Open-Source Prototyping of 5G Wireless Systems for UGV/UAV

DESIGN DOCUMENT

Team 36

Client: Iowa State University

Adviser: Hongwei Zhang

Team Members: Nathan Whitcome, Ibro Tutic, Andrew Eschweiler, Samuel Stanek, William Byers, Nicholas Lorenz

Email: sdmay20-36@iastate.edu

Website: http://sdmay20-36.sd.ece.iastate.edu/

Executive Summary

Development Standards & Practices Used

- 3GPP
- E-UTRAN
- EPC
- IEEE

Summary of Requirements

Primary Requirement:

- Ensure per-packet communication reliability while achieving high throughput/concurrency

Other Requirements:

- Low Latency
- High Throughput
- High Reliability

Applicable Courses from Iowa State University Curriculum

- CPRE 308 - Operating Systems
- CPRE 489 - Computer Networking and Data Transfer
- CPRE 430/530 - Network Security
- CPRE 543 - Wireless Network Architecture
- COMS 486 - Fundamental Concepts in Computer Networking

New Skills/Knowledge acquired that was not taught in courses

As a team we needed to learn about how network scheduling algorithms helped increase reliability and throughput and lower latency. To do this, we first needed to gain an understanding of the various tools needed to simulate the networks in various environments. For this project, we are studying how to meet the above listed requirements in highly mobile vehicular networks. So just to begin, two simulators are needed: one to simulate the network and another to simulate traffic conditions and positions of vehicles. The network simulation is done by software called Open Air Interface and the traffic/vehicle simulation is done by software called SUMO.

Both simulators need to interact with each other, so the network stack has access to the vehicle positioning data from SUMO. Once this foundation for the project was acquired, to develop the algorithm two research papers were read on algorithms designed for high throughput, low latency, and high reliability. A proposed scheduling algorithm called PRKS was proposed to help meet the reliability requirements of the project, however, it needs to be fully implemented and tested in simulators to verify its performance vs current solutions. PRKS is built on a scheduling algorithm called PRK and PRKS is the basis of a scheduling algorithm that can meet the requirements of this project. It provides a high level of reliability by coordinating nodes that are close to each other. Ensuring that the reliability between individual nodes is extremely high bubbles up to the network. This provides high network reliability for nodes in different network and environmental conditions without a prior knowledge of these conditions

To apply PRKS to ground vehicles, some extra work needs to be done because vehicles are highly mobile in most use cases. This is detrimental to the performance of PRKS because there is no way to do predictable interface control in a highly mobile setting. To combat this, cyber-physical scheduling (CPS) is applied to PRKS. This creates a geometric approximation of the PRKS scheduling algorithm. Applying CPS to PRKS allows vehicles to know of each other locality without dedicating large portions of network bandwidth to transport this information to and from each vehicle.

Table of Contents

List of figures/tables/symbols/definitions

OpenAirInterface (OAI) - Open source software that simulates 3G, 4G, or 5G communication between two devices.

SUMO - Open source software that simulates traffic patterns on a given part of the world.

Unmanned Ground Vehicle (UGV) - A mode of transport, such as a car or truck, that is controlled remotely.

1 Introduction

1.1 Acknowledgement

This project would not be possible without the technical advice, planning advice, and material support of our faculty advisor Hongwei Zhang, and his doctoral student Chen Ye Lim.

1.2 Problem and Project Statement

This project deals with the lack of modern 5G implementations that allow for low latency and high throughput and reliability specifically in highly mobile networks. Current solutions for 5G networks do not guarantee any type of reliability between two nodes, especially in highly mobile environments like communication between ground or air vehicles. It involves developing and prototyping advanced 5G wireless solutions for unmanned ground and aerial vehicles, which have broad applications in domains such as connected autonomous transportation, smart agriculture, and advanced logistics. Furthermore, creating a highly reliable and low latency 5G network can allow self-driving cars to be safer and could potentially allow doctors to do surgeries from hundreds of miles away. This would keep people safer in more ways than one.

To create a solution to this problem, a new network scheduling algorithm needs to be developed that can help reorganize connections between nodes in highly mobile environments. Two proposed algorithms exist that can meet the requirements of this project, but they need to be implemented and tested in both simulation and real-world environments. These algorithms are called PKRS and CPS. PKRS is an algorithm that allows a network to achieve high reliability between two nodes that are close to each other. CPS takes this a step further and applies cyber physical scheduling (CPS) to extend PKRS to mobile networks, as it was originally intended for stationary nodes. CPS essentially extends PKRS to include data about the nodes positioning relative to each other without dedicating a lot of bandwidth solely to the transfer of positional information. This project will need to use a variation of the CPS algorithm in the MAC scheduling module of Open Air Interface to determine performance metrics associated with the new 5G implementation vs solutions that are currently available on the market.

1.3 Operational Environment

The operational environment for this project will be primarily in vehicles, which means the network will need to be able to operate under a variety of external conditions. However, this project does not deal with the hardware involved in the transfer of information, it is primarily the scheduling algorithm that allows the nodes to communicate with each other. However, we will need to ensure that things like storms, blizzards, and other natural disasters don't hinder the algorithms ability to reach high levels of packet reliability and throughput.

