Open-Source Prototyping of Advanced Wireless Systems for Smart Agriculture and Connected Rural Communities

Final Report

Team Number: sdmay20-37
Client: ISU

Advisers:
Hongwei Zhang
Matthias Sander-Frigau

Team Members/Roles:
Dylan Sharp - Meeting Facilitator
Zequn Wang - Meeting Scribe
Yulin Song - Test Engineer
Jiawei Deng - Chief Engineer
Shaohang Hu - Test Engineer
Zhengwei Su - Report Manager
Team Email: sdmay20-37@iastate.edu
Team Website: https://sdmay20-37.sd.ece.iastate.edu/

Revised: April 25, 2020

Executive Summary

Development Standards & Practices Used

The following development standards and practices applied to this project:

- Agile Development
- IEEE 802.11. This is the standard for port-based Network Access Control.
    - IEEE 802.11af - Amendment that focuses on WLAN operation in TVWS (TV white space) spectrum
- pktRT: Per packet real-time guarantees
- PRKS: Predictable link reliability

Summary of Requirements

- Implementation of PRKS Algorithm
    - Application of the PRKS Algorithm will help reduce the link transmission latency and decrease package loss rate.
- Implementation of pktRT Algorithm
    - The application of the pktRT Algorithm will help to minimize the latency and make the package transmission in a real-time manner.
- Implement PRKS and pktRT within TV white space spectrum
    - TVWS spectrum is a platform we used to send and receive packages.
- Proof of concept implementation within the smart agriculture industry

Applicable Courses from Iowa State University Curriculum

- CPRE 308 - Operating system, principles, and practice
- COM S 311 - Introduction to the design and analysis of algorithms
- CPRE 430/530 - Network protocols and security
- CPRE 489 - Computer networking and data communications
- CPRE 548X - Cyber-Physical Systems Networking

New Skills/Knowledge acquired that was not taught in courses

- TVWS (TV-White Space)
- PRKS algorithm
- pktRT algorithm
- ONAMA algorithm
- Wireless kernel development
- Wireless networking protocols

Table of Contents

List of Figures

List of Tables

List of Definitions

ONAMA (Optimal Node Activation Multiple Access): Scheduling protocol for wireless networks that ensures maximal concurrency and low latency.

OpenWrt: Open source embedded Linux operating system focused on the routing and management of network traffic.

PRKS (Physical Ratio K Scheduling): Wireless network scheduling algorithm that ensures predictable link reliability.

pktRT: Probabilistic Per-Packet real-time guarantee algorithm for wireless networks.

TVWS (TV White Space): Refers to unused frequencies between TV network channels which can span from 470 MHz to 790 MHz.

1 Introduction

1.1 Acknowledgement

We would like to express our appreciation to Prof. Hongwei Zhang and Dr. Matthias Sander-Frigau, for their patient guidance, enthusiastic encouragement, and useful critiques of this senior design project. Dr. Sander-Frigau has been a critical key mentor throughout this project by meeting with us weekly and providing advice and direction.

1.2 Problem and Project Statement

Rural regions are home to many industries such as agriculture, renewable energy, and manufacturing, and they are major sources of food and energy for our society. For instance, the US agriculture and food sectors alone contribute more than $750 billion to the national GDP every year. However, rural regions are subject to growing challenges such as population shrinkage and labor shortage, the need to feed a growing population and food demand while subject to climate changes, and the rural-urban education gap. One basic enabler for addressing these challenges and for ensuring the prosperity of rural regions and communities and our society, in general, is broadband rural connectivity. Rural broadband is the digital superhighway for the broad rural communities and industries, but 39% of the rural US lacks broadband access, and most farms are not connected at all. Through this project, we are going to develop and prototype advanced wireless solutions for smart agriculture and connected rural communities. We are going to use cutting-edge TV white space (TVWS) wireless platforms and advanced wireless algorithms. At the end of this project, we hope to accomplish a spiral evaluation and refinement of the aforementioned novel 5G wireless solution for smart agriculture and connected rural communities.

1.3 Operational Environment

Our implementation will be within the OpenWrt operating system environment. OpenWrt is a flavor of embedded Linux focused on the routing and management of network traffic. This operating system will be running on top of the Qualcomm Atheros QCA9533 Rev. 2 physical chipset. The spectrum that this implementation will operate within will be that of the TVWS spectrum. Using this spectrum will allow for wireless applications to be developed in rural areas where channels in the TVWS spectrum tend to be more available.

