Intel Cloud Orchestration Networking

Winter Midterm Progress Report

Matthew Johnson, Cody Malick, and Garrett Smith

Team 51, Cloud Orchestra

Abstract

This document outlines the progress of the Cloud Orchestration Networking project over the Fall and Winter terms. It contains a short description of the project's purposes and goals, current progress, current issues, and any solutions to those issues. It also contains a week by week retrospective for all ten weeks of Fall term and the first half of Winter term.

Contents

I Project Goals 2

II Purpose 2

III Fall Progress 2

IV Winter Progress 3

IV-A Network Testing Progress - Garrett Smith . . . 3

IV-B Development Progress - Matthew Johnson . . . 4

IV-C Issues - Cody Malick . . . 6

IV-C1 Environment Setup - Cody Malick . . . 6

IV-C2 Network Testing - Garrett Smith . . . 8

V Fall Term Week by Week Reports 9

V-A Background . . . 9

V-B Weeks Zero Through Two . . . 10

V-C Week Three . . . 10

V-D Week Four . . . 10

V-E Week Five . . . 11

V-F Week Six . . . 11

V-G Week Seven . . . 11

V-H Week Eight . . . 11

V-I Week Nine . . . 11

V-J Week Ten . . . 12

VI Fall Term Retrospective 12

VII Winter Term Week by Week Reports 12

VII-A Winter Break and Week One . . . 12

VII-B Week Two . . . 13

VII-C Week Three . . . 13

VII-D Week Four . . . 13

VII-E Week Five . . . 14

VII-F Week Six . . . 14

VIII Winter Midterm Retrospective 14

References 15

I. Project Goals

Our project is to first switch the Linux-created GRE tunnel implementation in Ciao to use GRE tunnels created by Open vSwitch. From that point we will switch the actual tunneling implementation from GRE to VxLAN/nvGRE based on performance measurements of each on data center networking cards. After this is completed, a stretch goal is to replace Linux bridges with Open vSwitch switch instances.

These goals changed somewhat by the middle of the Winter term. The primary goal now is to replace the Linux bridges with Open vSwitch switch instances because of an assumption that was found to be incorrect. This change will be discussed in the Winter Progress section. This change also pushes the other two goals back behind the bridge goal.

II. Purpose

The current implementation of Ciao tightly integrates software defined networking principles to leverage a limited local awareness of just enough of the global cloud's state. Tenant overlay networks are used to overcome traditional hardware networking challenges by using a distributed, stateless, self-configuring network topology running over dedicated network software appliances. This design is achieved using Linux-native Global Routing Encapsulation (GRE) tunnels and Linux bridges, and scales well in an environment of a few hundred nodes.

While this initial network implementation in Ciao satisfies current simple networking needs in Ciao, all innovation around software defined networks has shifted to the Open vSwitch (OVS) framework. Moving Ciao to OVS will allow leverage of packet acceleration frameworks like the Data Plane Development Kit (DPDK) as well as provide support for multiple tunneling protocols such as VxLAN and nvGRE. VxLAN and nvGRE are equal cost multipath routing (ECMP) friendly, which could increase network performance overall.

III. Fall Progress

At present, the project is moving along smoothly. Our testing environment has been set up and is networked appropriately. Each Intel NUC (Next Unit of Computing) has Clear Linux installed. Come Winter term, we will get Ciao set up on each machine and begin development on Ciao. Software development on the project has yet to begin as we have just wrapped up the design phase.

Designing has been quite helpful in developing our understanding of the project, its goals, and purpose. Because this is a small component of a very complicated system, taking the time to investigate what a software defined network is, why it's being used, and why we are implementing the piece that we are has been quite beneficial.