1.4 Requirements

Functional Requirements:

- Low Latency
- High Throughput
- High Reliability
- Interoperability with current solutions

Economic Requirements:

- Easy to simulate to avoid expensive hardware testing and implementations
- Code written to be maintainable (i.e. does not require days of work for simple changes, costing money)

1.5 Intended Users and Uses

This project can have multiple types of end users, as it is an extension of the Internet that almost everyone uses today. Some examples of these are:

- Surgeons - With low latency and high reliability, surgeons would be able to operate on patients from hundreds of miles away, giving everyone more access to better healthcare.
- Self Driving Cars/UGV's- Meeting the requirements of the project will also allow cars to communicate safety information between each other in near real time.
- Military Applications - This project would also allow for the control of ground vehicles from a distance. Transport trucks and scouting vehicles could be controlled in near real time by an operator far from the site of the vehicle, increasing the safety of those involved.

1.6 Assumptions and Limitations

Limitations:

- 5G signals are low range and degrade quickly
- May apply for ground vehicles or air vehicles, but not both due to environmental differences

Assumptions:

- Supports networks where there is fairly flat ground between nodes (i.e. no mountains).
- Supports vehicle networks where vehicles (i.e. nodes) aren't extremely sparse.

1.7 Expected End Product and Deliverables

Algorithm Simulation/Extension (January - Febuary 2020)

This is the primary deliverable associated with the project. To achieve the requirements of the project, the algorithm needs to be implemented in Open Air Interface and tested by using SUMO to deliver traffic data and vehicle positions which Open Air Interface will use to simulate the network scheduling algorithm. To ensure that this deliverable meets the requirements listed above, the new scheduling algorithm will be tested and its performance measured to allow us to compare it to current solutions.

Report / IEEE article <IEEE Communications Magazine>

Because this project deals with a fairly new and upcoming technology, it would be extremely valuable to report our work done on the 5G communication between mobile nodes so others have access to our research on the subject. This work would be in parallel with the algorithm design and simulation. The plan is to try and have something published to further the research on low latency, high throughput and reliability mobile networks.

2. Specifications and Analysis

2.1 Proposed design

Our proposed solution is to integrate a 5G algorithm written by Hongwei Zhang and his team into Open Air Interface and to simulate the implementation using with SUMO. We will do this by spending two months preparing and learning about the software needed, writing the software, and integrating it, and the rest testing with the hardware and working on a final report.

All our code is following standards set by 3GPP, including E-UTRAN and EPC. We are modifying the code rather than changing the stack completely, so it should follow those standards before and after.

2.2 Design Analysis

Due to the scope of the project focusing heavily on scheduling algorithms and using the Open Air Interface/SUMO, the team will need to research and understand some of the requirements that would need to be met.

As a team we have yet to see the algorithms of our design implemented and through our own research are currently gaining an understanding of how to utilize, OAI, SUMO, and other 5G utilities.

As it stands, we are all still learning about OAI and SUMO, so there haven't been any reasons to modify or change the scope of our project. A strength is that using OAI and SUMO we will be able to continuously test/debug any problems we have with the algorithm we are trying to implement. However, the raw content is new to each member to the time to learn the material will be large.

2.3 Development Process

We will be using the Waterfall model for our project. We were looking at Agile initially but we don't have a specific customer and our project cannot have continuous updates since it relies on a lot of research beforehand. We also cannot meet in person every day due to conflicts in schedules for team members. Waterfall lends itself to individual stages, which is the only real way we can do our project.

2.4 Design Plan

The emphasis of the packet scheduling algorithms currently being used in Open Air Interface is throughput in a non-mobile network. The plan for our design is to replace the scheduling algorithms already in Open Air Interface with a new algorithm provided by our faculty advisor. This implementation must fit within the existing architecture of Open Air Interface. Then, using SUMO we will be able to simulate a network of vehicles to feed data into Open Air Interface to simulate how the scheduling algorithm will perform. Using the metrics from these simulations, we will be able to compare to the previous benchmarks for other algorithms and determine if we've met our requirements as outlined in section 1.4.

3. Statement of Work

3.1 Previous Work And Literature

There aren't similar products for mobile networks, current 5G implementations focus on non-mobile nodes (so like setting up a commercial 5G system for customers, like Verizon). This project focuses a lot more on situations like communications between air/land vehicles and how to optimize the network for maximum throughput and reliability.

To begin researching the algorithm to use in this highly mobile network, the team familizared itself with Dr. Hongwei's work on 5G scheduling algorithms, this research included the following published articles:

- Scheduling with Predictable Link Reliability for Wireless Networked Control
    - Hongwei Zhang, Xiaohui Liu, Chuan Li, Yu Chen, Xin Che, Le Yi Wang, Feng Lin, George Yin
- Cyber-Physical Scheduling for Predictable Reliability of Inter-Vehicle Communications
    - Chuan Li , Hongwei Zhang , Jayanthi Rao , Le Yi Wang , George Yin
- Probabilistic Per-Packet Real-Time Guarantees for Wireless Networked Sensing and Control
    - Yu Chen, Hongwei Zhang, Nathan Fisher, Le Yi Wang, George Yin

These papers described initial versions of the algorithm that we are going to implement (in bold above) and how the test bed was set up to ensure the proposed algorithm performed as good or better than previous solutions. The bold paper Scheduling with Predictable Link Reliability for Wireless Networked Contro**l** (PKRS) defines the algorithm to achieve a high rate of reliability, throughput, and low latency. This algorithm works by defined an exclusion region around nodes to avoid interference in wireless transmissions through a specific parameter that depends on desired link reliability. (See figure below).

