IOWA STATE UNIVERSITY

CyWi: An Open-Source Wireless Innovation Lab for SmartAg, AR/VR, and Beyond

Final Report

Chenye Lim *Ryan Cullinan*

Jian Chew *Shay Willems*

Pawel Darowski *Tyler Beder*

SD-DEC19-02

December 11, 2019

1. Frontal Material

1.1 Table of Contents

2. Introductory Material

2.1 Problem Statement

Over the last decade, the development of Cyber-Physical Systems (CPS) and Internet of Things (IoT) has been in full force because of the multitude of advantages it can bring to everyday life. Despite many years of research, we are still at the infancy of IoT and Industry 4.0 capabilities. IoT can be used for a wide range of products such as SmartAg, connected autonomous vehicles, smart grids, and AR/VR. Powered with the capabilities of 5G, IoT promises to change every aspect of our lives. But to continue with this pace of innovation, researchers need space to run experiments and gather real-world data.

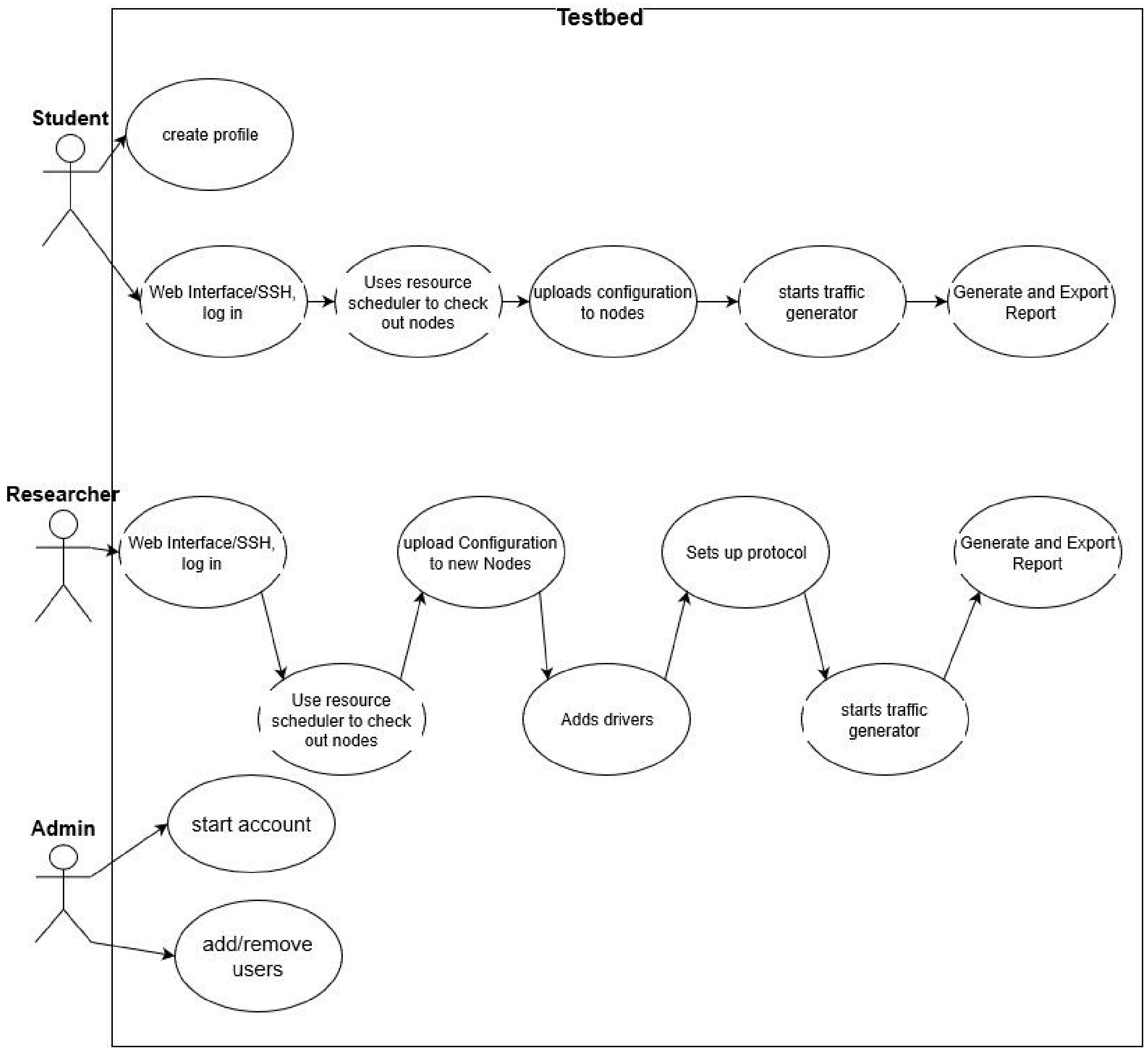

Testbed
Student
create profile
Web Interface/SSH, log in
Uses resource scheduler to check out nodes
uploads configuration to nodes
starts traffic generator
Generate and Export Report
Researcher
Web Interface/SSH, log in
Use resource scheduler to check out nodes
upload Configuration to new Nodes
Adds drivers
Sets up protocol
starts traffic generator
Generate and Export Report
Admin
start account
add/remove users

Figure 1: Use Case Diagrams

Iowa State University has decided to explore this field with the development of the CyWi lab. This lab consists of a multi-node testbed located on the Iowa State campus and is available to researchers and testers to run their experimentation code. This lab features the most bleeding-edge wireless innovation platforms as well as emerging wireless solutions. It also features 5G wireless for new learning, teaching, and researching across the globe.

2.2 Operating Environment

The CyWi's testbed is located in 3038 Coover Hall at Iowa State University (ISU). This is a climate-controlled room with a keycard secured door. The only people allowed into the lab room are those with keycards: the professor/client, the ISU Electronics Technology Group (ETG), and a handful of researchers. Users can access the testbed via the remote web interface, never via physical access to the lab. One wall has windows facing west but the blinds will be closed so sunshine never touches the equipment. While it is not expected for the hardware to experience anything but optimal operating conditions, our web-based service will be exposed to the Internet so cyber attacks, such as DDoS, are possible.

2.3 Intended Users and Intended Uses

The CyWi testbed is intended for two general types of users: students and researchers.

Students learn about wireless signals and protocols in their courses via lectures, assignments, and small projects. Theoretical knowledge is crucial. However, building upon that foundation with extensive lab experience configuring real-world hardware will improve students' understanding of the subject matter. Implementing wireless topologies such as mesh, star, and point-to-point could inspire future IoT developers. Comparing Zigbee and Bluetooth Low Energy performance, for example, over assorted ranges, signal strengths, and conditions will extend students' knowledge of technology strengths and limitations. CyWi will provide students the opportunity to experiment with a variety of popular, existing wireless technologies and to expand their understanding in a safe environment.

Researchers, on the other hand, will appreciate the cutting-edge communication technologies represented by the CyWi testbed. Access to powerful and configurable software-defined radios (SDRs) will allow researchers to study a wide spectrum of new heterogeneous networks and explore exciting innovative ideas. Researchers will evaluate performance monitoring and statistics after each experiment to determine the feasibility of their chosen path. Emerging technologies such as 5G, SmartAg, augmented reality, virtual reality, and Internet-of-Things (IoT) will generate opportunities for decades of continuous communications development. As Figure 2 shows, CyWi is particularly equipped to handle experiments of various throughput and latency specifications.

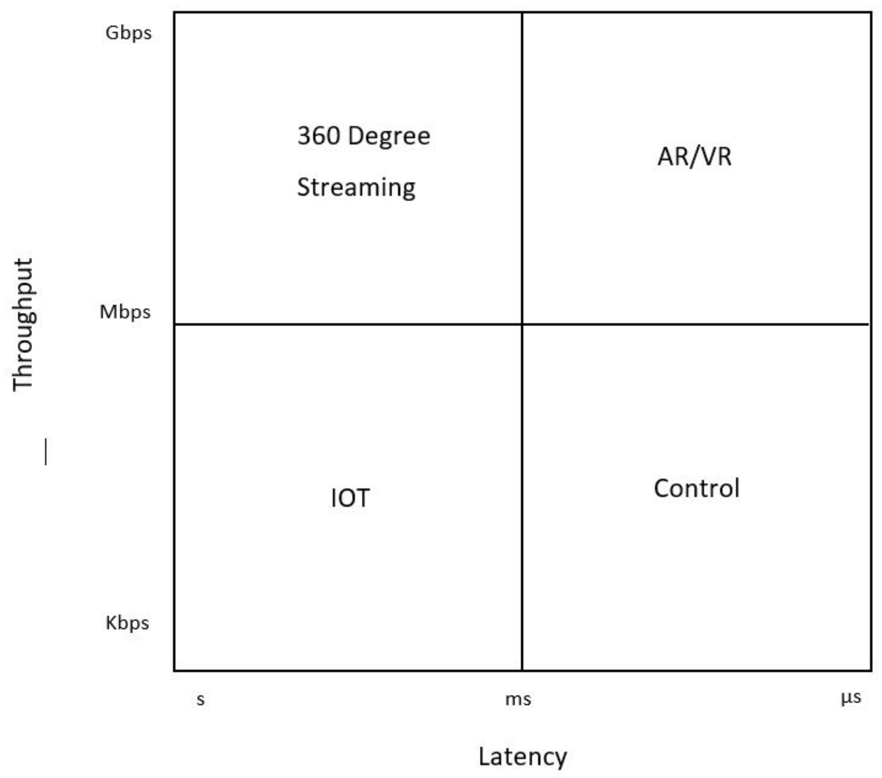


Figure 2: Bandwidth to Latency Relations

2.4 Assumptions and Limitations

2.4.1 Assumptions

- Environmental conditions in the lab room will remain nominal for electronics.
- Internet access will be sufficient and reliable.
- Outside wireless signal interference will be negligible.
- Users will be students and researchers so they will be somewhat familiar with testbeds.
- Users will participate in reporting bugs if or when they arise for continued improvement.
- ISU Electrical and Computer Engineering department will publish or advertise that this testbed is ready for students and researchers to use.