While in the design phase, we found an extremely useful library for interfacing with Open vSwitch, libovsdb. Libovsdb is a library written in the Go programming language that allows for simple and efficient calls to the OVS Database Management Protocol. Interfacing with OVS is going to be a very large portion of the project for us, so finding the library is quite the boon. Here is a quick example of how this library functions:[1]

Listing 1. Example insert operation using libovsdb

```
// simple insert operation
insertOp := libovsdb.Operation{
   Op:  "insert",
   Table:  "Bridge",
   Row:    bridge,
   UUIDName: namedUUID,
}
```

The above example can be reused for all major operations in the OVS Database Management Protocol. Other example operations include select, delete, and update. Using just the operations listed here, we can accomplish most of the needed configuration changes within Open vSwitch.

IV. Winter Progress

Early Winter term was defined by various errors created first by the operating system we were using, and further compounded by networking issues from the MAC locked OSU network. We worked through these issues one-by-one and managed to make progress with development of the first goal. This progress was stymied by the discovery that our assumption about Open vSwitch compatibility was wrong about halfway through the term, which has changed the scope of our project considerably. While we originally planned to have that first goal completed by the end of the fifth week, this scope change altered our schedule and pushed it back. Good progress has been made on testing the basic cluster setup, but this has been hindered by the OSU network restrictions. All these topics will be covered in-depth in the following sections.

A. *Network Testing Progress - Garrett Smith*

To help handle the data from the network testing tools iperf and qperf Garrett wrote some scripts in Ruby to convert the output from multiple runs of iperf and qperf into CSV files. Ruby was chosen because it has good support for consuming plain text and JSON (author's opinion), and because Garrett is familiar with it. The plain text support is useful because qperf outputs its results in human readable format which is not useful when comparing the results of multiple tests. The JSON support is useful because we are using the iperf option to output results in JSON format, and again, we want to change the format the network performance data is in because the format is not great for using in performance comparisons.

We ran into more issues running the Ciao cluster on OSU's network so we moved it off of the OSU network. More details on those issues are in the Issues section. Garrett set up the Cisco SG300 switch Intel is lending us for this project in layer 3 mode. Running the switch in layer 3 mode allows us to use its more advanced features including its built in DHCP server. Using the built in DHCP server means we have control over how IP addresses are assigned, and we are not restricted by OSU's policies that require MAC addresses to be registered before a device can be connected. Garrett configured DHCP on the switch to reserve 5 IP addresses for the 5 Nucs, and use a pool of 75 dynamically assigned IP addresses for other devices that request an IP address on the network. In general this setup provides us with more control over the environment we are running the network testing in.

This setup will also be useful at expo. In one of the Senior Capstone meetings we were told we will not have access to OSU's wired network in the expo hall. If we want to demonstrate Ciao running on multiple machines, not just in single VM mode on one of our laptops we will need a setup similar to what Garrett set up. We should be able to move the setup Garrett has to our booth at the expo without any modifications because it is completely standalone. All we will require in the expo hall is power for the switch, 5 Nucs, and our laptops. The downsides to this setup are that we do not have remote access to the Nucs, and they cannot connect to the internet for updates. The Nucs are still registered on OSU's network so we can reconnect them if we need to access the internet on them. We can also SCP files from our laptops to the Nucs if we connect the laptops to the local network.

Currently four of the Nucs have static IP addresses on the local network Garrett set up, and he confirmed that other devices can be attached to the network and the DHCP server will assign them an IP address. One of the Nucs is being used as a development machine with Ciao running in single VM mode on it because Cody was having problems with nested KVM on his personal machine. That Nuc is still connected to the OSU network so we can remotely connect to it, and use it for development. There are more details on that in the issues section. We have not finished setting Ciao up on the new network yet. We need to clear out the configuration we had set for running Ciao on the OSU network before it will work correctly in the new configuration.

We are in the process of setting up Ciao on the four Nucs on the new network, but it has taken a lower priority than development.

B. Development Progress - Matthew Johnson

The overall progress for development has not been what we hoped due to some assumptions that we made being wrong. We started the term assuming the only thing needing to be accomplished for the first step was implementing Open vSwitch (OVS) created Generic Routing Encapsulation (GRE) tunnels without touching implementation of other network features. This is a very simple requirement for systems with OVS already installed as a dependency. As far as the command line tools go, OVS provides ovs-vsctl to create OVS bridges and tunnels. The simplicity of this tool makes creating a network topology easy to do.