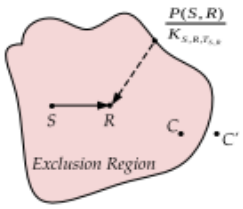


Fig. 1. PRK interference model

The PRKS model defines and exclusion region for each link in the network (S, R in the Fig 1) around the receiving link (R in this case). The node C is in the exclusion region because the strength of the signal from C to R is greater than the ratio of the strength of the signal from S to R to the parameter **K** needed to take into account the presence of background noise and interference in the entire network. This parameter is chosen to maintain some minimum link reliability between two nodes.

The paper on Cyber-Physical Scheduling (CPS) for Predictable Reliability of Inter-Vehicle Communications applies the PRKS interference model to vehicular networks by extending the model to provide a geometric approximation to allow it to work in highly mobile networks. The initial PRKS model applies only to very low mobility stations. The CPS scheduling extends this model by instantiating the parameter K (same K as above) at every node. This algorithm leverages control theory to allow every link instantiated with the PRK model and the local signal maps that contain average signal power between S, R, and every other close-by node C that may interfere with the reference node. In this manner, the CPS algorithm is extremely similar to the PRKS model, except that is has been extended to account for highly mobile networks by having each node instantiate its own K parameter and have its own exclusion region for which it is responsible for.

3.2 Technology Considerations

The only technology consideration that will work for this project is a 5G simulator called Open Air Interface (OAI). The OAI simulator providers the high-level abstraction needed to both simulate the algorithm and to eventually run the new algorithm on actual 5G communication devices. The OAI codebase contains nearly 2 million lines of code (including some build scripts and documentation) so it is extremely difficult to begin to understand all of the "moving" parts of the simulator. Luckily our scheduling algorithm is going to be implemented in the MAC layer, so that significantly reduces the amount of code that has to be altered outside of this module.

OAI has extensive and very specific hardware requirements. It can only be ran on an Intel CPU with all power saving features disabled (to avoid any variance in clock speed) and all c-states disabled (again to avoid any issues due to the system wanting to enter a power saving mode). It can not be reliably run in a virtual machine because the interface between the VM and the hardware might not allow certain CPU flags to be addressed by the simulator, so it needs to be developed and ran directly on the hardware using Ubuntu 14.04 with a low-latency Linux kernel to ensure low overhead for processing time.

We investigated a new simulator called NS-3 that was written exclusively in C++ and it had a much more modern design and feel to it, but it only allowed for the simulation of networks. There was no way to go directly from a simulation to hardware testing like there is in OAI. This means that if we chose to use NS-3, we would need to write the code two times and learn two different simulators, which would take up a lot of time. Because of this, it is advantageous to use OAI right off of the bat, regardless of the steep learning curve, as it will allow us to run the algorithm directly on hardware when we get to that point.

The final piece of technology that we need to use is called SUMO. This is a traffic simulator designed to model vehicular networks. SUMO provides an interface (called TraCI) that uses socket communication to send vehicle positions to another socket. SUMO allows users to draw their own maps and model their own streets and traffic flows, so our team could model Iowa State University's campus for example.

OAI contains legacy code that indicates some integration with SUMO, however, it seems that there are comments throughout that say that there are issues with how certain things are indexed. This will require some more investigation to determine how serious an issue these errors are if any. Luckily, the code seems outline the general design of how SUMO sends positioning data for nodes in OAI.

3.3 Task Decomposition

Below is an outline of the various tasks for our team and their timeline. Each week contains key tasks for the specific teams to work on. Our team is currently broken up into those who work on the simulator and its integration with SUMO and those who do the algorithm analysis and implementation in OAI.

1. October
    1. Week 1
        1. Algorithm Analyzers
            1. Read PRKS, CPS article. Gain understanding of how the algorithms work and their performance
        2. Simulation Specialists
            1. Install Ubuntu, OpenAirInterface, and SUMO installed on a PC.
    2. Week 2
        1. Algorithm Analyzers
            1. Look at OAI to find out where the new scheduling algorithm will "live"
            2. Analyze the OAI stack around the scheduling algorithm to understand input and outputs and how they are connected
        2. Simulation Specialists
            1. Run tutorials on OAI and SUMO. Look into OAI code to have a better understanding of how it works.
            2.
    3. Week 3
        1. Algorithm Analyzers
            1. Analyze current algorithms, see if any changes can be made to make it more performant
                1. Will help us gain an understanding to future design of new algorithm
        2. Simulation Specialists
            1. Do more tests with OAI and SUMO. Work on integration of the two.
    4. Week 4
        1. Algorithm Analyzers
            1. Begin initial algorithm redesign, think about what environmental factors affect performance
                1. Vehicle movement
                2. Sparse traffic (greater distance between nodes)
        2. Simulation Specialists
            1. Go more in depth on OAI code.