1.4 Requirements

1.4.1 Functional Requirements

- The protocol must store a buffer of nodes
- Each node in the exclusive region must update every 1 s
- Every node must monitor the signal strength
- Every node must store their local signal map
- Value of K must update for each exclusive region every 1 s

1.4.2 Non-Functional Requirements

- Package loss rate should below 10%
- Link reliability should be higher than 90%
- Real-time latency should be below 50 ms
- Network throughput should be higher than 10 Mbs

1.5 Intended Users and Uses

Our direct users will be engineers and developers who work in the industry of wireless communications. This open-source prototyped solution can serve as a plan to solve connectivity problems in rural regions by utilizing the TVWS spectrum.

The product will benefit customers/users in the smart agriculture industry and rural communities. In the US, tons of industries are based in rural areas, as well as agriculture. A big number of them, especially farms, have a lack of broadband connectivity or have no reception at all. So if these regions can have a reliable wireless network setup, there will be big progress towards smart industry/agriculture.

1.6 Assumptions and Limitations

1.6.1 Assumptions

The following assumptions have been made thus far:

- The end developer is sufficient in embedded Linux development to use our libraries/code.
- The end developer is using our code to develop an application within the smart agriculture industry.

1.6.2 Limitations

The following are limitations that have been defined thus far:

- Local unused channels within the TVWS Spectrum to operate on.
- Hardware limitations (RAM, CPU Speed, etc.).

- Any noise/interference within the TVWS Spectrum.

1.7 Expected End Product and Deliverables

Final product deliverables will be split up into the following three categories:

- Preliminary algorithm implementations on OpenWrt (Mid-February)
    - The implementation of the algorithms shall be completed and delivered in mid February.
    - Will be delivered in a way such that developers can use it as a library for their applications on OpenWrt to communicate over the TVWS spectrum.
- Proof of concept application(Late-March)
    - Proof of concept will be completed and delivered in late March.
    - Will use our libraries delivered above within application.
- Final Deliverables (April)
    - Finalized implementation of algorithms will be delivered.
    - Final Report / Demonstration will be delivered to finish up the project.

2. Specifications and Analysis

2.1 Proposed Design

Our current approach to this project has been focused mainly on knowledge capture and requirement clarification. The algorithms required to be implemented are complex and challenging. Due to the complexity we have been focusing on understanding the fundamentals for the algorithms and what each algorithm needs in order to function properly. In addition to the algorithm understanding we have also started setting up our development environment by setting up a virtualized embedded operating system running on emulated hardware so that we can develop on our local machines without the need of the physical hardware.

In the process of accomplishing those tasks we split up into two groups, one focused on learning algorithms and the other focused on setting up the virtualized OpenWrt and emulated hardware.

2.2 Design Analysis

The algorithm group mainly focused on the PRKS algorithm and has created a list of parameters needed for the algorithm. In addition, they focused on the pktRT algorithm second and have also created a list of parameters needed for the algorithm. Due to the complexity of the algorithms we have a general understanding of the algorithms but still

do not have a full understanding of them. At the time of writing this design document, this group is in the process of implementing the algorithms into Linux kernel.

The OpenWrt group installed VirtualBox and set up OpenWrt within a virtualized environment. This group also set up QEMU to emulate the hardware under neight OpenWrt. This group finished the initial set up of those two tasks and then merged with the other group to help assist in learning the algorithms. Part of that help included mapping the parameters defined from the algorithms to locations within OpenWrt where the values could be found and assistance with the pseudocode generation.

The process of learning the algorithms has been pretty challenging due to the complexity, that is why we had the OpenWrt group merge with the other group to help assist. This has slowed us down a little bit due to the fact that it is harder to split up tasks since we are all focusing and helping each other on the same thing.

2.3 Development Process

For this project, we are following the Agile approach during the development processes. This will help with any variable components of our project and also help the team stay up to date with what everyone is doing. In following the Agile process, we hold weekly sprints and weekly retrospectives on the week. This helps us talk about any concerns as a team and be able to have the flexibility to bring in new tasks each week.