We found a great Golang library called libovsdb [2] that programmatically creates and manages an Open vSwitch network. The libovsdb provides an interface to an OVS database on the host system and allows a program to create and destroy bridges and tunnels.

Creating a tunnel in libovsdb is rather simple, as we displayed for our fall progress for bridge creation (see above - Example insert operation using libovsdb). Creating a GRE tunnel with libovsdb is just as simple if taken as a single operation. The operation is generally the same, except that tunnels are created in the Port table instead of the Bridge table.

Listing 2. Example tunnel insert operation using libovsdb

```
// simple insert operation
insertOp := libovsdb.Operation{
   Op:  "insert",
   Table:  "Port",
   Row:    <port row information, including remote ip>,
   Type:     gre  ,
   UUIDName: namedUUID,
}
```

It was our hope to have the first step done by the fifth or sixth week of the term. At the beginning of this term, and during the writing of all our requirements documentation, it was assumed by us and our clients that GRE tunnels could be created independently of Linux bridges and that the new OVS tunnels could be connected to Linux bridges. There was nothing through the research we had done up to this point that implied differently. This was also the desired result by our client due to the flexibility of Linux bridges. Linux bridges are more versatile than OVS bridges so it was their hope to leave the bridge implementation alone for the first part of the project, then later add an alternate implementation in OVS that could be used if desired.

This assumption led to attempts to use OVS to do exactly that. When we tried this simple implementation of just the Open vSwitch GRE tunnels without creating OVS bridges, we encountered many errors with network setup. With what we know now these errors were very predictable. The errors we encountered drove us to look at other examples of open source projects using libovsdb and we noticed that everyone else was creating OVS bridges before they created OVS tunnels. We were unable to find any examples of mixing networking feature types.

As a result of the above discoveries, we communicated our concerns to our clients and tried to test manual creation of OVS GRE tunnels and Linux bridges. This was easy to do with the command line tool brctl to create the bridge and the Open vSwitch command line interface ovs-vsctl. First we created a Linux bridge with brctl called br1. Then we tried to connect a GRE tunnel created with ovs-vsctl to the bridge br1, but it turned out that Open vSwitch could not even see Linux bridges on the system.

Listing 3. No Linux Bridge and OVS Tunnel

```
mrsj@singlevm:~# sudo brctl addbr br1
mrsj@singlevm:~# sudo brctl show
bridge name bridge id         STP enabled interfaces
br1         8000.000000000000 no
docker0     8000.02427ef725d4 no
mrsj@singlevm:~# sudo ovs-vsctl add-port br1 gre0 -- set interface gre0 type=gre
    options:remote_ip=192.215.10.0
ovs-vsctl: no bridge named br1
```

We communicated our findings to our clients and they agreed that this shows it is necessary to first create the OVS bridge if we want to use the OVS created GRE tunnels. This means it is now necessary to implement the entire network framework with Open vSwitch, a considerable change in scope. This was originally considered a stretch goal for the project, but has now changed to the primary goal, pushing the goal one back to goal two and goal two back to the stretch goal position. Our future work will be to create the OVS framework and create OVS bridges (previously the stretch goal), then create OVS tunnels to connect them (previously goal one). Finally we hope to complete the final goal of creating VxLAN and nvGRE modules (previously goal two).

This problem was identified during the beginning of week six of Winter term. The steps we now have to take include initializing different constructs to watch the Open vSwtich database for updates to the various tables, initialize database caches, create an OVS bridge, create an OVS GRE tunnel, mutate the Bridge table to connect the new tunnel to the bridge. When the tunnel and bridge are no longer needed they must be destroyed and removed from the OVS database. Fortunately, there exist examples, including in the libovsdb source code, for how to do this.