2. November
    1. Week 1
        1. Algorithm Analyzers
            1. Continue implementing and testing the initial algorithm
            2. Use lessons learned from initial algorithm implementation to begin outlining the new algorithm design
                1. Will the inputs/outputs be the same?
        2. Simulation Specialists
            1. Continue to understand OAI code and learn how previous networking standards worked. More SUMO tests.
    2. Week 2 - 4
        1. Algorithm Analyzers
            1. Begin implementing new algorithm based on research article and above initial algorithm design
            2. Continue implementing and testing the initial algorithm
            3. Ensure code is readable and testable, with proper modularity
            4. **Begin porting physical layer abstraction to version 1.1.1 of OAI**
        2. Simulation Specialists
            1. Continue to understand OAI code and learn how previous networking standards worked. More SUMO tests.
            2. **Begin porting physical layer abstraction to version 1.1.1 of OAI**
3. December
    1. Week 1
        1. Prepare for 30 minute design presentation
        2. **Continue working on physical layer porting.**
    2. Week 2
        1. Reevaluate roles on team, as initial development phase may be over and the same roles might not apply.
        2. **Continue working on physical layer porting.**
4. November, December, January: Algorithm Implementation (roles of team members?)
    1. Roles
        1. Product Owner - Figure out what features/work to prioritize
        2. Developer - Develop and implement the scheduling algorithm
        3. QA - Quality Analyst, ensure any new code is tested and verifies nothing old is broken through regression testing.
        4. Report writer(s) - Begin writing report with information about the algorithm implementation
    2. Extending existing algorithms for more mobile application (UAVs/UGVs) [2-3 weeks]
    3. Continuous/iterative/spiral development, prototyping & testing [4-5 weeks? Need more time?]
    4. Performance evaluation [3 weeks]
        1. Comparison vs current 5G implementations

5. Report / article writing: algorithms, implementation, evaluation results [Parallel task]

5. February, March, April: Hardware Implementation, Report work (roles of team members?)
    1. Roles
        1. Lab testers - analyze implementation in lab setting
        2. Field testers - analyze implementation in field setting
        3. Report writer(s) - Finalize the report and include comparison of new 5G scheduling algorithm vs current implementations
    2. Lab testing with SDRs
    3. Field testing with SDRs
    4. Demo [2-3 weeks]
    5. Report / IEEE article <IEEE Communications Magazine> [parallel]
    6. Potentially communicate with John Deere for UGV applications [parallel]

3.4 Possible Risks And Risk Management

This project has a lot of potential risks. OAI is an extremely large codebase with a lot of functionality. From the initial work with OAI, it is clear that it is missing key documentation in various locations and sometimes has code that has something wrong with a comment saying it needs to be fixed. Furthermore, the OAI simulator recently dropped support for SUMO so we needed to backtrack to find the most recent version of OAI with (potential) SUMO support.

OAI also requires very specific hardware requirements as mentioned in Section 3.4 so it would be extremely difficult to track down an issue that could be due to code or to an issue with the hardware that it is being ran on. OAI also has 13 total test cases that only check for things like segmentation faults and execution errors in very specific locations in the code. These tests would not give any indication that a particular scheduling algorithm isn't work or where to begin debugging. This will make development much harder as it will be very difficult to verify that the code is doing what it needs to be doing.

There is only one way to manage these risks and it is to thoroughly understand the interface between OAI and SUMO and to understand exactly how the MAC layer (and other layers in the stack) work. Even this will be hard, as we will have to dedicate even more time to just understanding how everything works before we can start implementing anything.

3.5 Project Proposed Milestones and Evaluation Criteria

What are some key milestones in your proposed project? Consider developing task-wise milestones. What tests will your group perform to confirm it works?

The first key milestone in our project is getting OAI working. This is the first major portion of our project as our whole project relies upon the simulator working and producing accurate results. OAI is divided up into three subsystems, each which need to be compiled and ran separately. These subsystems mimic systems found in wireless networks and allow for maximum modularity and customization.

The second key milestone will be getting the OAI and SUMO integration working and figured out. This will be another integral part of our project because we will need to use the traffic data to validate that our algorithm produces the results that we expect.

The final milestone will be seeing if the algorithm that we create meets the required project specifications:

- Low Latency
- High Throughput
- High Reliability
- Interoperability with current solutions

3.6 Project Tracking Procedures

Our group will use Trello and GitLab to track progress and manage features and work. The Trello board has cards that show what everyone is working on currently and GitLab will be used for code reviews so everyone can look at proposed changes before we commit them into the master branch. This will hopefully avoid easy to catch mistakes that might get missed by the person submitting the pull request.

3.7 Expected Results and Validation

The desired outcome of this project is to have a new 5G scheduling algorithm that guarantees high packet reliability, high throughput, and low latency communications for vehicular networks. This can then be applied to other mobile networks like planes and UAVs. Hopefully, this project will also create a starting point for other teams that will use OAI and SUMO to generate test data for network simulations. Getting the simulator working was one of the hardest parts of the project as there were a lot of moving parts and sparse documentation.