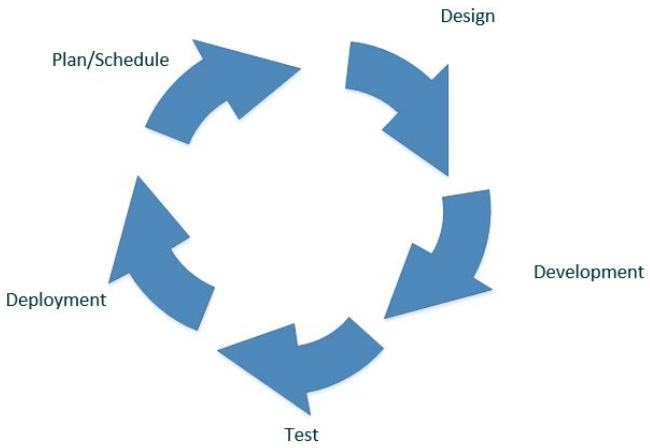

Fig 2.3 . The Agile Model

In the development process:

- We developed a Gantt chart which listed detailed deadlines for each of our objectives to different stages. For instance, we set up deadlines for OpenWrt virtualization, Algorithms understanding, Prototype development.

- For each single task, we created a corresponding card on our trello and synchronize the task to trello every week. So, team members who take responsibility to their jobs can publish some links or descriptions on the card.

2.4 Design Plan

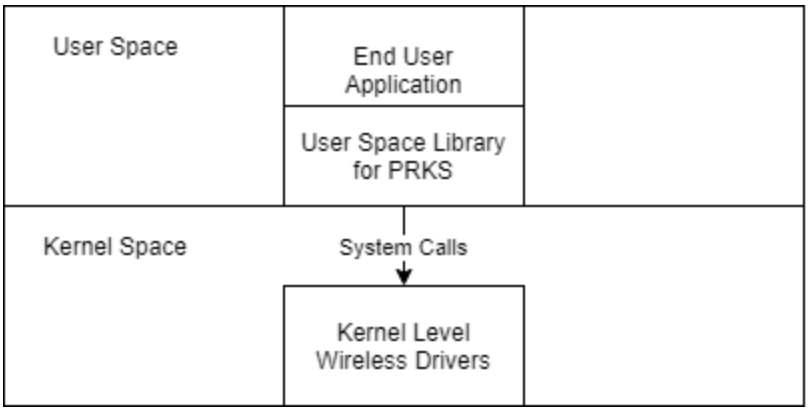

Fig. 2.4.1 System Design Plan

Our current design plan at a high level is to modify kernel wireless drivers that are controlled by system calls from a user space library. That will be the first deliverable and then using that the proof of concept application will be built on top of that using the user space library.

To break that down further we will implement the wireless network system in modules or chunks. Below is the architectural diagram of the PRKS algorithm below, which was grabbed from the algorithms research paper (See reference [2]).

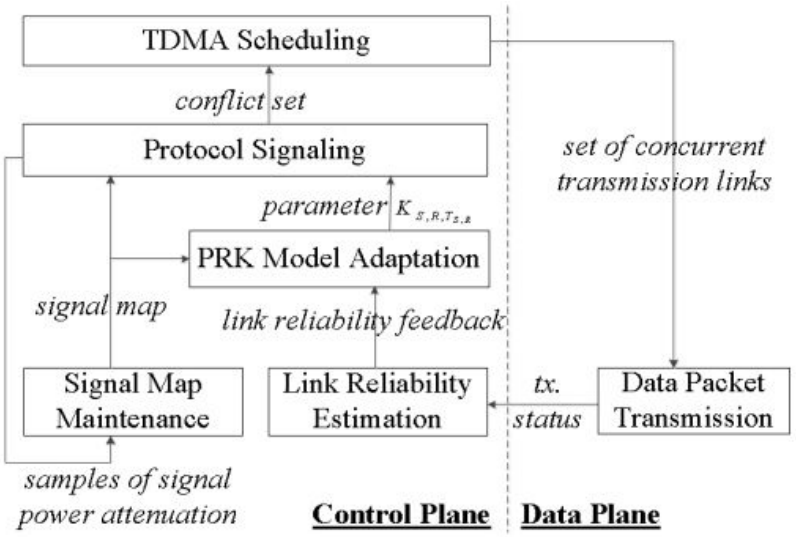

Fig. 2.4.2 PRKS Architecture Diagram.

This diagram helps display the different modules within the PRKS algorithm and what relational arguments are passed from one module to another. All of these modules listed above will be implemented individually and tested individually in a series like fashion. Note instead of using TDMA for scheduling we will be implementing the ONAMA scheduling algorithm.