2.4.2 Limitations

- All software used must be open-source.
- Lab room has space for only 110 total nodes.
- Budget is a factor so the amount of SDRs is limited.
- Two semesters is the maximum time to spend designing and developing this project.
- Some elements of this project are new to us and required extensive research

3. Specifications and Analysis

3.1 System Design

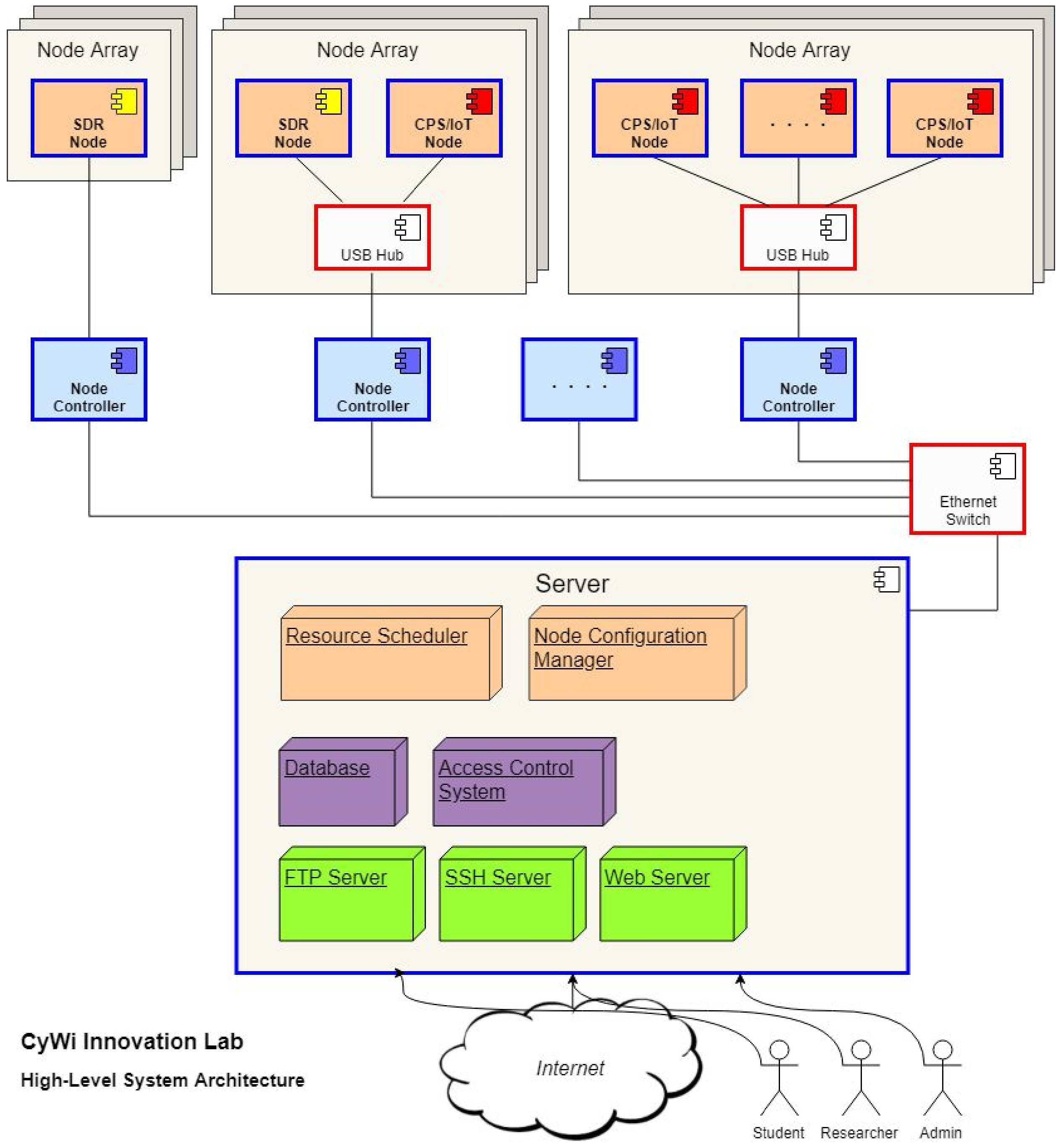


Figure 3: High Level Block Diagram

Our testbed design features two separate sets of wireless devices: software-defined radio (SDR) nodes and CPS (cyber-physical system) motes. Both sets of nodes are mounted to a 11x10 grid of square ceiling tiles -- each tile measuring two feet in width and length. To facilitate remote configuring of the nodes, USB cables connect each node to one of two node controllers via intermediate USB hubs. The node controllers are then connected to a networking switch via Ethernet cables and ultimately to our Internet-connected server.

SDR nodes account for a majority of the project budget so at the moment we are limited to three SDRs which will be sufficient to run interesting 5G experiments. We have settled on the Ettus USRP B210 platform due to its flexibility, reliability, and compatibility with the OpenAirInterface framework. These SDRs are capable of generating, transmitting, and receiving signals at up to 6 GHz. OpenAirInterface is a framework that provides software and tools to research 5G radio access networks.

CPS motes are relatively inexpensive so we've purchased and mounted 60 Texas Instruments LAUNCHXL-CC26X2R1 development kits. These devices are capable of implementing multiple wireless protocols including Wi-Fi, Bluetooth Low Energy, and IEEE 802.15.4 standard protocols such as Thread, Zigbee, and Sub-1 GHz. These development kits support TI-RTOS (an open-source real time operating system under a BSD-like license) and implement the SimpleLink SDK platform which provides well-documented hardware drivers and stacks. During our search for suitable CPS motes, we compared this Texas Instruments development kit with others developed by Qualcomm, NXP, and Nordic but none of those were able to offer a powerful MCU, open documentation, and community support at the right price.

Node controllers are installed on the ceiling tile grid to provide access to our SDR and CPS motes. Node controllers are mini PCs running Linux/GNU that provide configuration and communication to the nodes. The SDR nodes, in particular, need powerful node controllers to implement their software radio components. For this task, we chose the Intel NUC8. We carefully analyzed a variety of NUC7 and NUC8 mini PCs as well as the cheaper yet much less powerful MintBox.

One server hosts open-source software designed to allow remote access, store data, and interact with the node controllers. After discussing with ETG what the best choice for a server machine would be, we decided on a powerful Dell workstation. This machine offered us a Xeon processor and 32GB of RAM, perfectly suited to handle multitasking. Additionally we added two 2TB enterprise hard drives to be set up in a RAID1 configuration. The RAID setup was a critical part of the overall robustness of the test bed. We planned on implementing the Emulab software platform for many of the testbed features but ultimately could not get it working in the allotted time of the project. More on this and other alternative design decisions can be found in Appendix II. Instead, the server now hosts Web, SSH, and FTP services. Anyone can view the website, but those wishing testbed access can request a username and gain login ability to the server. Once logged into the server, they can SSH to the node controllers and begin experiments. Each user has a home directory on the server where they can temporarily store any experiment results before FTPing them off site.