To confirm the solution works at a high level, we will need to compare the results of our custom 5G scheduling algorithm implementation to current solutions to see if it ensures a higher reliability than current solutions. Current 5G scheduling algorithms do not handle mobile networks and are targeted toward wireless carriers providing internet to consumers who pay for the service. This project extends 5G systems to potentially autonomous vehicles and other use cases where nodes are mobile and high packet reliability is needed.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

Assignment/Task	9/9-9/23	9/23-10/7	10/7-10/21	10/21-11/4	11/4-11/18	11/18-12/2	12/2-12/16	Winter Break	1/13-1/27	1/27-2/10	2/10-2/24	2/24-3/9	3/9-3/25	3/25-5/1
Read 4G LTE/5G book														
Research PRKS Paper														
Research CPS Paper														
Research SUMO														
Research OAI and specific install steps														
Set Up Hardware/Operating System														
Install OAI														
Install SUMO														
Verify/Run OAI on Server														
OAI/SUMO Integration Work														
Algorithm Development								??????????						
Verify and Test Simulation														
Begin Writing Report														
Compare Simulation Results to Control														
Begin Hardware Deployment														
Finalize Report and Project														

This timeline has three main “stages” of the project. The first stage, shown in green, is the research stage. For this stage, the team will be researching 4G, LTE, and 5G radios, wireless networking algorithms, and the open-source software we will be using. This is extremely important because our team does not have much background in these areas, and knowledge with these topics is necessary for success. We also will not have some of the required resources to progress meaningfully on our technical tasks until late October. This stage has no deliverables.

The second stage, shown in yellow before winter break, is our technical stage. During this stage, the team will begin setting up the hardware and operating environment, installing OAI and SUMO for use in the third stage. The bulk of our technical work is contained in the OAI/SUMO integration and algorithm development. Our goal is to finish these tasks before winter break so that we have plenty of time to test and gather data for the reports. There will be no deliverables to the customer from these tasks, but our internal deliverables will be a working interface between OAI and SUMO, and a working eNB MAC layer in OAI for the scheduling algorithm.

The final stage of the project, shown in yellow after winter break, is the verification and documentation stage. We will be working to test our implementations from the technical stage and produce data that we can use in our reports. The reports are the main deliverable, and hardware deployment is a secondary deliverable at the end of the Spring semester.

4.2 Feasibility Assessment

We believe the majority of the project is feasible, however the hardware deployment deliverable may be unfeasible. The main challenges we have identified lie with the OAI codebase. The existing codebase is going to be difficult to work with and we are expecting to spend a lot of time debugging and learning how parts other than what we're developing work. The existing interface between OAI and SUMO also comes with known pre-existing issues, which means effort will be required before we are even able to start our integration. We don't expect these issues to prevent the delivery of our comparison reports, but they will possibly be exacerbated when we switch from the hardware testbed to the hardware deployment. This could cause the deployment deliverable to be delayed too much to be finished by the end of the semester.

4.3 Personnel Effort Requirements

Assignment/Task	**Estimated Hours (Total)**
Read 4G LTE/5G Book	12
Research PRKS Paper	12
Research CPS Paper	12
Research SUMO	12
Research OAI and Specific Install Steps	12
Set Up Hardware/Operating Environment	5
Install OAI	5
Install SUMO	5
Verify/Run OAI on Server	15
OAI/SUMO Integration	30
Algorithm Development	45
Verify and Test Simulation	15
Writing Report	20
Compare Simulation Results to Control	10
Hardware Deployment	30
Finalize Report and Project	20

The estimated times in this table are total hours, not per member. The research tasks involved in this project are fairly technical and do require a decent amount of effort for each member to gain understanding in each topic. It's estimated that each research task will take around 2 hours for each member of the group to complete. Setting up the hardware, operating environment, and installing OAI/SUMO are relatively simple tasks, but they will have some technical aspects and processes to follow. For this reason, we are estimating it would take one member 5 hours to complete each one.

As the team digs into the later tasks of the project, they become much more technical and have much larger scope. The OAI/SUMO integration will require extensive coding and debugging of the interfacing between the two simulators. With two members focused on this task we are estimating each one will put in 15 hours to complete it. The algorithm development is possibly our biggest challenge. The OAI codebase is large and not well-documented, which means the coding and debugging for this portion of the project will be susceptible to a lot of hidden issues. For this reason we are estimating it will take up to three team members 15 hours each to complete this task. The final large task will be hardware deployment, which will include integrating the code we have developed into a new hardware environment outside the testbed. We are expecting some debugging will be required and are estimating 3 members will take 10 hours to complete the task.

4.4 Other Resource Requirements

This project will require the use of hardware capable of running both Open Air Interface and SUMO. A test bed of hardware specifically set up for Open Air Interface will facilitate quick and accurate simulations, reducing the need for implementation specific hardware. For team related code repository management and software access, a Linux box will also be needed that can be connected to via SSH. This system will have Open Air Interface and SUMO installed so that basic changes and tests can be run without the test bed.