The majority of our implementation will be within the linux kernel where we will need to modify the linux kernel WiFi drivers. Below is a diagram that represents the linux kernel IEEE 802.11 WiFi implementation:

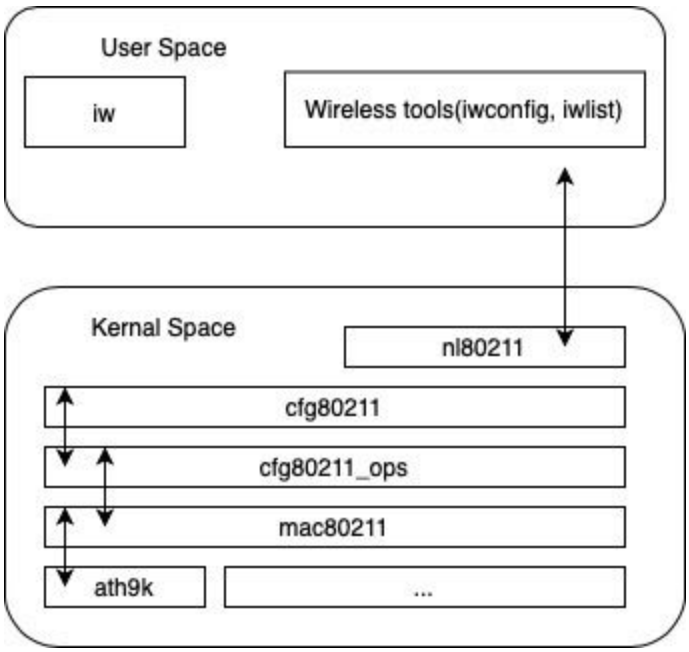

Fig. 2.4.3 IEEE 802.11 Linux kernel implementation architecture.

For our implementation we will be needing to make roughly the following edits / additions to this current linux implementation:

- Modification to the ath9k device driver to allow operation of IEEE 802.11af (an amendment to IEEE 802.11 for the use of the TVWS spectrum. This includes enabling the frequency bands needed for the TVWS spectrum.
- Within the mac80211 subsystem we will need to add a PRKS mode to support using the PRKS algorithm.
- Within the cfg80211 we will need to add a PRKS configuration mode to support our mac80211 PRKS mode.
- Add commands to nl80211 to add the support of joining and leaving the PRKS network on the TVWS spectrum.
- Edit iw (user space application) to support the new PRKS modes.

3. Statement of Work

3.1 Previous Work And Literature

Not very much previous work has been done to our knowledge in implementing 5G advanced algorithms within the TVWS, specifically PRKS and pktRT. These algorithms have been implemented before in order to test their feasibility and some of the documentation / results from those tests are included in the algorithm research papers. (See references [1] and [2]). These implementations have been strictly feasibility tests in a lab, and those tests concluded that the implementation would be feasible and effective, so now our team is conducting the field implementation.

3.2 Technology Considerations

This project's technologies considerations have mainly been taken into account by the client and have been defined as part of the requirements.. Given that, below is listed the technology options chosen:

- OpenWrt - Operating System
  - Makes sense to start implementation on this OS because of the fact that it is focused on network traffic.
- PRKS, pktRT, and ONAMA
  - These algorithms were part of the requirements to implement to help solve the rural broadband problem.
- Qualcomm Atheros QCA9533 Rev. 2
  - This hardware platform was also part of the requirements in the project, as that is what our advisors have on hand.

3.3 Task Decomposition

Main higher level tasks for this project are defined below:

- Algorithm understanding
  - pktRT
  - PRKS
  - ONAMA
- Algorithm Parameter mapping for each of the algorithm
  - Define each algorithm's parameters
  - Map each parameter to OpenWrt
- Algorithm pseudo code

    - Create pseudo code for each algorithm
- Virtualizing Development
    - Emulate Hardware
        - Set up QEMU to Emulate MIPS 24K devices with Atheros QCA9533 chip
    - Virtualize OS
        - Set up OpenWrt with QEMU above to setup for development
- Implementation
    - Edit kernel drivers to support the algorithms
    - Edit user level library to interact with kernel drivers for the algorithms
- Proof of concept
    - Define proof of concept application
    - Write application
    - Test application (potentially on physical devices)
- Testing
    - Non-functional tests
    - Functional tests

3.4 Possible Risks And Risk Management

One of the biggest risks to this project is the team's technical understanding. Pretty much every part of this project is new to us, no one on the team has experience with wireless networking, kernel development, or knew the algorithms previously. Given this as a risk we have spent a large chunk of time in the research and understanding of these topics, in particular the understanding of the algorithms. To do that effectively we started by dividing up the learning tasks of the team and then getting back together and having each other teach what they had learned to help mitigate the risk and burden on each individual facing the same struggles of learning the algorithms.