The CyWi Innovation Lab satisfies the following functional and non-functional requirements:

3.2 Functional Requirements

- Users have remote access to the lab
- Users have the ability to flash wireless devices
- Experiment output data can be exportable
- Radio attenuation is configurable

3.3 Non-Functional Requirements

- Software is open-source
- Only registered users have remote access
- Testbed availability is shown to the users
- System has backup redundancy

The CyWi Innovation Lab works in accordance with several standards including:

- Ethernet (IEEE 802.3)
- Wi-Fi (IEEE 802.11)
- Bluetooth (IEEE 802.15)
- Thread, Zigbee, and Sub-1 GHz (IEEE 802.15.4)

3.4 Design Analysis

At the start of the project, our goal was to have 110 CPS motes and 20 SDRs but the initial funding was not as rich as we had hoped. We had to scale back to 60 CPS motes and 3 SDRs for the time being. More nodes can always be added as time goes by. Since we currently have at least one of each unit, adding more quantity of each device (additional 60 CPS motes and 17 SDRs) is simply duplicating what we currently have. Whether we have 20 or 110 motes, the server-side of the system will remain mostly the same. Every server component (such as SSH, FTP, Access Control, etc) will need to be implemented regardless of node quantity so project feasibility is established. In fact, it is easier to test each subsystem with only a couple devices at one time. We have sufficient nodes to provide a proof-of-concept for the project.

Our testbed hardware selections are based on our solid research spanning many vendors and products. For our CPS motes, we selected the Texas Instruments LAUNCHXL-CC26X2R1 development kits. This TI development board supports open-source software and offers excellent documentation, both of which will allow our testbed to be more accessible. The next piece of hardware that we picked was the node controller. Our node controllers are mini PCs, specifically Intel NUC8s. Size is a factor because these PCs need to be mounted above the ceiling tile, but the PCs also need to be powerful enough to run several wireless devices at a time. SDRs are resource-demanding. The decision to go with the NUC8i7BEH was based primarily on its high performance but this was balanced with its price. After determining that the tasks these node controllers will be performing were going to be more CPU than GPU intensive, we placed more weight on the cost-to-performance ratio of the CPU rather than the GPU. We compared this CPU ratio across six different Intel NUC models and concluded that the NUC8i7BEH was the best choice for our application.

Overall, we are pleased with our hardware choices. We feel that a major strength of our decision process has been selecting our node controllers. The model we decided on after conducting research is both significantly cheaper and higher in performance than the NUC our client had originally suggested. One weakness of the project has been that we underestimated just how much time would be spent researching.

4. Implementation

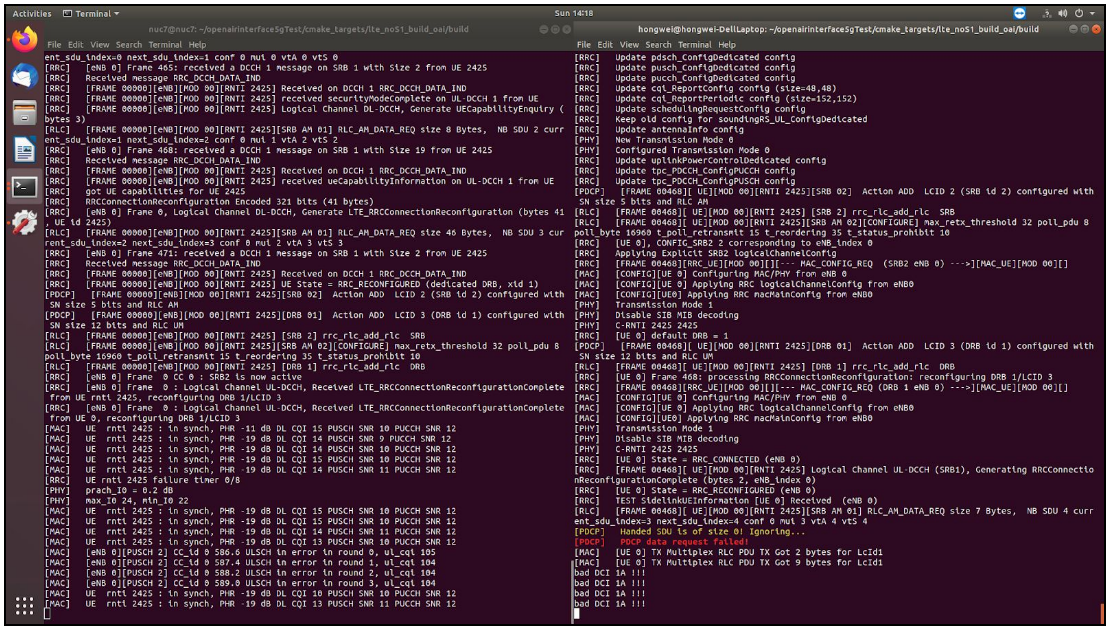
4.1 Local Area Network

The CyWi testbed is located at 3038 Coover Hall and is on the Iowa State network. We have installed a border router to segregate our traffic from Iowa State and a smart switch to provide us with more ports for devices. This means that to reach our private 192.168.1.0/24 network, users must first be on Iowa State's private 10.29.0.0/16 network.

The router has port forwarding configured to forward all HTTP, SSH, and FTP traffic straight to the CyWi server. The node controllers only have 192.168.1.0/24 IPs so they are only reachable through SSH via the CyWi server. This lets us simplify by keeping the user profiles entirely on the server while the node controllers can have generic user accounts. All our internal IPs are static for stability during and after node controller firmware/software flashes.