4.5 Financial Requirements

No financial requirements have been defined for this project. The software we are using such as Open Air Interface and SUMO are open source. The faculty advisor has also indicated that any hardware needs for the project have been or will be taken care of at no cost to the project.

5. Testing and Implementation

5.1 Interface Specifications

In terms of software interfacing, we are now tasked with porting over physical layer abstraction to a newer version of OAI (version 1.1.1). The older version of OAI proved to be extremely difficult to use and lacked basic build scripts, so we moved over to a newer version of OAI. This abstraction (called OAISim) will handle estimations in the physical layer so we can run simulations with multiple nodes, which currently is not possible with OAI.

OAI Sim

- Designed to aid in physical layer abstraction. This will estimate physical layer calculations, so the CPU/hardware does not need to maintain a large amount of accuracy and waste performance on that. This allows us to run large simulations with many vehicles, so we can better test the resultant algorithm.
- This layer will also handle the traffic positioning. We are building a VehiclePositioning module that will get updates from SUMO and update the vehicle positions in OAI to allow them to move around. This is described below.
- The VehiclePositioning module will feed data into OAISim, which well then update the structures associated with each node (as x, y) positions.
- Once the positions are updated, the connections between nodes will be adjusted according to the scheduling algorithm implemented in the MAC layer.

Vehicle Positioning Module

- This module is designed to abstract the logic of handling the transfer of vehicle data from SUMO to OAI. It will be designed to update all vehicles at some given interval (1 ms or 10ms) and these positional updates will be applied to each node in the network.
- All the logic needed to format and handle the data from the TRACI server will be handled here. The module will be used in OAISim to get positioning data without cluttering the physical layer abstraction with code that does not really have anything to do with it.

5.2 Hardware and Software

Hardware Used – None so far but explain the radios.

Software Used – Open Air Interface 5G

The only technology consideration that will work for this project is a 5G simulator called Open Air Interface (OAI). The OAI simulator providers the high level abstraction needed to both simulate the algorithm and to eventually run the new algorithm on actual 5G communication devices. This software allows us to test scheduling algorithms to verify that they work and produce results that make it worth the work to deploying to hardware. This is an important step in the development process because deploying to hardware is expensive in both terms of time and money.

5.3 Functional Testing

Unit Testing:

- The unit testing that will be done in our case will be specifically in the Vehicle Positioning Module. This testing will involve making sure that updates are made at the requested interval (i.e. if an input interval is 1ms, then the module updates every 1ms and so on.).
- Another unit test will be ensuring the simulator builds with the new physical layer abstraction when the proper flags are provided. This will make sure that any updates don't ruin the link between OAISim (the physical layer abstraction code) and the base OAI simulator.

Integration Testing:

- Integration testing will be simple in this case. Because OAI is 2 million lines long, there is no way to feasibility test a large portion of the functionality. However, we can test that our build scripts call the proper functions/modules and that they instantiate connections to the SUMO server that will need to be running.
- We can also verify that the positional data from SUMO is being applied to the nodes in the vehicle network and that the algorithm is working as intended.

Acceptance Testing:

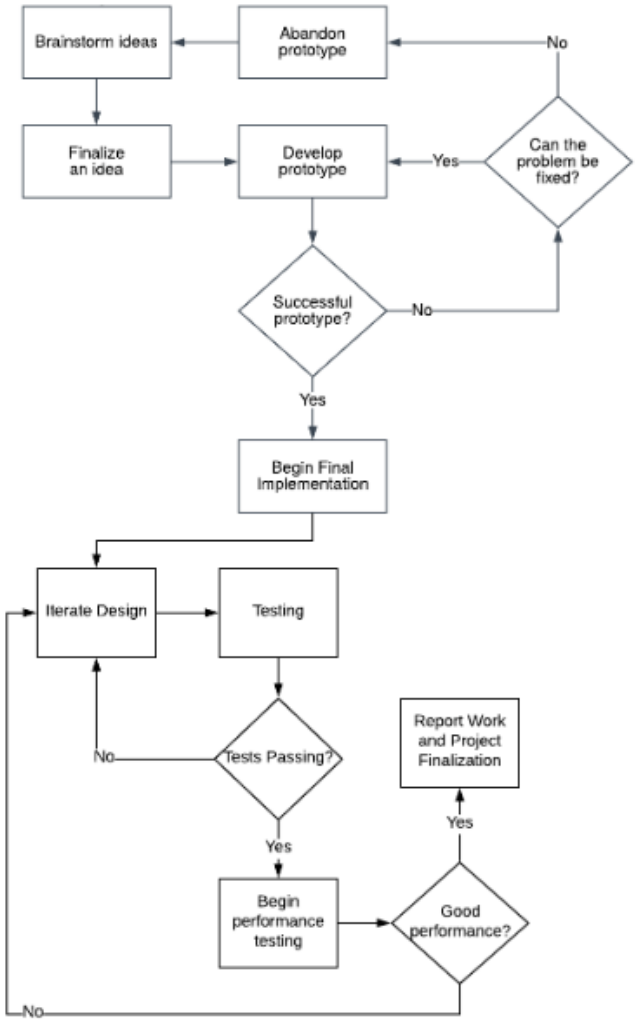
- Acceptance testing that will be done will be verifying that the output throughput, latency, and reliability match that of the implementation in the NS3 simulator. This will allow us to fix any bugs if we see any differences in the data vs the two simulators.