There is another technological hurdle that needs to be tested, however, and we have not had time to do the testing this week due to the midterm report and presentation we have been working on. There may be a hitch when creating an OVS bridge on the controller nodes (CNCIs). The issue may arise because the CNCI runs a DHCP server to assign IP addresses to the compute nodes. Manohar Castelino, the networking expert on the Ciao development team, was concerned there may be issues connecting a DHCP server to an Open vSwitch bridge instance. The way we plan on testing this is to set up two separate virtual machines and using the command line tool ovs-vsctl to initiate first a bridge on both and a tunnel between them. At this point we will try to connect a DHCP server to the bridge on one of the nodes. If network connectivity between the two nodes is maintained and the DHCP server is still able to successfully assign IP addresses then there will be no issues with running OVS bridges on the CNCIs.

Going forward in the next week or so we will first test that a DHCP server can be attached to an Open vSwitch bridge. This should not be an issue but would be the last technical hurdle to overcome to create an OVS framework for Ciao. Once that is determined, and if there are no issues, we will proceed with setting up the OVS framework. This will include all the caching and update-watching for the tables, the creation of bridges on the CNCI and on the compute nodes, and creation of the GRE tunnels to connect them.

C. Issues - Cody Malick

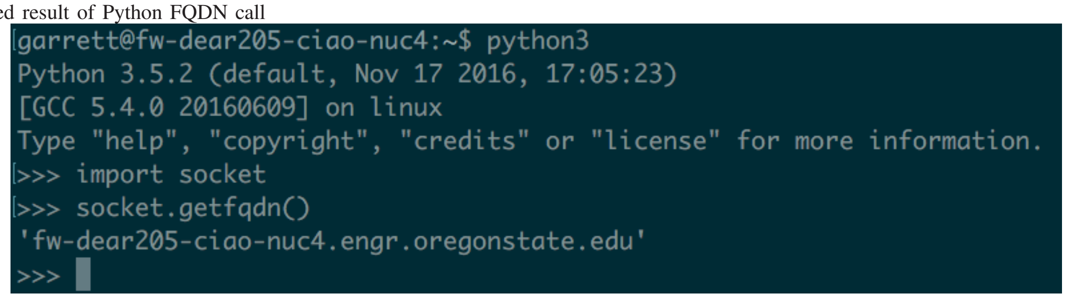
In this section, we will cover the issues we ran into at the beginning of term, as well as the solutions, and discussions that were had around them. The first two major issues are related to environment setup. These two issues consumed a large portion of the first part of the term.

1) Environment Setup - Cody Malick: Environment setup is a pivotal part of our project, as developing and deploying to a non-cloud environment would not show what the project has accomplished. As such, it is important that our team have a good testing environment. The first, and arguably most major issue that was experienced, was setting up the Ciao cluster. The first step of the project is, quite simply, getting Ciao working in the state that it was provided to us. This ultimately should have been fairly straightforward, but we ran into two major issues. The first of which was the management of certificates in Clear Linux.

Clear Linux, Intel's custom Linux distribution, is a fast operating system designed to take advantage of Intel CPU's advanced features that often go under utilized. This was an obvious choice for our team. We initially set up all the NUCs with Clear Linux installed, and proceeded with deployment. We spent about a week tackling some minor configuration issues that came from simply being new to the system. Later, about a week was spent tracking down an issue within Clear Linux. Specifically, that Clear Linux manually manages the network trust store. This isn't an issue in general, but this behavior of Clear Linux is slightly different than Ubuntu's. At first we thought we had simply set a value wrong in a configuration. But after further investigation, it became clear it was an issue unrelated to configuration. The difference in behavior was reported to the Clear Linux development team. They have since shipped a bug fix in a later release. On that front, we were able to iron out Ciao's behavior on Clear Linux.