The second biggest risk for this project and this team is our availability, our team members this fall semester where in enrolled in a lot of credits which limited the number of hours put into the project. To minimize this risk we plan sprints every week so that the team meets up typically twice to discuss any concerns or technical challenges. Doing this helps bring up potential problems to the entire team of which we can help each other out and if someone was too busy to take on specific tasks, the team can help delegate those tasks to other members.

3.5 Project Proposed Milestones and Evaluation Criteria

3.5.1 Milestones

The following major milestones have been set up for this project:

- OpenWrt Kernel Development Learning
- Algorithm Learning
- Design Phase
- Linux Implementation
- Proof of Concept Implementation

3.5.2 Evaluation Criteria

To evaluate the team has accomplished the milestones defined the team will do the following:

- For Learning and Design milestones the team will meet up and share each other's understanding on the topic at hand. When everyone has the same understanding and agreement we will present to our advisors our shared understanding to confirm if in fact we understand the topics correctly. If our understanding of the topics is correct, then we have accomplished the milestone.
- Evaluating whether or not our implementation milestones have been met will be done by performing functional and non-function testing to verify that we in fact meet our requirements defined in section 1.4. More information regarding these tests is outlined in section 5.

3.6 Project Tracking Procedures

We use Trello to track our projects progress through each sprint in our agile development process. Our team typically does one week sprints and relies on Trello for documenting any progress throughout the sprint / project. This allows for all team members to currently see where each other are currently working and any documentation related to the tasks they are working on. Currently we create tasks based on higher level tasks and tie them together in order to track the higher level task progress. That process is usually done as a team and put into our backlog where we pull tasks from to plan current sprints.

Team communication is handled in a group iMessage chat, this is typically used for scheduling meet ups or asking questions. If any questions pertain to a particular task then details from that conversation are carried over to trello for tracking purposes.

Files are stored within a shared Cybox drive and include notes taken for tasks, meeting notes with advisors, bi-weekly / weekly reports, and any other documents pertaining to this project.

Code is tracked via GitLab and tied to tasks in trello. We practice branching off master and creating pull request views for merging completed work back into master. This allows for the team to peer review each other's code and point out any possible flaws.

3.7 Deliverables

The outcome of this project is to have an implementation strategy of a new wireless network system that guarantees high throughput, low latency communications and high packet reliability for rural areas and smart agriculture.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

This project is split over two semesters. The fall semester we primarily are focusing on knowledge absorption and design while the spring semester we will focus more on implementation and integration.

Task Name	Start Date	Finish Date	Estimated Hours
Semester 1			
OpenWrt Kernel Development Learning			
Virtualize OpenWrt	9/16/19	9/23/19	8
Cross Compile Hello World	9/16/19	9/30/19	10
Hardware Virtualization w/ QEMU	9/23/19	9/30/19	16
Linux Kernel Wifi System Understanding	10/21/19	10/28/19	25
Hello World Kernel Module	10/1/19	10/28/19	20
Algorithm Understanding (Learning)			
Define Algorithm Parameters	9/1/19	10/27/19	110
Map Algorithm Parameters to	10/28/19	11/18/19	16

OpenWrt			
Design			
Design Document	11/17/19	12/8/19	30
Proof of Concept Design Defined	1/13/20	1/20/20	8
Semester 2			
Documentations for Implementation Strategies	2/3/20	5/1/20	85
Router Hands On	4/6/20	4/24/20	30