4.2 OpenAirInterface (OAI)

We have installed OAI on the SDR node controllers which run on Linux OS. In order to install OAI, we have to install Ubuntu 16.04.6 along with the 4.15 low latency kernel. After going through the process of setting up OAI, we have decided to make an installation guide for students who continue on our work, which can be found in appendix.



```
nuc7@nuc7: ~/openairinterface5gTest/cmake_targets/lte_noS1_build_oai/build
ent_sdu_index=0 next_sdu_index=1 conf 0 mui 0 vtA 0 vtS 0
[RRC]   [eNB 0] Frame 465: received a DCCH 1 message on SRB 1 with Size 2 from UE 2425
[RRC]   Received message RRC_DCCH_DATA_IND
[RRC]   [FRAME 00000][eNB][MOD 00][RNTI 2425] Received on DCCH 1 RRC_DCCH_DATA_IND
[RRC]   [FRAME 00000][eNB][MOD 00][RNTI 2425] received securityModeComplete on UL-DCCH 1 from UE
[RRC]   [FRAME 00000][eNB][MOD 00][RNTI 2425] Logical Channel DL-DCCH, Generate UECapabilityEnquiry (
bytes 3)
[RLC]   [FRAME 00000][eNB][MOD 00][RNTI 2425][SRB AM 01] RLC_AM_DATA_REQ size 8 Bytes,  NB SDU 2 curr
ent_sdu_index=1 next_sdu_index=2 conf 0 mui 1 vtA 2 vtS 2
[RRC]   [eNB 0] Frame 468: received a DCCH 1 message on SRB 1 with Size 19 from UE 2425
[RRC]   Received message RRC_DCCH_DATA_IND
[RRC]   [FRAME 00000][eNB][MOD 00][RNTI 2425] Received on DCCH 1 RRC_DCCH_DATA_IND
[RRC]   [FRAME 00000][eNB][MOD 00][RNTI 2425] received ueCapabilityInformation on UL-DCCH 1 from UE
[RRC]   got UE capabilities for UE 2425
[RRC]   RRCConnectionReconfiguration Encoded 321 bits (41 bytes)
[RRC]   [eNB 0] Frame 0, Logical Channel DL-DCCH, Generate LTE_RRCConnectionReconfiguration (bytes 41
, UE id 2425)
[RLC]   [FRAME 00000][eNB][MOD 00][RNTI 2425][SRB AM 01] RLC_AM_DATA_REQ size 46 Bytes,  NB SDU 3 cur
rent_sdu_index=2 next_sdu_index=3 conf 0 mui 2 vtA 3 vtS 3
[RRC]   [eNB 0] Frame 471: received a DCCH 1 message on SRB 1 with Size 2 from UE 2425
[RRC]   Received message RRC_DCCH_DATA_IND
[RRC]   [FRAME 00000][eNB][MOD 00][RNTI 2425] Received on DCCH 1 RRC_DCCH_DATA_IND
[RRC]   [FRAME 00000][eNB][MOD 00][RNTI 2425] UE State = RRC_RECONFIGURED (dedicated DRB, xid 1)
[PDCP]   [FRAME 00000][eNB][MOD 00][RNTI 2425][SRB 02]  Action ADD  LCID 2 (SRB id 2) configured with
 SN size 5 bits and RLC AM
[PDCP]   [FRAME 00000][eNB][MOD 00][RNTI 2425][DRB 01]  Action ADD  LCID 3 (DRB id 1) configured with
 SN size 12 bits and RLC UM
[RLC]   [FRAME 00000][eNB][MOD 00][RNTI 2425] [SRB 2] rrc_rlc_add_rlc  SRB
[RLC]   [FRAME 00000][eNB][MOD 00][RNTI 2425][SRB AM 02][CONFIGURE] max_retx_threshold 32 poll_pdu 8
poll_byte 16960 t_poll_retransmit 15 t_reordering 35 t_status_prohibit 10
[RLC]   [FRAME 00000][eNB][MOD 00][RNTI 2425] [DRB 1] rrc_rlc_add_rlc  DRB
[RRC]   [eNB 0] Frame  0 CC 0 : SRB2 is now active
[RRC]   [eNB 0] Frame  0 : Logical Channel UL-DCCH, Received LTE_RRCConnectionReconfigurationComplete
 from UE rnti 2425, reconfiguring DRB 1/LCID 3
[RRC]   [eNB 0] Frame  0 : Logical Channel UL-DCCH, Received LTE_RRCConnectionReconfigurationComplete
 from UE 0, reconfiguring DRB 1/LCID 3
[MAC]   UE  rnti 2425 : in synch, PHR -11 dB DL CQI 15 PUSCH SNR 10 PUCCH SNR 12
[MAC]   UE  rnti 2425 : in synch, PHR -19 dB DL CQI 14 PUSCH SNR 9 PUCCH SNR 12
[MAC]   UE  rnti 2425 : in synch, PHR -19 dB DL CQI 14 PUSCH SNR 10 PUCCH SNR 12
[MAC]   UE  rnti 2425 : in synch, PHR -19 dB DL CQI 15 PUSCH SNR 10 PUCCH SNR 12
[MAC]   UE  rnti 2425 : in synch, PHR -19 dB DL CQI 14 PUSCH SNR 11 PUCCH SNR 12
[RRC]   UE rnti 2425 failure timer 0/8
[PHY]   prach_I0 = 0.2 dB
[PHY]   max_I0 24, min_I0 22
[MAC]   UE  rnti 2425 : in synch, PHR -19 dB DL CQI 15 PUSCH SNR 10 PUCCH SNR 12
[MAC]   UE  rnti 2425 : in synch, PHR -19 dB DL CQI 15 PUSCH SNR 10 PUCCH SNR 12
[MAC]   UE  rnti 2425 : in synch, PHR -19 dB DL CQI 14 PUSCH SNR 11 PUCCH SNR 12
[MAC]   UE  rnti 2425 : in synch, PHR -19 dB DL CQI 13 PUSCH SNR 10 PUCCH SNR 12
[MAC]   [eNB 0][PUSCH 2] CC_id 0 586.6 ULSCH in error in round 0, ul_cqi 105
[MAC]   [eNB 0][PUSCH 2] CC_id 0 587.4 ULSCH in error in round 1, ul_cqi 104
[MAC]   [eNB 0][PUSCH 2] CC_id 0 588.2 ULSCH in error in round 2, ul_cqi 104
[MAC]   [eNB 0][PUSCH 2] CC_id 0 589.0 ULSCH in error in round 3, ul_cqi 104
[MAC]   UE  rnti 2425 : in synch, PHR -19 dB DL CQI 10 PUSCH SNR 10 PUCCH SNR 12
[MAC]   UE  rnti 2425 : in synch, PHR -19 dB DL CQI 13 PUSCH SNR 11 PUCCH SNR 12
```

```
hongwei@hongwei-DellLaptop: ~/openairinterface5gTest/cmake_targets/lte_noS1_build_oai/build
[RRC]   Update pdsch_ConfigDedicated config
[RRC]   Update pusch_ConfigDedicated config
[RRC]   Update pucch_ConfigDedicated config
[RRC]   Update cqi_ReportConfig config (size=48,48)
[RRC]   Update cqi_ReportPeriodic config (size=152,152)
[RRC]   Update schedulingRequestConfig config
[RRC]   Keep old config for soundingRS_UL_ConfigDedicated
[RRC]   Update antennaInfo config
[PHY]   New Transmission Mode 0
[PHY]   Configured Transmission Mode 0
[RRC]   Update uplinkPowerControlDedicated config
[RRC]   Update tpc_PDCCH_ConfigPUCCH config
[RRC]   Update tpc_PDCCH_ConfigPUSCH config
[PDCP]   [FRAME 00468][ UE][MOD 00][RNTI 2425][SRB 02]  Action ADD  LCID 2 (SRB id 2) configured with
 SN size 5 bits and RLC AM
[RLC]   [FRAME 00468][ UE][MOD 00][RNTI 2425] [SRB 2] rrc_rlc_add_rlc  SRB
[RLC]   [FRAME 00468][ UE][MOD 00][RNTI 2425][SRB AM 02][CONFIGURE] max_retx_threshold 32 poll_pdu 8
poll_byte 16960 t_poll_retransmit 15 t_reordering 35 t_status_prohibit 10
[RRC]   [UE 0], CONFIG_SRB2 2 corresponding to eNB_index 0
[RRC]   Applying Explicit SRB2 logicalChannelConfig
[RRC]   [FRAME 00468][RRC_UE][MOD 00][][--- MAC_CONFIG_REQ  (SRB2 eNB 0) --->][MAC_UE][MOD 00][]
[MAC]   [CONFIG][UE 0] Configuring MAC/PHY from eNB 0
[MAC]   [CONFIG][UE 0] Applying RRC logicalChannelConfig from eNB0
[MAC]   [CONFIG][UE0] Applying RRC macMainConfig from eNB0
[PHY]   Transmission Mode 1
[PHY]   Disable SIB MIB decoding
[PHY]   C-RNTI 2425 2425
[RRC]   [UE 0] default DRB = 1
[PDCP]   [FRAME 00468][ UE][MOD 00][RNTI 2425][DRB 01]  Action ADD  LCID 3 (DRB id 1) configured with
 SN size 12 bits and RLC UM
[RLC]   [FRAME 00468][ UE][MOD 00][RNTI 2425] [DRB 1] rrc_rlc_add_rlc  DRB
[RRC]   [UE 0] Frame 468: processing RRCConnectionReconfiguration: reconfiguring DRB 1/LCID 3
[RRC]   [FRAME 00468][RRC_UE][MOD 00][][--- MAC_CONFIG_REQ (DRB 1 eNB 0) --->][MAC_UE][MOD 00][]
[MAC]   [CONFIG][UE 0] Configuring MAC/PHY from eNB 0
[MAC]   [CONFIG][UE 0] Applying RRC logicalChannelConfig from eNB0
[MAC]   [CONFIG][UE0] Applying RRC macMainConfig from eNB0
[PHY]   Transmission Mode 1
[PHY]   Disable SIB MIB decoding
[PHY]   C-RNTI 2425 2425
[RRC]   [UE 0] State = RRC_CONNECTED (eNB 0)
[RRC]   [FRAME 00468][ UE][MOD 00][RNTI 2425] Logical Channel UL-DCCH (SRB1), Generating RRCConnectio
nReconfigurationComplete (bytes 2, eNB_index 0)
[RRC]   [UE 0] State = RRC_RECONFIGURED (eNB 0)
[RRC]   TEST SidelinkUEInformation [UE 0] Received  (eNB 0)
[RLC]   [FRAME 00468][ UE][MOD 00][RNTI 2425][SRB AM 01] RLC_AM_DATA_REQ size 7 Bytes,  NB SDU 4 curr
ent_sdu_index=3 next_sdu_index=4 conf 0 mui 3 vtA 4 vtS 4
[PDCP]   Handed SDU is of size 0! Ignoring...
[PDCP]   PDCP data request failed!
[MAC]   [UE 0] TX Multiplex RLC PDU TX Got 2 bytes for LcId1
[MAC]   [UE 0] TX Multiplex RLC PDU TX Got 9 bytes for LcId1
bad DCI 1A !!!
bad DCI 1A !!!
bad DCI 1A !!!
bad DCI 1A !!!
```

Figure 4: Connection between eNB and UE

4.3 Interface Specifications

To fully test our system, we have to look at each component to determine how it interfaces with the others. We can begin by defining the interfaces between our four types of physical hardware: CPS motes, SDR nodes, Node Controllers, and server. Then we can define the interfaces of our various services: web server, SSH server, FTP server, access control system.

The physical connections are as follows. The server connects to an Ethernet switch in the lab room with Cat5 cable. Each of the Node Controllers also have an Ethernet connection to the switch. All the Node Controllers and the server will be on the same VLAN as we have no need for more complicated switching or routing. Up to twenty CPS motes will connect to one Node Controller via USB cables and a USB hub.

Since the CPS motes run with an operating system and are able to communicate wirelessly, they are not as reliant on host PCs as the SDR nodes are. In fact, the Node Controller will be transparent to the user in regards to using CPS motes. Configurations are completely up to the experiment owner but could include initial RF signal power, which radios to initialize, which TI software stacks to enable, etc. If live access to individual CPS motes is required, the network session will flow from server, to Node Controller, to CPS mote. Testing CPS motes will involve trying the following functionalities: live access to the mote, and transmit and receive test data.