5.4 Non-Functional Testing

Testing for performance, security, usability, compatibility.

- Performance and usability testing will be done using SUMO. SUMO will provide positional data to OAI and will allow us to simulate the movement of actual traffic patterns. This will help us maintain near real world simulation situations and will help us determine how well our scheduling algorithm works.

5.5 Process



5.6 Results

-Results Obtained So Far

- The current state of the project is that we have the simulator working and can simulate uploads and downloads and verify that OAI is working. We still have more work to do in building a vehicle position module and to port over the physical layer abstraction from version .5.2 of OAI.
- We have learned a lot of the limitations of the various code bases of OAI. This has been an area of frustration for a lot of the project because version .5.2 is extremely hard to build and use but contains features that we need. Version 1.1.1 is much more up to date and is much easier to build and use, but it lacks critical features that we need to be algorithm development.

-Modeling and Simulation:

```
**********rb: 25 ***mcs : 10  *********SNR = 27.100000 dB (0.000000): TX 510 dB
(gain 5.246080 dB), N0W 30.000000 dB, I0 0 dB, delta_IF -161 [ (58,0) dB / (0,0)
 dB ]*************************
Errors (1/1 0/0 0/0 0/0), Pe = (1.000000e+00,0.000000e+00,0.000000e+00,0.000000e
+00) => effective rate 0.477143 (100.0%,0.477143,0.954286), normalized delay 0.0
00000 (0.000000)
```

Figure 1: Upload Test

Figure 1 shows an upload test ran in OAI. This tests the connection between the UE and the eNB and checks various performance factors and error rates. Things like speed, signal strength, error rates, packet losses, and latencies are available from this data.

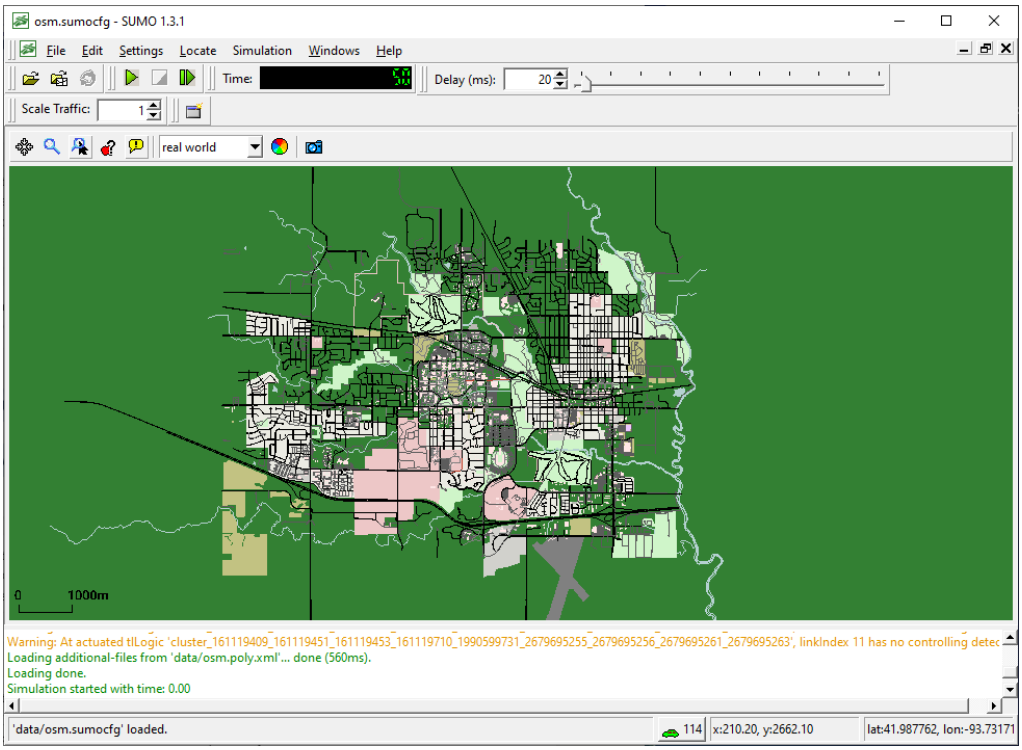


Figure 2: ISU In SUMO

In Figure 2 above, we simulated the Iowa State area and designed a simulation where vehicles can move around on the streets that exist around campus. This helps us understand how to use SUMO and how to begin exporting vehicle positional data from SUMO into OAI for our simulations.