Table 4.1 Task Timeline

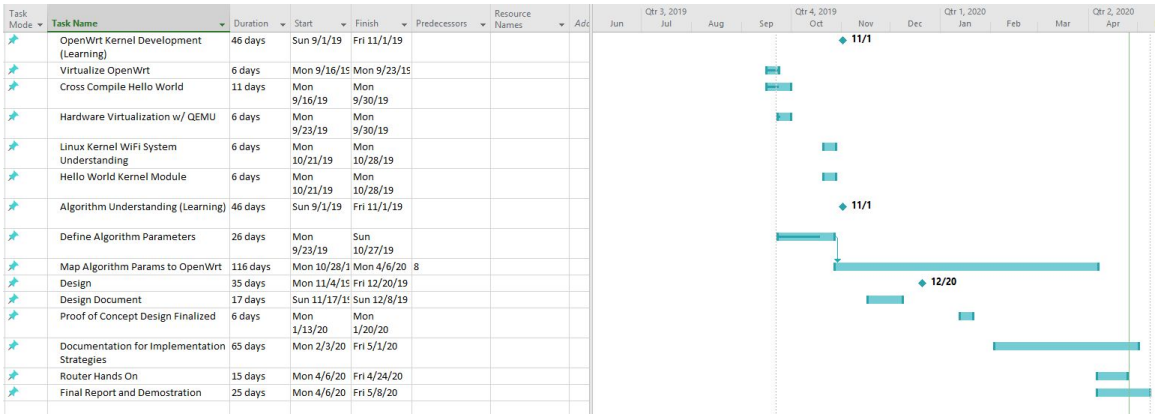

Fig. 4.1 Gantt Chart

4.2 Feasibility Assessment

We believe that the timeline outlined above is feasible as a team. We attempted to anticipe some lag in the schedule in case tasks do not go as planned.

The biggest challenge so far has been just understanding of the challenging algorithms and communication with the team. The understanding of the algorithms has already begun to show some lag but our hopes are that over the break we will continue our learning and gain a full understanding. Communication with the team has also proven to be slacking we have noticed and are continuing to improve upon that aspect to minimize risk to the project.

4.3 Personnel Effort Requirements

Task estimations can be found in table 4.1 in the *Estimated Hours Needed* column. The estimations were made as a team and are subject to change as the team understands its velocity better.

4.4 Other Resource Requirements

The following additional resources are required:

- Legal Access to TVWS Spectrum
- Physical Chipset and Antenna

Access to the TVWS is required and has been fulfilled as our advisors have gotten us legal access to the spectrum around ISU. Our advisors are also providing us with the Qualcomm Atheros QCA9533 rev 2 ChipSet for the application.

4.5 Financial Requirements

Most of our tasks will be understanding the algorithms referenced and physical device driver, creating code to achieve the related requirements and finally running everything on the virtual machine. From a financial point of view, besides the certain chip, everything else will be created and run them on our own computer. The instructor will help to provide the chips we need, so actually there are no financial requirements for this project.

5. Testing and Implementation

5.1 Interface Specifications

Our main software interfaces are defined in section 2.4 of our design plan. They include kernel level WiFi subsystems and userspace tools / protocols for interacting with the kernel level subsystems.

5.2 Hardware and software

One of the main software tools that we will be using is the Linux strace tool. This tool is a debug / diagnostic tool used to debug interactions in and with the linux kernel. Besides this most of our testing will likely be printing out calculated statistics in a debug like

fashion. There is no additional hardware used for testing other than the physical target of our proof of concept.

5.3 Functional Testing

As mentioned in section 2.4 of our design plan, we will develop our implementation in a modular way. Does this will help us individually test each functional block in a black box testing type of way. To validate that a module was implemented the right way, we will develop test cases that given specific parameters into each module we will calculate the expected outputs of the module and compare that against the actual output. For example for the protocol signalling we will test for when the change in interference power increases over time and verify that the exclusive region (or conflict set) shrinks and when the change in interference power decreases the exclusive region grows. These types of tests will be accomplished in a unit test like fashion for all modules and shall cover all scenarios.

5.4 Non-Functional Testing

When it comes to non-functional testing, this is testing to verify our non-functional performance requirements are met from the system as a whole. This is planned to be done using debug messages that will print out / log statistics over time that then can be aggregated together to determine if our performance requirements have been met or not.

For example:

- Package loss rate can be calculated by monitoring / printing & calculating # of packets lost during transmitting, due to no acknowledgement back from target.
- Latency can be calculated by aggregating when a client sends a packet and the receiver receives it. The difference will be the latency (assuming clocks are synchronized).