SDRs need a host PC to function so each SDR will be paired with a Node Controller. OpenAirInterface will be the software platform we will support for mobile network experimentation and the SDRs will function as cellular base stations. Testing the SDR functions will involve logging into the Node Controller via SSH to gain access to the Linux operating system. From there, we will run OpenAirInterface and run several of its core functions which will include programming the SDR's FPGA to send and receive multiple different types of RF signals and receiving them on another SDR. If both SDRs can transmit and receive across multiple frequencies, we can be sure that the user experience will be a good one. Finally, we must gather experiment logs (such as traffic performance) and send them back to the user via web server.

4.4 Hardware and Software

Hardware used for testing:

- Texas Instrument CC26X2R1 Launchpad
- Ettus Research USRP B210 SDR
- Ettus Research Vert2450 Antenna
- Intel NUC8

Software used for testing:

- Performance analysis tool built into Chrome
- SimpleLink testing software
- OpenAirInterface (OAI) testing tools
- Code Composer Studio.
- TI-RTOS

5. Testing

5.1 CPS Motes

Attenuation testing

For this test bed to work correctly, the motes need to only communicate with their neighbors when running on minimum power. They also must communicate across the whole room when running on maximum power. We needed to test certain attenuation values on the motes to make sure that they can not reach too far running on minimum power. This testing was done by sending 1000 packets at minimum power from different distances starting from 1 foot and increasing the distance by 1 foot every time. We continued this until packets were no longer being received. The last test that needed to be done was to test if the motes could communicate across the room at maximum power.

Result

We found that the optimal attenuation for all of the motes was 20dbm. This allowed us to have an average packet loss of 5-10% within the range of 2-4 feet and the packet loss would spike up to around 50% when the range was increased to 5 feet. Packets were no longer being received after a distance of 8 feet. Packets were received with 1% packet loss at maximum power across the whole room.

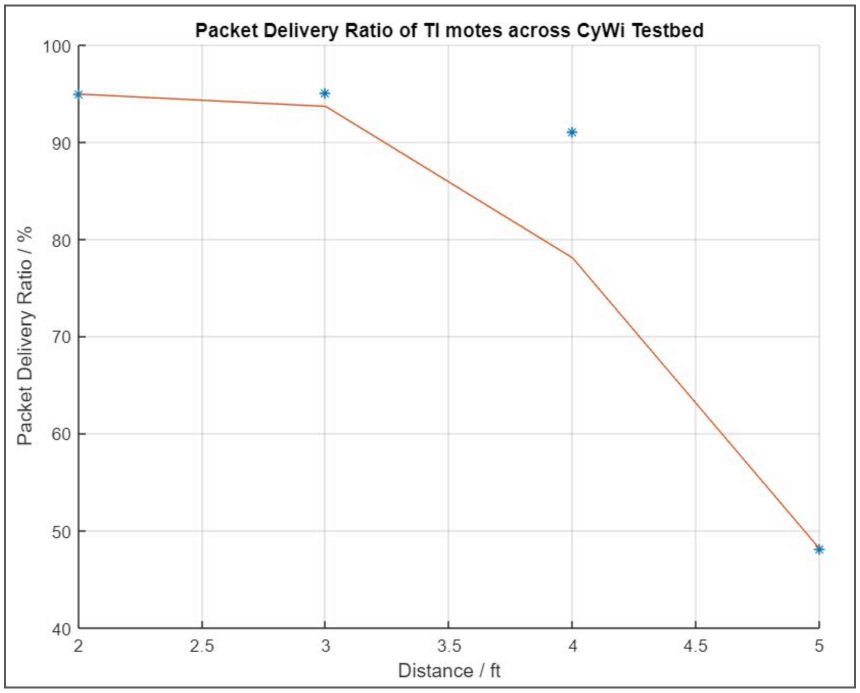


Figure 5: TI Motes at Minimum Power

5.2 SDR Nodes

Attenuation

Before we attempt to determine the right attenuation, we first find the configuration that works with USRP B210 the best under stable condition. Configuration and measurement values were empirical, further explanations about the reason behind it is likely to be related to radio's FPGA which is beyond our level. As soon as the configuration values were optimized, we then further tested different attenuations (i.e. from 6dB to 30dB) to find out an optimal effective gain value that can realize multihop networks in the lab.

OpenAirInterface (OAI)

In order to test our SDR, we use OAI to mimic cellular devices communication in an enclosed environment. OAI allowed us to emulate standard compliant network functions to establish LTE connection between a base station and a user equipment. After that, we used large sample ping tests (e.g. 500 icmp packets) to determine packet delivery ratio via the wireless connection.

Result

The optimal attenuation we should add to a point-to-point wireless connection is 18dB which results in 9dB attenuator at each radio front end. With 9dB attenuator connected between each B210 RX/TX output and VERT2450 3dBi antenna, we managed to model large scale wireless performance at the minimum power under stable operation as shown in the graph below. Note that we do have the flexibility to scale down the network size by tuning up the transmission power, which gave packet delivery ratio of 90% and above at any point in the lab (i.e. single hop).

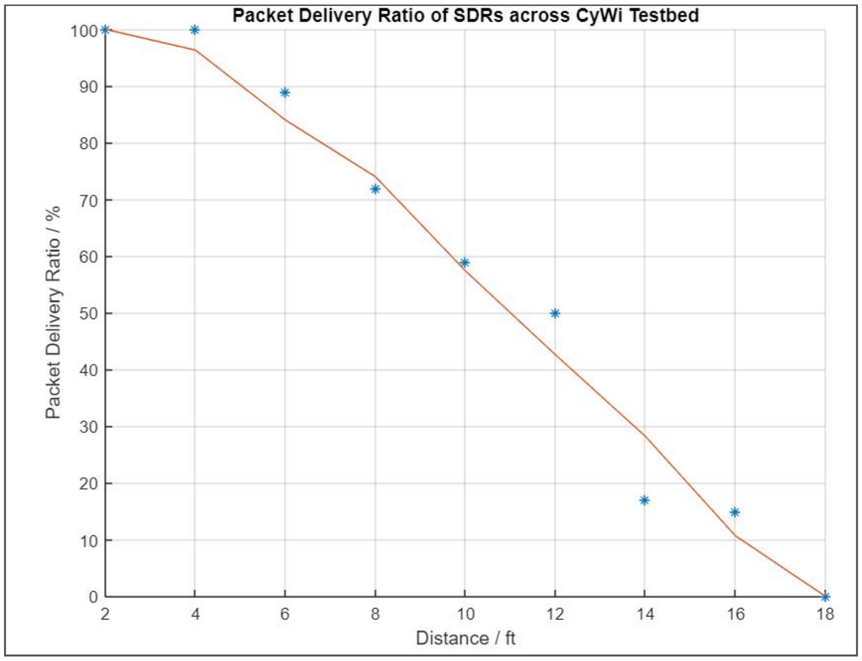
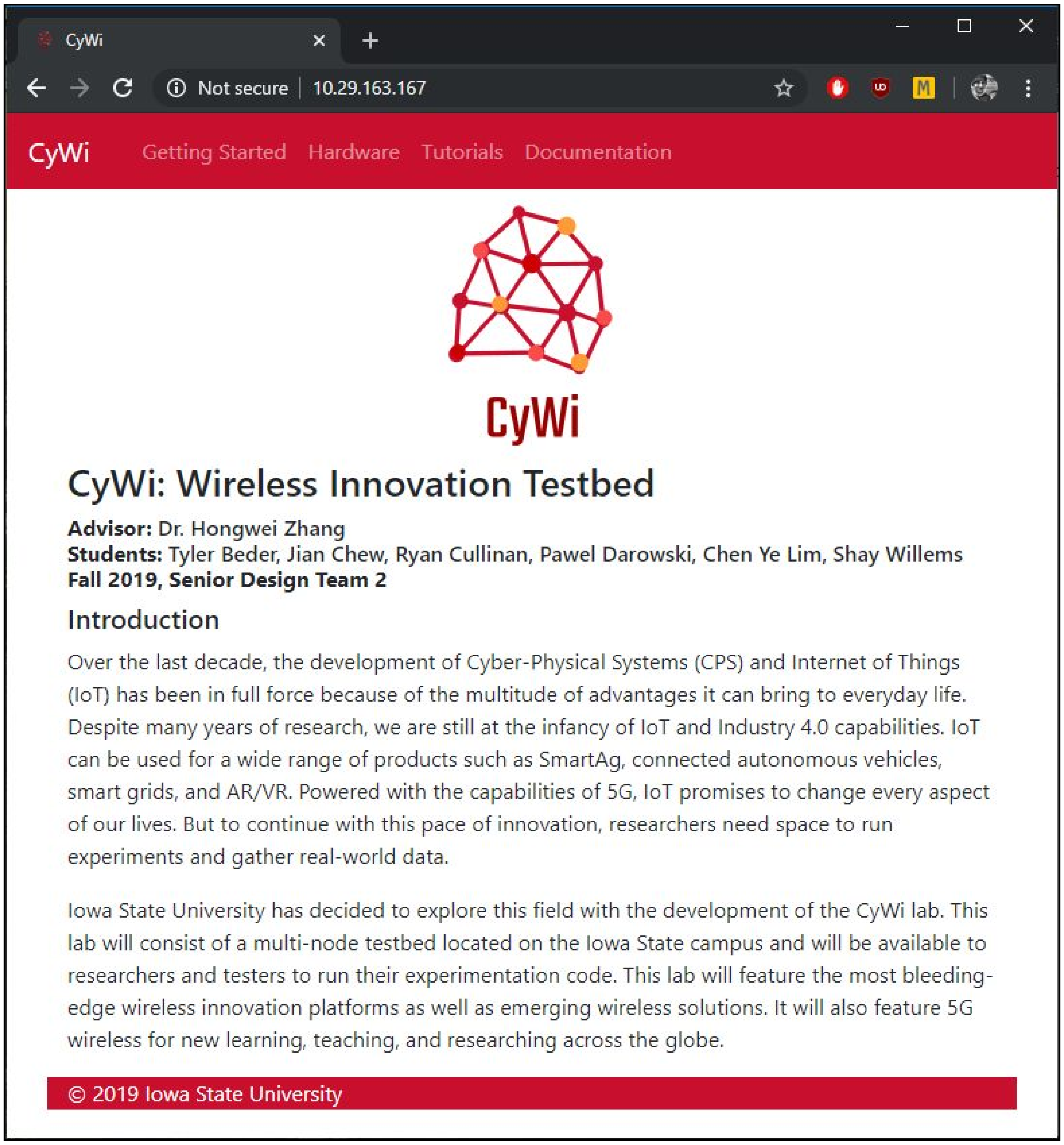