The second major issue we encountered was that of getting FQDN's in Ciao, or Fully Qualified Domain Names. These were important to the controller node of Ciao as it needed to identify the compute nodes needed to deploy software onto. As we were trying to deploy, we found that Ciao kept attempting to connect to the hostname of the device instead of the FQDN. For example, the control node would try to connect to `fw-dear205-ciao-nuc0` instead of the FQDN, `fw-dear205-ciao-nuc0.engr.oregonstate.edu`. A screenshot of what is expected of the python call is show below. At first, we thought this issue was because of caching in the OS, but quickly found out this was not the case after manually clearing the cache, as well as rebooting the system. After some further investigation, we found that Python3 was only getting the hostname on Clear Linux instead of the FQDN. Ciao relies on certain Python3 function calls. We reported this problem to the Clear Linux team as well, and they have since shipped a fix.

Fig. 1. Expected result of Python FQDN call



```
[garrett@fw-dear205-ciao-nuc4:~$ python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
[>>> import socket
[>>> socket.getfqdn()
'fw-dear205-ciao-nuc4.engr.oregonstate.edu'
>>> 
```

After running into these two issues, we spoke with the Ciao team about our setup, as we wanted to quickly iron out any other testing environment issues. We found out that the Ciao dev team usually uses Ubuntu as its development environment. With that in mind, we decided to move to an Ubuntu development environment to make it as consistent as possible. We have since then installed Ubuntu 16.10 on all the machines. Also at the recommendation of the Ciao team, we started using Ciao-Down.

Ciao-down is a single virtual machine development environment used by the Ciao development team to work on code locally, and simulate a full cluster environment. This is quite a boon for the team as far as independent development. It allows ease of setup, and quick deployment of changes to a simulated environment for testing. An example of how easy it is to set up:[3]

Listing 4. Simplicity of Ciao-Down setup

```
   $ go get github.com/01org/ciao/testutil/ciao-down
   $ ciao-down prepare
```

That's how simple it is to get a single environment setup working with Ciao-Down. Running `ciao-down prepare` spins up a new virtual machine on the local machine. It then installs any necessary tools needed, such as the Go programming language, an open-ssh server, and other tools. After `ciao-down prepare` is run, simply running `ciao-down connect` connects to the newly created virtual machine, which pulls from the Ciao repository on the local machine.

The only caveat to this was that the machine Ciao-Down is running on requires nested kernel-based virtualization. While this is not an uncommon feature, the steps for enabling it were a bit tedious. On the first attempt, we manually edited Grub's (the boot loader for Ubuntu) configuration in order to enable it. But after a helpful hint from one of the good folks at Intel, there is a simple one line command that enables it:

Listing 5. An easy way to enable nested KVM support

```
   echo "options kvm-intel nested=1 | sudo tee /etc/modprobe.d/kvm-intel.conf
```

With this set up, we were able to quickly get to work on the development of the first feature. While we are not currently using a fully deployed version of Ciao for development, we will work on getting Ciao tested with the new features in our full five NUC setup once progress has been made on development.

2) Network Testing - Garrett Smith: Using the automated deployment Ansible play book we were able to deploy Ciao, but we ran into errors running workloads using the Ciao command line interface. For example running `ciao-cli instance add -workload` resulted in the following error message.

Listing 6. Failing to run a workload

```
garrett@fw-dear205-ciao-nuc1:~\$ ciao-cli instance add -workload
   ab68111c-03a6-11e6-87de-001320fb6e31 -instances 1
F0202 09:02:18.353666 8527 instance.go:531] ciao-cli FATAL: HTTP Error [500]
for [POST https://fw-dear205-ciao-nuc1.engr.oregonstate.edu:8774/v2.1/
8624fb4b8d8f489a9a32f051770aeed2/servers]: {"error":{"code":500,"name":"Internal
   Server Error","message":"Unable to Launch Tenant CNCI"}}
```

We obtained a stack trace for the error and traced the error back to the ciao-cli returning a fairly generic error when there is an error

Listing 7. 500 error stack trace portion

```
main.main()
   /usr/src/packages/BUILD/src/github.com/01org/ciao/ciao-cli/main.go:466 +0xf7
```

Garrett did some digging through the Ciao source code and located the source of the error as a new CNCI being created, but not receiving an IP address.

