFDM with 16QAM complex baseband symbols
- ½ rate block coding
- Bit rate *R* > 100 [Kbps]

OFDM is the ideal choice due to its simplicity to design and its performance in the presence of interference. The most complicated algorithm of the front-end system is the DFT which can be performed quickly and efficiently using the Fast Fourier Transform (FFT). Although CDM would work for this project, its performance in fading channels is poor due to the data being superimposed; if a deep fade occurs in part of the effective bandwidth of CDM signal all of the data is lost. OFDM mitigates this effect by spacing out the data across the band in individual subcarriers[7]. If a deep fade occurs in one of the subcarriers it can be compensated for by recording the error and sending the missing data with the next frame or by placing the data around this poor subchannel. The modulation scheme is the biggest component of the design and the forward error correction will merely boost its performance. The selected approach for the forward error correction is to use a ½ rate block code that pads the data with an equal number of redundant parity bits. Although this method will not perform as well as a convolution code, it will provide a gain over simply sending the mapped information bits through the channel. The FEC is an assurance that the BER is lower than sending uncoded data through a channel[7].

The performance will be evaluated in pieces with respect to the entire network. To properly quantify the speed and BER of the physical layer, a large series of random bits will be generated and saved to a file. This file will then be read by the physical layer directly and sent over the air interface where the received signal will attempt to recover the data. The received data bits will then be compared to those of the large binary file to recover the BER. The time for this entire process to occur will be collected to estimate the speed of the physical layer, where the number of received bits will be divided by total number of time to obtain the average bit rate *R* for the system. The robustness of the system will be measured directly by comparing the BER to a theoretical value and the speed will be compared in a similar fashion. The physical layer metrics will be gathered in different environments and the results will be recorded as part of the testing to show how the BER differs for various testing facilities.

The selected design approach for the networking side is TUN/TAP. TUN/TAP is the best solution because it provides easy-to-use and high performance network capabilities that are independent of the physical medium in use.[15]

TUN/TAP provides packet reception and transmission for user space programs, while appearing as a point-to-point or ethernet device. TUN and TAP are both linux kernel device drivers implemented in software. TUN/TAP sends and receives packets to and from user space programs. This functionality will enable user programs, like chat applications or web browsers, to send/receive data to/from TUN/TAP. This gives the user space programs the appearance of sending data to IP addressable physical devices. In reality, that data will enter user space again when it reaches GNU Radio. GNU Radio then takes care of the Physical layer services.[15]

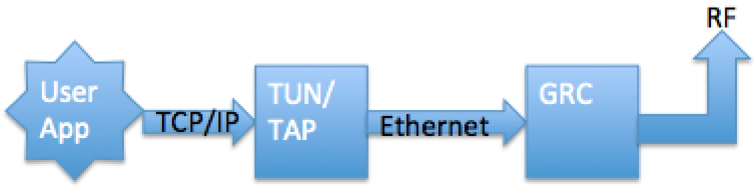

Figure 2 - Example of how TUN/TAP will work.

TUN/TAP will allow for an IP addressable network. A virtual network interface in Linux will be created to enable network connectivity. The MAC addresses of the USRPs will need to be translated to IP addresses for routing purposes. This will be accomplished with a simplified ARP table that is initially hard coded.

Evaluation and testing will be done with the Netcat tool. Netcat allows for reading from network connections. It can be used to view the data that is sent and received at the point between user applications and TUN/TAP. [17]

The tests to be performed are the same as mentioned above, but the observations will differ. Netcat will be used to view packet information as it is sent and received. Any missing packets will be noted.[17]

<u>Minimum Standard</u>

The final product of the software defined radio network will be designed to meet a minimum standard that satisfies all original design criteria. The minimum standard for this project can be summarized by three statements:

- The ability to set up network communication that is IP addressable
- The ability to transmit data with 0% information loss within the devices range
- The ability to send data through the network at speeds that support practical use using software radios

The ability to set up a network that is IP addressable is a crucial part of the design. If the system cannot identify certain users than it cannot be called a network, thus this is the foundation for the entire system. The importance of the IP addressability comes from the fact that if the system uses an IP address it can communicate with other devices looking for that particular address. This gives the system the ability to interface with various applications and send data regardless of its format.

The ability to transmit data with 0% information loss within the devices range is also very important. If the system transmits data but information is lost during this transmission the system has failed as the secondary user will not receive the intended message. The definition of a digital communication system is to send data to another user with no information loss, therefore it is suitable to make this part of the designs minimum standard.

The ability to send data through the network at speeds that support use using software radios is the final part of the minimum standard. This system is a software defined radio network that uses an open-source library for its signal processing, therefore it presents the challenge of obtaining real-time speeds with a general purpose processor. If the system is unable to send and receive data at speeds that are practical for basic use then this system has failed.

The requirements that our advisor will need in order to assign a passing grade are:

- A data sheet containing all of the technical information from the final product, including test data and verification that the standards outline above were met
- Visual confirmation that the system works as intended. The advisor will be able to use our product to send and receive data through the wireless network.

One of the primary goals for this project was the ability to send data through the air using the software radios and the GNU Radio SDR platform. This was accomplished two weeks ago by setting up the physical layer to send out three sinusoids of unit amplitude at 100 [kHz], 200 [kHz], and 300 [kHz] and filter these signals on the receiver side using a bandpass filter to show proof that data can in fact be sent by the front-end. The primary goal to have accomplished in the next few weeks is to start with a basic skeleton of the TUN/TAP architecture and implement the ability to pass data to the physical layer for transmission. Once this is accomplished the receiver will be implemented such that it recovers the data sent by the transmitter and passes it up to the TUN/TAP networking layers.