5.5 Process

Unfortunately, we haven't been able to implement any testing since we had difficulties at previous stages. However, our testing plan will look like this as of our plan :

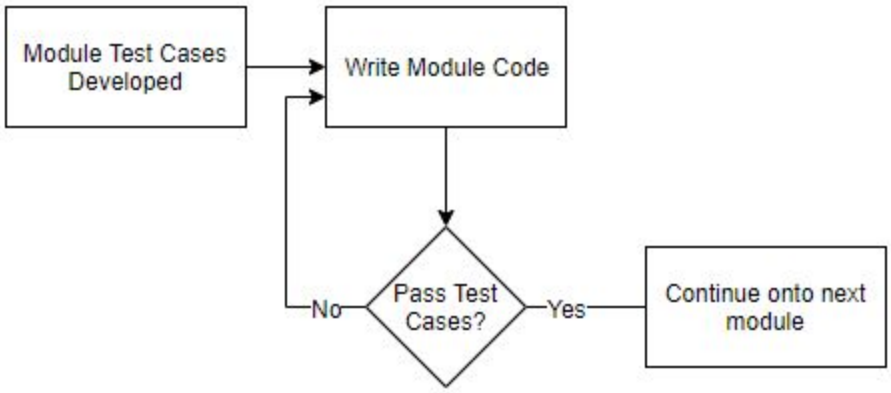

Fig. 5.5 Planned Testing Process

- Our test engineer will develop test cases for each module and write them
- Implementation of each module will be developed after test case has been written
- After Implementation has been developed then the test cases will serve as acceptance tests to determine whether or not that module behaves as it should.
    - If the tests fail, then go back and develop and try again.
    - If tests pass, then continue to the next module.

5.6 Results

Since we have not implemented any testing, thus there is no result available.

6. Closing Material

6.1 Conclusion

Throughout this semester we have tackled the tasks of learning OpenWrt and the three academic papers referenced. Tackling this new territory for the team presented to be quite a challenge but our advisors have helped us stay on track and power through the challenges.

In our second semester of the project. We had finished the pseudo code of PRKS algorithm and done identifying the parameters that requires. We have also acquired the hardware router and aim to virtualize it since we cannot test it in field at the lab since we have to maintain social distancing. However, the implementation does not go well as planned. We had more trouble while modifying the source code of IEEE80211. We try to look through the nl80211 source code and header file and the cfg80211's source code and

header files to find a place to implement, but we fail to finish it. Therefore, we modify our final deliverable to create a plan and strategy for the PRKS algorithm's implementation.

Each member in our team had devoted a decent amount of time digging into the project. Zhenwei had been making good documents throughout our project. He carefully noted down each meeting and made reports each week. Dylan has been working hard on the algorithm understanding, and he also helps with the team to ensure the knowledge is sufficient for all of us. Zequn and Yulin were working on identifying the parameters that appeared in the algorithm papers. Then they were digging into the source code to find any related functions that did help the implementation. Shaohang was working on the hardware virtualization and cross compilation for the first half of the project. Then he joined Zequn and Yulin to look for a possible implementation solution. Jiawei wrote the pseudo code for the multiple algorithms we use. He has been doing lots of good work and pushed the project forward.

Even though we did not come up with a full solution to smart agriculture. Our team still had a step by step methodology for the implementation. We believe this will help other teams if they wish to continue this project.

6.2 Appendices I - implementation strategy

1. In the first stage, we divide our group into 2 sub teams in order to speed up our progress:

    a. First team spends all time on understanding algorithms which will be implemented (PRKS Algorithm, PKTRT Algorithm, ONAMA scheduling) especially PRKS Algorithm because it participates a lot in the Linux Wi-Fi system. The ONAMA scheduling is TDMA scheduling inside of the architecture figure of PRKS Algorithm scheduling.

    b. Second team tries to run OpenWrt within a virtualized environment (QEMU emulator) to cross-compile hello world application.

2. In order to keep us on the same page we use trello and gitlab project to set up our work schedule and keep track of our work. Recently, we have only used the Gitlab project because compared to Trello we can push our codes to the working repository and the UI is more friendly for our project.

3. A very important implementation strategy is that assigning a team member or a team to understand the Linux Wi-Fi system, to look through the source code like nl80211 or mac80211 and try to find out the functions we needed for the program. Based on our experience, it took my teammates plenty of time on this.

Reference:

Hongwei Zhang, Xiaohui Liu, Chuan Li, Yu Chen, Xin Che, Feng Lin, Le Yi Wang, George Yin, Scheduling with Predictable Link Reliability for Wireless Networked Control, IEEE Transactions on Wireless Communications (TWC), 16(9):6135-6150, 2017