Listing 8. Unable to launch tenant CNCI error location.

```
else if tenant.CNCIIP == "" {
      if !*noNetwork {
         _ = c.addTenant(tenantID)
         tenant, err = c.ds.GetTenant(tenantID)
         if err != nil {
            return err
         }

         if tenant.CNCIIP == "" {
            return errors.New("Unable to Launch Tenant CNCI")
         }
      }
```

Going off of the error message there is network access, but the CNCI is not getting set up with an IP address. We were running Ciao on the OSU network and it looks like the CNCI was trying to obtain an IP address from the OSU network. This will not work because the OSU network is configured to only give out IP addresses to devices with registered MAC addresses.

To get more control over the network we are using Garrett set up an isolated local network using the Cisco SG300 switch Intel is lending us for the project, and is configuring Ciao on that. There are a few places we had hostnames and other settings hard coded into the Ansible playbook deployment configuration. The hostname we were using for Nuc1, the Nuc we are using as the Ciao controller, is set somewhere that is causing problems with the deployment because it is not longer valid. We can not find it in the settings for the Ansible playbook anywhere. Git grep does not return any results for the hostname when run against the deployment repository. Garrett is still investigating this issue at the time this was written.

V. Fall Term Week by Week Reports

A. *Background*

Over the Summer, 2016, Matthew worked as a Software Engineering Intern for the Advanced Systems Engineering (ASE) group within the Open Source Technology Center (OTC) that is in turn within the Software Services Group (SSG) at Intel. Matthew's coworkers had a need to integrate Open vSwitch in their cloud orchestration software (Cloud Integrated Advanced Orchestrator, or Ciao) but did not have the man hours to contribute time to it. Matthew worked with the team to propose the project as a Senior Capstone project at Oregon State University.

Matthew identified two other students, Cody Malick and Garrett Smith, as intelligent hard workers who would benefit the project. Because of this Robert Nesius, the Intel Engineering Manager serving as our client, requested Matthew, Cody, and Garrett specifically for this project.

B. Weeks Zero Through Two

During the first week of class, we all visited Intel in Hillsboro. The principal engineer in charge of Ciao networking, Manohar Castelino, gave us all an explanation of Ciao, how the networking works, and what he expects us to accomplish.

It was also during this time that Rob provided us with five Intel NUCs that would serve as our local cluster. We found out that we needed to register the MAC addresses with the university, and communicated this need to Todd Shechter, the Oregon State University Director of Information Technology.

C. Week Three

During week three we attempted to install Clear Linux OS for Intel Architecture [4] on all five Intel NUCs. We were unsuccessful because the installer requires a network connection to download the packaging. At this point, network access had yet to be approved by OSU IT.

Network access is required to install Clear Linux, Ciao, and access the machines remotely. Since Clear Linux is a datacenter OS, not a desktop OS, it does not support wireless internet connections. Because ethernet is required our hardware must be registered with the university. If we were unable to obtain network access for the hardware on OSU's network we would have needed to find somewhere else to house it.

We also wrote our problem statement in week three, earlier than most groups were able to, since we were ahead of schedule with regard to meeting our team and choosing a project.

D. Week Four

During week four Matthew installed Clear Linux on the Intel NUCs. Since there the networking issues had still not been solved he brought them to his house to use the wired connection there. At this point our hardware had been registered with the university, but for unknown reasons our NUCs were not connecting to the network. Todd Shechter was devoting a lot of his time to help us debug, but we were not yet successful. He set us up with two five-port switches in Kevin McGrath's lab, but they were not receiving IP addresses from the network.

We had our problem statement reviewed and were waiting for feedback by the end of the week. We also started working on the client requirements document, though much of our contributions were simple outline and templating work.

E. Week Five

This week we spent most of our time writing the rough draft for the requirements document. We turned it in by the end of the week and were satisfied with our progress. The final draft of the requirements document was due the next week, week six. This week we also turned in our signed copy of the problem statement.

This week our networking issues were resolved. We had another email conversation with Todd Shechter, who was at a loss as to why we could not access the network from the NUCs. He granted us access to an HP switch we had successfully connected via in the past. On Thursday, we went to move all our NUCs to the new switch, but tried out the network on the mini switches one last time. This time they all worked. Our hardware was now set up and ready to go.