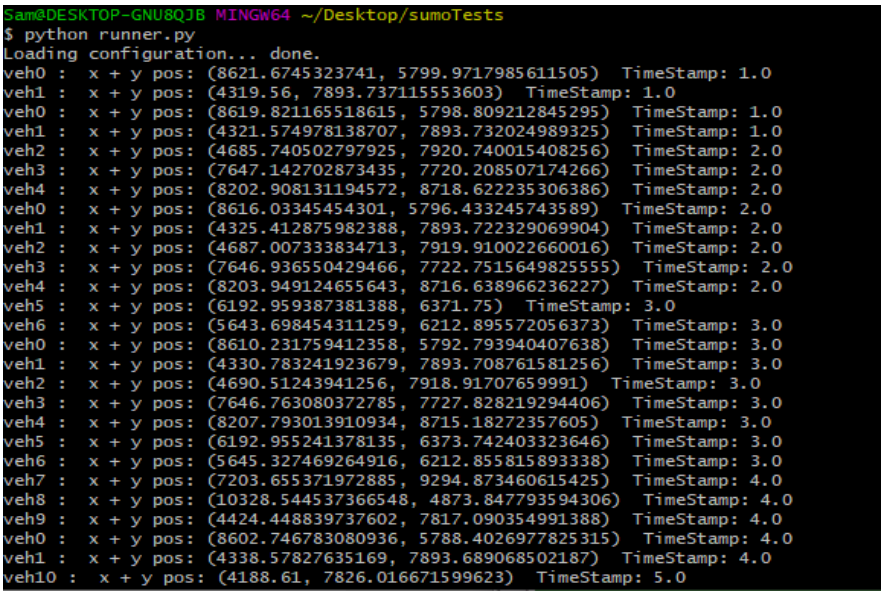


Figure 3: Information feed from SUMO

Figure 3 shows a live feed of information generated by SUMO when the program is running. In this instance, SUMO is started as a "server" and sends the info to the python script which is running as the "client". The python script uses the TraCI (Traffic Control Interface) libraries to send commands to SUMO using a TCP connection.

Implementation Issues and Challenges:

A massive implementation issue that has set us back a lot is the fact that we switched versions of simulators late into the semester. We spent the better part of two month studying version .5.2 of the simulator and attempting to get it working. From our work, we realized that it was extremely difficult to work with and required a large amount of previous knowledge to even get all the dependencies linked up. This version of the simulator included key features that are needed to simulate many UE (essentially vehicles in the network) and features that allow OAI to communicate with SUMO to get traffic data.

Because of difficulties in building version .5.2 of the simulator, we moved to version 1.1.1. This version was much easier to build, but it lacked physical layer abstraction and the software needed to communicate with a SUMO server for vehicle position data. So we are trading off ease of use with critical features that we need to being development of the scheduling algorithm. To even begin simulating, we need to port over the physical layer abstraction from version .5.2 of OAI so we can reduce the performance requirement of network simulations. This will allow us to simulate with hundreds of vehicles in a network. Without this feature, one UE (vehicle) would need an entire computer/server to be able to run properly, so with OAI's current state, large scales simulations are simply not possible.

Secondly, we need to create a Vehicle Positioning Module to periodically retrieve data from the SUMO server to update the locations of each of the vehicles. This means that it will need to update the positioning of the nodes deep inside of OAI by updating the struct associated with the situation geometry (essentially positions of each node). Once these two things are done, they will need to be integrated into a single file that can get built and ran via flag in the ./build_oai command. Only after these two things are completed can actual algorithm implementation begin because there would be no way to simulate a realistic situation without it.

6. Closing Material

6.1 Conclusion

So far, we have installed the simulator on our server and some of our personal devices and have gotten the simulator working in a very simple one node and one eNB scenario (seen above). We have ran into many problems with switching versions (which was a massive code overhaul by the maintainers, changing the architecture of the simulator almost entirely) and we have to implement things that were present in version .5.2 that are no longer present in the version we are using v.1.1.1. We have updated our roadmap to include the two primary things that we need to get up and running: physical layer abstraction and a SUMO module.

The physical layer abstraction is an element of another senior design team, so we will need to implement this portion before we can begin implementing the algorithm. Unfortunately, we cannot wait until they have their final results to use their ported code, so we have been set back in our final deliverable of the algorithm to get this work done.

This move from version .5.2 to version 1.1.1 has set us back far and we hope to have the physical layer abstraction and vehicle positioning module done by mid to late January. Given the size of the codebase and lack of documentation/comments in most of it, this will probably be more difficult than what we are anticipating. Given the level of effort needed to even complete the physical layer abstraction and vehicle positioning module, if we are able to complete that work that will open the door for anybody else to implement a new scheduling algorithm if we don't happen to get that far. Highly mobile networks is a very active area of research and both of the previously referenced work items will give others the ability to simulate large scale networks without using an old and clunky version of the simulator that lacks some of the new 5G features.

6.2 References

C. Li, H. Zhang, J. Rao, L. Y. Wang and G. Yin, "Cyber-Physical Scheduling for Predictable Reliability of Inter-Vehicle Communications," *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, Orlando, FL, 2018, pp. 267-272. doi: 10.1109/IoTDI.2018.00035

Navid Nikaein, Mahesh K. Marina, Saravana Manickam, Alex Dawson , Raymond Knopp , Christian Bonnet, "OpenAirInterface: A Flexible Platform for 5G Research" EURECOM

Latif, Imran and Kaltenberger, Florian and Nikaein, Navid and Knopp, Raymon. "Large Scale System Evaluations Using PHY Abstraction for LTE with OpenAirInterface" Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques Pages 24-30