<u>Non-Technical Constraints</u>

The five areas of non-technical constraints are economic constraints, environmental constraints, manufacturing constraints, ethical constraints, and social constraints. Our project is limited by economic and manufacturing constraints.

The economic constraints that could limit this project is the cost of one of the USRP software radios. They are several thousand dollars each, and if one of them were to fail we would have no way of obtaining a new one.

The manufacturing constraints that limit this project are due to the difficulty in reproducing our work for a new system. If we wanted to package up our final solution and ship it, we would need to flash the software onto a group of software defined radios. This would be both expensive, because of the radios, and difficult, because only a few types of radios would work.

There are no environmental constraints in this project, because RF has no effect on the environment. There are no ethical constraints because this project does not violate the IEEE Code of Ethics. There are no social constraints because this project does not impose on any social liberties.[16]

Preliminary Results

Nathan has accomplished getting GNU Radio to communicate with the Linux operating system through a local port connection. He has also determined that using TUN/TAP to satisfy Link Layer protocol services is a better solution than writing a custom protocol. Cody has accomplished getting the software radios setup and testing them to ensure they do in fact transmit and receive the appropriate signal waveforms. He has also finished the software implementation of the USRP interface to the receiver and transmitter, allowing them to be dynamically changed using a configuration file. Cody has also tested the basic abilities of the software radios to send and receive unformatted OFDM packets and is currently testing the system to obtain upper bound on the bit rate.

The problems that have been encountered are in the front-end performance of the system. During the testing of transmission and reception of basic signals it was noted that the frequencies being received were slightly shifted, introducing carrier frequency offset (CFO). This is a problem as OFDM is sensitive to CFO; this can be overcome by using a feedback loop that automatically pushes this offset back to zero.

Project Revisions

There have been a few revisions since the project proposal. One design requirement changed. The minimum bit rate was changed from 1 Mbps to 100 Kbps. The solution has changed slightly. The proposal indicated that we would be designing some sort of a custom network protocol. We have decided to use Linux's built in TUN/TAP software device drivers for networking support.

The milestones and schedule will remain the same, except for the change noted above. The budget and supplies are the same.

References

[1] Bourdoux, A. and F. Horlin. *Digital Compensation for Analog Front-Ends*. England: John Wiley & Sons Ltd., 2008. Print.

[2] Couch, L. *Digital and Analog Communication Systems: 8th Edition*. New Jersey: Pearson Education Inc., 2013. Print.

[3] Hanzo, L., Wong, C., and Yee, M. *Adaptive Wireless Transceivers*. England: John Wiley & Sons Ltd., 2002. Print.

[4] Kurose, J., and K. Ross. *Computer Networking: A Top-down Approach Featuring the Internet: 3rd Edition*. New York: Pearson Education Inc., 2005. Print.

[5] Rackley, S. *Wireless Networking Technology*. Burlington: Newnes, 2007. Print.

[6] Tonguz, O., and G. Ferrari. *Ad Hoc Wireless Networks*. England: John Wiley & Sons Ltd., 2006. Print.

[7] Van Nee, R. and R. Prasad. *OFDM for Wireless Multimedia Communications*. Boston: Artech House Publishers, 2000. Print.

[8] Del Re, E. *Software Radio: Technologies and Services*. London: Springer-Verlag, 2001. Print.

[9] Manske, M. *OSI-Model*. Web: Wikipedia, 2006. Digital Image. Viewed on Jan 25, 2015. <http://commons.wikimedia.org/wiki/File:Osi-model.png#mediaviewer/File:Osi-model.png>

[10] Hannan, M. A., Islam, M., Samad, S. A., and Hussain, A. *QAM in Software Defined Radio for Vehicle Safety Application*. Web: Australian Journal of Basic and Applied Sciences, 2010. Viewed on Jan 25, 2015.

[11] Prasad, P. and Koch, R. *The Universal Handset*. Web: IEEE Spectrum, 2009. Web Page. Viewed on Jan 25, 2015. <http://spectrum.ieee.org/computing/embedded-systems/the-universal-handset>

[12] Yeh, H. G. and Ingerson, P. *Software-Defined Radio for OFDM Transceivers*. Web: Proc. of Systems Conference, 2010 4th Annual IEEE, San Diego, CA.

[13] "Welcome to GNU Radio!" Web:_WikiStart GNURadio. Web Page. Viewed on Jan 25, 2015. <http://gnuradio.org/redmine/projects/gnuradio/wiki>

[14] FCC. *Understanding the FCC Regulations for Low-Power, Non-Licensed Transmitters*. Maryland: FCC, 1993. Print.

[15] Krasnyansky, Maxim, and Florian Thiel. "Universal TUN/TAP Device Driver." *Kernel.org*. Linux Kernel Organization, Inc., 2002. Web. 25 Feb. 2015. <https://www.kernel.org/doc/Documentation/networking/tuntap.txt>.

[16] "IEEE IEEE Code of Ethics." *IEEE*. IEEE, 2015. Web. 17 Feb. 2015. <http://www.ieee.org/about/corporate/governance/p7-8.html>.

[17] "Ncat - Netcat for the 21st Century." *Nmap.org*. Nmap, n.d. Web. 25 Feb. 2015. <http://nmap.org/ncat/>.