F. Week Six

This week we finished writing the requirements document that was due at the end of the week. After the rough draft we turned in the previous week we continued working on it ourselves until Tuesday. On Wednesday Frank emailed us some suggestions and we addressed those right away. We got the document signed that afternoon by Rob Nesius, our client at Intel.

G. Week Seven

This week we started working on the technical review document due the following Monday. This document outlines nine different components of our system. For each component we explored three different technologies that could be used to implement the component. Since our project is implementing a component of a larger system, it was difficult for us to come up with nine components and three technologies each (twenty-seven different options). We spent much of our week working together to figure out how to split the project up.

H. Week Eight

This week we submitted the tech review document. It was a lot of work, but we got it in on time unlike many other capstone groups. Garrett researched software switch options, network latency tools, and network throughput tools. Matthew dealt with high-level language, testing, and logging tools. Cody handled the network-specific implementation pieces, such as packet protocols, network virtualization implementation, and bridge implementation.

I. Week Nine

This was a short week with Thursday and Friday given over to the Thanksgiving holiday. We started working on the design document but did not make much headway before breaking for the holiday weekend.

Our team also got together and talked about how we were going to execute the final presentation video for a few minutes this week.

J. Week Ten

This week we focused on the design document due Friday. We spent time researching design strategies and writing up our plan to execute. During this research we found a very useful Go library that interfaces with Open vSwitch. This library will simplify our implementation, allowing us more time to do network performance testing, which the client is very interested in. Our client signed the document with a half-hour to spare before the deadline, and the technical advisors for the project, Manohar Castelino and Tim Pepper, indicated they were impressed with the design we had outlined.

VI. Fall Term Retrospective

Week	Positives	Deltas	Actions
0-2	Met the Intel team, studied project goals, purposes	Write project abstract	Research project, write project abstract
3	Started testing hardware setup, problem statement first draft submitted	Resolve networking issues	Contact Todd to get NUC network authorization, write final draft of problem statement
4	Completed hardware setup	Get problem statement signed	Email project owners and get problem statement approved
5	Problem statement submitted, completed first draft of requirements document	Submit final draft of requirements document	Update requirements document, email project owners, get approval via signature
6	Final draft of requirements document submitted	Begin work on tech review	Research technologies for tech review
7	First draft of tech review completed	Finalize tech review	Update and submit tech review
8	Submitted tech review	Begin design document	Research project design steps and implementation details
9	Began work on design document	Complete design document	Fill out the rest of the design document over Thanksgiving weekend
10	Completed design document, began work on final report and final presentation	Complete final report, complete final presentation	Over the weekend, complete final report, create slides for final presentation

VII. Winter Term Week by Week Reports

A. Winter Break and Week One

Over Christmas break Cody and Matthew tried to get Ciao set up on the cluster using a manual installation method. We ran into some tough issues and communicated them with the Intel team. The Intel team pointed us to a much more recently updated setup document that included automated deployment.

During week one there was still an issue in the version of Clear Linux (our target distribution) that we were running that caused docker certification to be broken. This was fixed within the week and we continued trying to set up Ciao.

B. Week Two

Garrett wrote several scripts to collect network performance data between nodes. He made significant progress in parsing bandwidth data into csv format for graphical representation.

This week we continued to work on the Ciao deployment via automated ansible playbooks. We encountered several issues from the start and worked through them one by one. Issues included errors in the ansible playbooks regarding yaml parsing and fqdn configuration. This type of issue was resolved by hardcoding the playbooks for our specific setup. By the end of the week we were seeing certification management issues in the build.

C. Week Three

This week we believed we were successful in deploying Ciao and began implementation of Open vSwitch components. Garrett has begun working out network measurements for our initial benchmarks. Garrett discovered that the network was not being set up properly, however, due to issues with the OSU network.