Figure 6: SDR at Minimum Power

5.3 Server

Website

The server runs Apache and hosts its own website for CyWi details such as getting started instructions, hardware available, tutorials, and other documentation. Testing the web site simply involved using the browser to view the website from on campus as well as from the Iowa State VPN. It is available and responsive.



CyWi: Wireless Innovation Testbed

Advisor: Dr. Hongwei Zhang
Students: Tyler Beder, Jian Chew, Ryan Cullinan, Pawel Darowski, Chen Ye Lim, Shay Willems
Fall 2019, Senior Design Team 2

Introduction

Over the last decade, the development of Cyber-Physical Systems (CPS) and Internet of Things (IoT) has been in full force because of the multitude of advantages it can bring to everyday life. Despite many years of research, we are still at the infancy of IoT and Industry 4.0 capabilities. IoT can be used for a wide range of products such as SmartAg, connected autonomous vehicles, smart grids, and AR/VR. Powered with the capabilities of 5G, IoT promises to change every aspect of our lives. But to continue with this pace of innovation, researchers need space to run experiments and gather real-world data.

Iowa State University has decided to explore this field with the development of the CyWi lab. This lab will consist of a multi-node testbed located on the Iowa State campus and will be available to researchers and testers to run their experimentation code. This lab will feature the most bleeding-edge wireless innovation platforms as well as emerging wireless solutions. It will also feature 5G wireless for new learning, teaching, and researching across the globe.

© 2019 Iowa State University

Figure 7: CyWi Testbed Website

SSH Access

The server also provides access via SSH from within the Iowa State campus or VPN. To test that it works, we simply logged in and were able to work. SSH is how we configured a lot of the project so this test was ingrained with the rest of development. Nevertheless, we tested that other users were working as well, as the figure below shows.

```
cywi@cywi-server:~$ ssh localhost -l student
student@localhost's password:
```

```
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 5.0.0-37-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

106 packages can be updated.
0 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Tue Dec 10 16:10:56 2019 from 10.26.40.41
student@cywi-server:~$ pwd
/home/student
student@cywi-server:~$ ls
examples.desktop
student@cywi-server:~$
```

Figure 8: SSH to the CyWi Server with Multiple Users

Node controllers are accessed via SSH through the CyWi server. The figure below shows that a user logged into the cywi-server can easily SSH into a neighboring node controller.

```
cywi@cywi-server:~$ ssh cywi-nc-sdr01 -l nuc7
nuc7@cywi-nc-sdr01's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-46-lowlatency x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

58 packages can be updated.
10 updates are security updates.

New release '18.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed Dec 11 05:01:07 2019 from 192.168.1.39
nuc7@cywi-nc-sdr01:~$
```

Figure 9: SSH to a Node Controller

FTP Transfer

Finally, the CyWi server also hosts an FTP server to transfer test results and documents. To test that the FTP server has been setup correctly, we used FileZilla on a personal laptop logged into the Iowa State network via VPN to login to CyWi and download a test file from CyWi. The download had no errors and completed quickly.

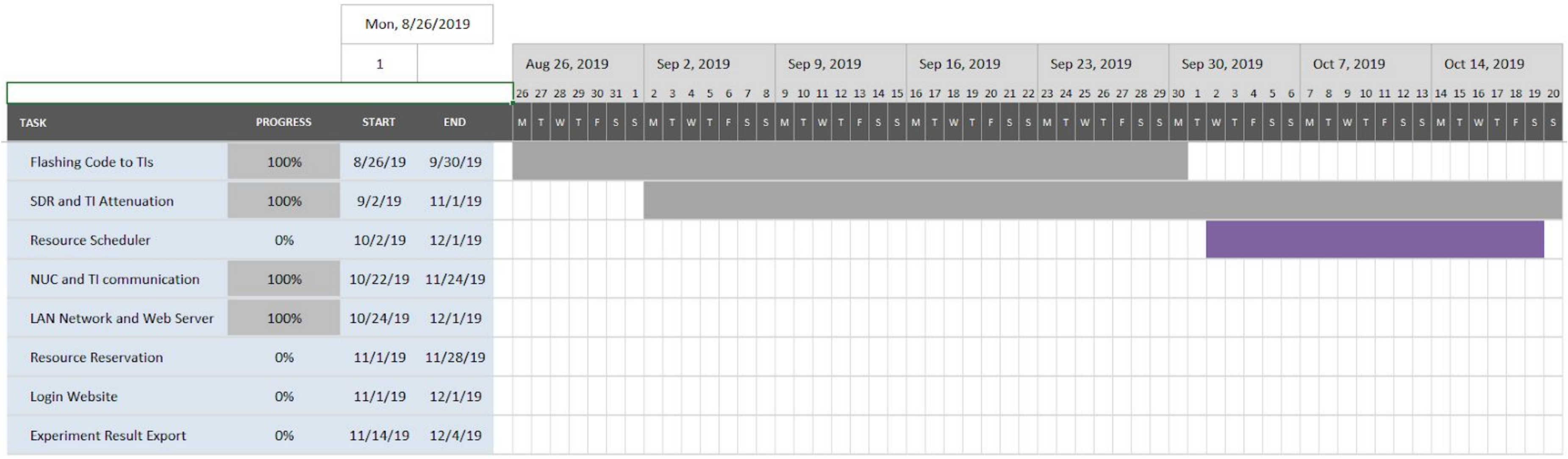
6. Market Survey

We researched several existing wireless testbeds in our project’s initial stages to define what a testbed is and how its elements must perform together to comprise a system. One such platform is Powder located in Salt Lake City, Utah. Powder’s goal is to provide a wireless testing environment that spans city-wide areas including a downtown area, a residential neighborhood, and the University of Utah campus. Researchers are able to build mobile wireless networks using 4G and MIMO technologies. One Powder testbed cluster is built using the well-known, open-source Emulab environment.

Orbit is another wireless testbed we researched. It is managed by several universities in the New York and New Jersey region and has been around since 2005. The project hosts multiple testbeds of various sizes from 2x2 radio node grids up to a 20x20 grid. The CyWi lab will start out with a maximum of 11x10 nodes. Both Powder and Orbit have the same experiment lifecycle including specifying the architecture, parsing the specs to a server, allocating resources, configuring the nodes, and gathering logs. The CyWi lab will have the same general functions. However, our focus will be on providing a platform specifically for IoT and 5G experimentation.