The first half of the week was spent debugging our Ciao deploy with members of the Ciao development team at Intel. After several email conversations and debug steps, we were advised to switch our operating system to Ubuntu because of various certification issues in Clear Linux. This, along with running from within a ciao-deploy docker container, fixed our issues and allowed us to properly initialize the cluster. As mentioned earlier, our network was not properly set up.

D. Week Four

During week four we continued tentative development on the OVS modules for Ciao. Some of the necessary functions have been written, but it has been difficult to test this over the physical cluster. Garrett, who has worked on the OSU network in the past, thinks the network may not be assigning IPs to the cluster. The OSU network DHCP servers will not assign an IP unless the MAC address has been registered with the university. To get around this we may have to do single-vm setup or find a way to set up the cluster independent from the OSU network.

Ciao provides ciao-down, a tool that helps set up a single-vm environment for testing. This will be helpful once we get it spun up.

We were still having issues gathering initial network metrics due to inaccessible nodes on our physical cluster. Garrett is working through this but the best way to address this may be single-vm for now.

E. Week Five

Early in the week we got ciao-down (the single-vm setup for Ciao) working to test our code.

We have a schedule we are following, and our initial goal was to complete the OVS module by the end of week five. Although we had a module written of something we were hoping might work, we were unable to get it to integrate properly with the rest of Ciao. It is likely we were misunderstanding some things in the calling hierarchy. Matthew communicated the issues we were seeing with Manohar Castelino, one of our clients and a software defined networking expert. This led to a conversation about whether or not it was strictly necessary to create the bridges in Open vSwitch or if we could simply create the GRE tunnels themselves.

As far as the physical cluster goes, we realized that the OSU network will not lease IP addresses unless the MAC address is registered with the University. Since Ciao uses the network's DHCP server to assign IP addresses to the nodes this is a big issue. Garrett has been setting up the physical cluster with a DHCP server running on the switch, with only our deployment NUC connected to the internet. He has been running into issues setting up the Keystone server once he took our controller NUC off the internet. He was trying to modify ansible tasks to get around the issue when I left campus today. If he is not able to get this to work we will probably ask the university for a subnet we can run a non-MAC-locked DHCP server on, but Garrett has worked in OSU IT before and said we are unlikely to get permission for that. The number of errors the OSU network has generated for us has been a little frustrating.

F. Week Six

Matthew met with Manohar in-person at Intel on Tuesday to discuss the issues with OVS bridges. It turns out Open vSwitch will not attach tunnels to non-OVS bridges. This expands our scope somewhat to build a full OVS framework for Ciao, instead of just OVS tunnels.

The rest of the week was devoted to working on the midterm progress report for Winter term.

VIII. Winter Midterm Retrospective

Week	Positives	Deltas	Actions
1	Learned about automated deployment for Ciao	Worked through several deployment issues	Attempted to deploy Ciao
2	Scripts to collect network statistics completed	Resolved old issues in deployment but uncovered more	Hardcoding variables in the ansible deployment resolved some new issues
3	Successfully initiated the cluster	Switched OS on the cluster to Ubunto	Started working in single-vm mode
4	Wrote some of the module for OVS tunnels	OSU network DHCP will not assign IPs to unregistered MACs	Developed in ciao-down
5	OVS module mostly written, but not integrated	Discussed whether OVS bridges were required for OVS tunnels	Tried to set up local DHCP server on switch
6	Met with Manohar regarding OVS bridges	OVS bridges are required for OVS tunnels	Worked on progress report for midterm

References

[1] Socketplane. (2016, dec) play_with_ovs.go. [Online]. Available: https://github.com/socketplane/libovsdb/blob/example/play_with_ovs.go

[2] ——. (2016, dec) libovsdb. [Online]. Available: https://github.com/socketplane/libovsdb/blob/master/README.md

[3] m. mcastelino. (2017, January) Ciao single machine development and test environment. [Online]. Available: https://github.com/01org/ciao/blob/master/DeveloperQuickStart.md

[4] Intel. (2016, dec) clearlinux.org. [Online]. Available: https://clearlinux.org/