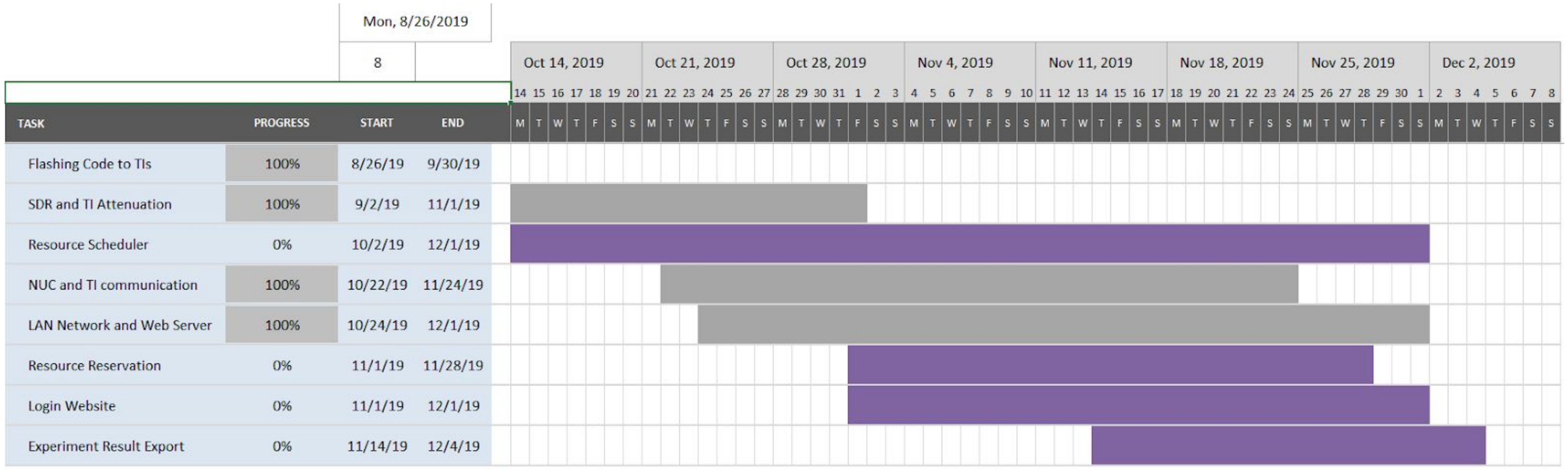
7. Project Tracking Procedure

The following Gantt chart shows the development schedule that we tried to follow during the second semester of the project.


Mon, 8/26/2019

1

TASK	PROGRESS	START	END
Flashing Code to TIs	100%	8/26/19	9/30/19
SDR and TI Attenuation	100%	9/2/19	11/1/19
Resource Scheduler	0%	10/2/19	12/1/19
NUC and TI communication	100%	10/22/19	11/24/19
LAN Network and Web Server	100%	10/24/19	12/1/19
Resource Reservation	0%	11/1/19	11/28/19
Login Website	0%	11/1/19	12/1/19
Experiment Result Export	0%	11/14/19	12/4/19


Mon, 8/26/2019

8

TASK	PROGRESS	START	END
Flashing Code to TIs	100%	8/26/19	9/30/19
SDR and TI Attenuation	100%	9/2/19	11/1/19
Resource Scheduler	0%	10/2/19	12/1/19
NUC and TI communication	100%	10/22/19	11/24/19
LAN Network and Web Server	100%	10/24/19	12/1/19
Resource Reservation	0%	11/1/19	11/28/19
Login Website	0%	11/1/19	12/1/19
Experiment Result Export	0%	11/14/19	12/4/19

8. Closure Material

8.1 Conclusion

Technological innovation has been steadily increasing over the last few decades. The future is fast approaching and Cyber-Physical Systems, the Internet of Things, and 5G wireless will soon be mainstream. These highly disruptive technologies will change every facet of modern life including healthcare, home safety, industrial automation, transportation, education, social connectivity, entertainment, and more.

Such a social seachange cannot be made possible without the continued efforts of researchers. The Iowa State University CyWi innovation lab aims to provide students and researchers the resources to learn about emerging technologies and to push the boundaries of possibility forward. Registered users will have a well-documented matrix of SDR and CPT/IoT nodes at their disposal with the ability to run custom experiments across several wireless protocols. Exportable experiment results will allow researchers to take their data offline for further analysis. Innovating in such a lab environment will speed up the next technological revolution.

8.2 References

[1] https://powderwireless.net
[2] https://www.orbit-lab.org
[3] https://www.intel.com/content/www/us/en/products/boards-kits/nuc/kits/nuc8i7beh.html
[4] https://www.ettus.com/product/details/UB210-KIT
[5] http://www.ti.com/tool/launchxl-cc26x2r1
[6] https://www.openairinterface.org
[7] http://www.emulab.net
[8]https://kb.ettus.com/Building_and_Installing_the_USRP_Open-Source_Toolchain_(UHD_and_GNU_Radio)_on_Linux

Appendix I: Operating Manual

Connecting to the Server

To see the CyWi website, go to http://10.29.163.167 in your browser while on the Iowa State network. Instructions there tell you who to email to request a username.

Once you have a user/pass, use a SSH client (such as Putty or similar program) to login to cywi-server at 10.29.163.167 on the standard SSH port 22.

After you are connected to cywi-server, you may SSH into a neighboring node controller by simply doing the following. Instead of searching for a list of hostnames or IPs, type `ssh cywi<TAB>` to populate the list of neighboring node controllers. Each one has been added to the cywi-server hosts file.

```
cywi@cywi-server:~$ ssh cywi-nc-sdr01 -l student
student@cywi-nc-sdr01's password:

student@cywi-nc-sdr01~$
```

For FTP access to CyWi, use an FTP client (such as Filezilla) while on the Iowa State network to login to 10.29.163.167 on the standard FTP port 21. From there, you can browse through the folders and download what you need.

Installation Guide on OAI

1. *Install the right linux image & headers for OAI*
   - ~ sudo apt-get install linux-image-4.15.0-50-lowlatency
   - ~ sudo apt-get install linux-headers-4.15.0-50-lowlatency
   - ~ dpkg -l | grep linux-image
   - ~ sudo apt-get remove --purge *<any other linux image in list>*
   - ~ dpkg -l | grep linux-headers
   - ~ sudo apt-get remove --purge *<any other linux headers in list>*
   - ~ sudo update-grub
   - ~ sudo reboot --r now
2. *Install UHD driver to support USRP SDR*
   - ~ sudo apt-get update
   - ~ sudo apt-get -y install git swig cmake doxygen build-essential libboost-all-dev libtool libusb-1.0-0 libusb-1.0-0-dev libudev-dev libncurses5-dev libfftw3-bin

libfftw3-dev libfftw3-doc libcppunit-1.13-0v5 libcppunit-dev libcppunit-doc ncurses-bin cpufrequtils python-numpy python-numpy-doc python-numpy-dbg python-scipy python-docutils qt4-bin-dbg qt4-default qt4-doc libqt4-dev libqt4-dev-bin python-qt4 python-qt4-dbg python-qt4-dev python-qt4-doc python-qt4-doc libqwt6abi1 libfftw3-bin libfftw3-dev libfftw3-doc ncurses-bin libncurses5 libncurses5-dev libncurses5-dbg libfontconfig1-dev libxrender-dev libpulse-dev swig g++ automake autoconf libtool python-dev libfftw3-dev libcppunit-dev libboost-all-dev libusb-dev libusb-1.0-0-dev fort77 libsdl1.2-dev python-wxgtk3.0 git-core libqt4-dev python-numpy ccache python-opengl libgsl-dev python-cheetah python-mako python-lxml doxygen qt4-default qt4-dev-tools libusb-1.0-0-dev libqwt5-qt4-dev libqwtplot3d-qt4-dev pyqt4-dev-tools python-qwt5-qt4 cmake git-core wget libxi-dev gtk2-engines-pixbuf r-base-dev python-tk liborc-0.4-0 liborc-0.4-dev libasound2-dev python-gtk2 libzmq-dev libzmq1 python-requests python-sphinx libcomedi-dev python-zmq python-setuptools

- ~ cd $HOME
- ~ mkdir workarea
- ~ cd workarea
- ~ git clone https://github.com/EttusResearch/uhd
- ~ cd uhd
- ~ git checkout release_003_010_300_000
- ~ cd host
- ~ mkdir build
- ~ cd build
- ~ cmake ../
- ~ make
- ~ make test
- ~ sudo make install
- ~ sudo ldconfig
- ~ sudo vim ~/.bashrc
- *Add the following lines to the end of the code*
    i. export LD_LIBRARY_PATH=/usr/local/lib
       sudo uhd_images_downloader
- *Now go into BIOS and disable two settings:*
    i. Secure Boot
    ii. Hyperthreading

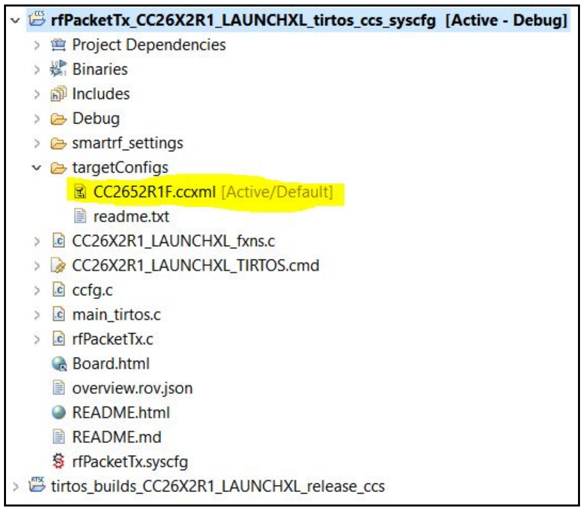
Running OpenAirInterface on the SDR Nodes

Jargons used in OAI: eNB == 4G base station; UE == user equipment (mobile phone)

1. *Go to the OAI folder directory*
   - cd openairinterface5gTest/
2. *Source OAI path for environment build up*
   - source oaienv
3. cd cmake_targets/
4. *Build dependencies and compile OAI c codes*
   sudo ./build_oai -w USRP --noS1 --eNB -c
   a. If you are configuring the SDR as a UE, replace "eNB" with "UE"
   b. To see more options, add -h for help
5. *Bring up oai IP for connection setup*
   a. source ../targets/bin/init_nas_nos1 eNB
   b. If you are configuring the SDR as a UE, replace "eNB" with "UE"
6. *Go to build directory*
   - cd lte_noS1_build_oai/build/
7. *To run the config file for eNB*
   - sudo -E ./lte-softmodem-nos1 -O ~/openairinterface5gTest/targets/PROJECTS/GENERIC-LTE-EPC/CONF/enb.tm1.25PRB.usrpb210.conf
8. *To run the config file for UE*
   - sudo -E ./lte-uesoftmodem-nos1 -O -U --ue-scan-carrier --ue-txgain *insert desired value* --ue-rxgain *insert desired value* -r 25 *insert desired frequency* --ue-max power *insert desired power*
   - Note: TX and RX gain are capped at 125 while the power is capped at 10

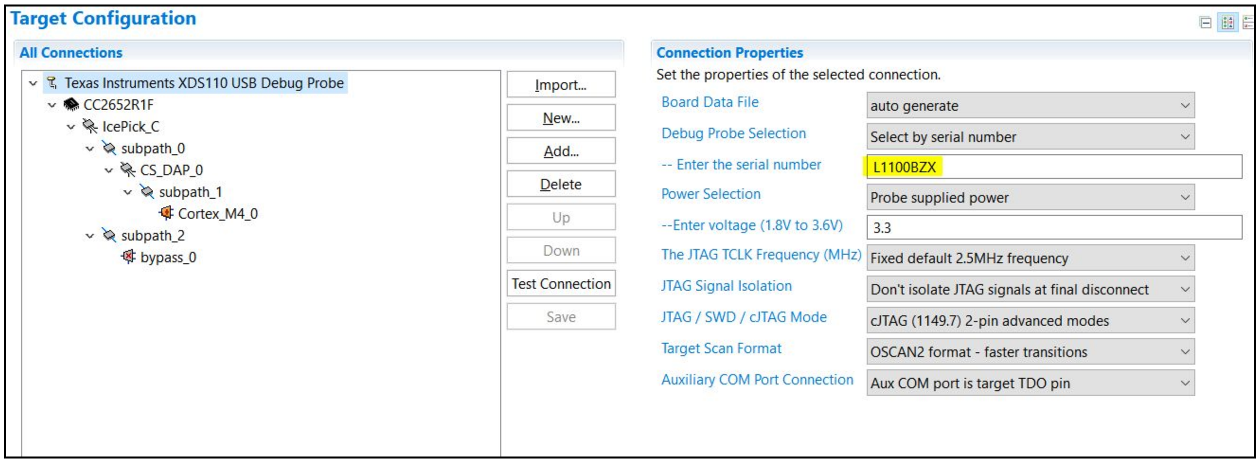
Configuring CPS nodes

1. Open Code Composer Studio on the NUC8 node controller
2. Set the Workspace to /home/Workspace1 and run
3. Click on rfPacketTx and click on the config file shown below



- rfPacketTx_CC26X2R1_LAUNCHXL_tirtos_ccs_syscfg [Active - Debug]
  - Project Dependencies
  - Binaries
  - Includes
  - Debug
  - smartrf_settings
  - targetConfigs
    - CC2652R1F.ccxml [Active/Default]
    - readme.txt
  - CC26X2R1_LAUNCHXL_fxns.c
  - CC26X2R1_LAUNCHXL_TIRTOS.cmd
  - ccfg.c
  - main_tirtos.c
  - rfPacketTx.c
  - Board.html
  - overview.rov.json
  - README.html
  - README.md
  - rfPacketTx.syscfg
- tirtos_builds_CC26X2R1_LAUNCHXL_release_ccs

4. Click on Target Configuration on the top right of the config file
5. Click on Texas Instruments XDS110 USB Debug Probe, you should now see a screen like below



Target Configuration

All Connections

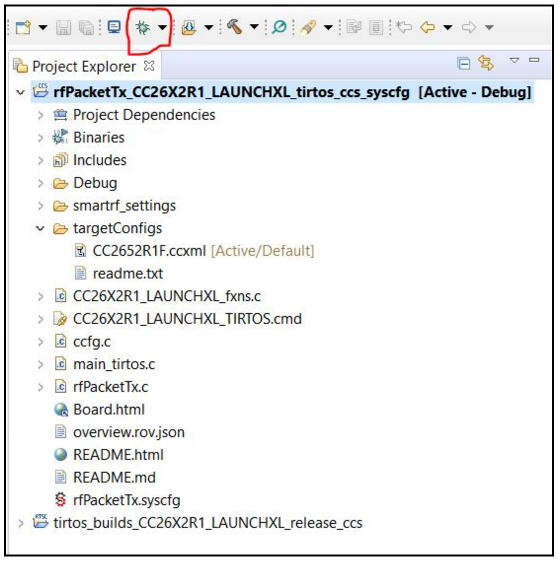
- Texas Instruments XDS110 USB Debug Probe
  - CC2652R1F
    - IcePick_C
      - subpath_0
        - CS_DAP_0
          - subpath_1
            - Cortex_M4_0
      - subpath_2
        - bypass_0

Connection Properties

Set the properties of the selected connection.

Board Data File	auto generate
Debug Probe Selection	Select by serial number
-- Enter the serial number	L1100BZX
Power Selection	Probe supplied power
--Enter voltage (1.8V to 3.6V)	3.3
The JTAG TCLK Frequency (MHz)	Fixed default 2.5MHz frequency
JTAG Signal Isolation	Don't isolate JTAG signals at final disconnect
JTAG / SWD / cJTAG Mode	cJTAG (1149.7) 2-pin advanced modes
Target Scan Format	OSCAN2 format - faster transitions
Auxiliary COM Port Connection	Aux COM port is target TDO pin

6. As shown in the previous picture, you can enter the serial number of the desire mote that you want to program
7. Click Debug at the top of code composer studio as shown below



rfPacketTx_CC26X2R1_LAUNCHXL_tirtos_ccs_syscfg [Active - Debug]
- Project Dependencies
- Binaries
- Includes
- Debug
- smartrf_settings
- targetConfigs
  - CC2652R1F.ccxml [Active/Default]
  - readme.txt
- CC26X2R1_LAUNCHXL_fxns.c
- CC26X2R1_LAUNCHXL_TIRTOS.cmd
- ccfg.c
- main_tirtos.c
- rfPacketTx.c
- Board.html
- overview.rov.json
- README.html
- README.md
- rfPacketTx.syscfg

tirtos_builds_CC26X2R1_LAUNCHXL_release_ccs

8. When it is done debugging, hit resume and you have now successfully flashed the code
9. Repeat steps 1-8 with rfPacketRx and insert a different serial number for step 6
10. You now have two motes communicating through TI-RTOS

Appendix II: Alternative Design

Emulab Testbed Software

During the design phase of the first semester, our plan was to write the resource scheduler and access control software ourselves. At the beginning of the second semester, our client advised us that testbed software already exists that would do most of that for us called Emulab. Other university testbeds such as Powder and PhantomNet have used Emulab and it is a great way to manage resources and experiment data.

Unfortunately for us, Emulab is very involved to set up. It requires two servers and a whole lot of configuring. We began the second semester by installing Xen hypervisor on our server. This brought its own challenges as none of us had any experience with Type-1 hypervisors. We had networking issues such as bridging between guest virtual machines that took up a lot of our time to troubleshoot. Eventually we did have two virtual machines running at the same time.

Another setback was that Emulab required FreeBSD rather than Linux. Again, none of us had any experience with FreeBSD and this brought new challenges. The package management system and filesystems were different than Linux. As time went by and troubleshooting got more and more complex, we found ourselves with just one or two weeks left.

At this point, we had to make a decision. One option was to risk trying to troubleshoot our Xen network bridging issues, get comfortable with FreeBSD and installing Emulab from source, and then configure Emulab which came with more less-than-perfect installation instructions. The other option was to drop Emulab and do what we could in-house so that we'd have a working, if limited in features, testbed.

We chose to stop work on Emulab, scrap Xen, and install Linux from scratch. With only about a week left before the project was due, we worked on installing and configuring all the necessary services on the server. We didn't have time to implement a resource scheduler or dynamic access control. These will be issues that the next senior design team will have to work out.