

Intel Cloud Orchestration Networking

Winter Final Report

Cody Malick

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	Current Progress	2
IV	Issues	3
IV-A	Environment Setup	3
V	Team Member Evaluations	5
V-A	Team Member Roles	5
V-B	Garrett Smith	5
V-C	Matthew Johnson	5
VI	Winter Term Week by Week Reports	6
VI-A	Winter Break and Week One	6
VI-B	Week Two	6
VI-C	Week Three	6
VI-D	Week Four	6
VI-E	Week Five	7
VI-F	Week Six	7
VI-G	Week Seven	7
VI-H	Week Eight	8
VI-I	Week Nine	8
VI-J	Week Ten	8
VII	Retrospective	9
	References	9

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. CURRENT 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.

In the latter half of the term, there was a large change in scope. We had discovered that, in order to utilize Open vSwitch, we had to use OVS generated bridges, as well as OVS generated tunnels. The primary difference between the old scope and the new is generating bridges using OVS. What this means is that there is quite a bit of implementation work that had to be done to get OVS bridges created. The changes were discussed and approved with the team at Intel. After further discussion, it was agreed that we could use a direct call to the OVS command line utility, `ovs-vsctl`.

To call the command line utility from Go code, we needed to use Go's `exec` function. What `exec` allows a program to do is call third party tools from a shell environment. Here is some sample code on creating an OVS generated bridge while passing a function's arguments as arguments to the command line tool:

Listing 1. Simple OVS bridge creation function

```
func createOvsBridge(bridgeId string) error {
    // Example: ovs-vsctl add-br ovs-br1
    args := []string{"add-br", bridgeId}
    cmd, err := exec.Command("ovs-vsctl", args...).Start()
    if err != nil {
        return err
    }
    return nil
}
```

With functions similar to this one, implementation of the OVS command line utility has gone quite smoothly. The main portion of the work that has been done in the last few weeks have been focused on adding a new network mode to Ciao. With the addition of a new type of bridge, all the accompanying programmatic logic to change modes must be put in place. Ciao did not previously support anything outside of the default mode. As such, it has been quite challenging and fun figuring out the best way to implement our code into the existing architecture. We are well on our way to a prototype, and will continue work over spring break. Our goal to is to having a working alpha to show on returning next term.

IV. ISSUES

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.

A. Environment Setup

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 shown 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.

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:[1]

Listing 2. 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 3. 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.

V. TEAM MEMBER EVALUATIONS

A. Team Member Roles

While all member contributed quite a bit this term, each of the team had specializations. My specialization was focusing on development, understanding how to use OVS, as well as being a language expert in the Go programming language. I am also the elected team leader and handle any administrative tasks as needed.

B. Garrett Smith

Garrett focused on setting up a proof of concept for networking, as well as writing and setting up network testing environments. Garrett has been a reliable and helpful teammate these last two terms. I have no suggestions to improve his behavior. Garrett has focused primarily on the network testing, performance, and prototyping of the Ciao cluster. While this portion of the project is far less glamorous than development, Garrett has tackled the many challenges accompanying it with no complaints. He has solved difficult problems, and done an excellent job documenting his work. Having completed the proof of concept portion of the networking, he has joined Matthew and I in completing the development of the Ciao prototype.

C. Matthew Johnson

Matthew specialized in understanding the Ciao architecture, and worked extensively on development. Matthew also served as the point of contact for our team and the team at Intel. Matthew has been a stronghold of production this term, and I feel the spearhead of development. I believe he has made the greatest contribution in terms of hours spent on development. He has quickly picked up the Go programming language, and has been our team's expert on the Ciao architecture. As far as behavior, I again have no complaints or suggestions. Matthew has always pushed our team to be productive, and continue moving forward even when things seemed grim. He has been a leader and a great teammate.

VI. 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.

G. Week Seven

Week seven was devoted entirely to development. A lot of good progress was made on implementing the Open vSwitch. We discovered that a scope change was needed. OVS requires a full integration and use of OVS bridges, not just the tunnels. We also were able to get basic tests running on code for fast debugging of code. We also were able to get our documents signed and submitted week seven.

H. Week Eight

Week eight was slow for the team. Cody was out sick, and Matthew was moving up to Portland. Matthew was able to speak to Manohar about the scope change. With approval, we got started on getting the command line functions set up. Garrett was also able to make some progress with getting docker containers working with an OVS bridge.

I. Week Nine

Week nine the team fully fleshed out the functions needed to call the OVS command line utility using the `exec` function call. With those functions in place, the next step was adding multiple network mode capability to Ciao.

J. Week Ten

A large portion of week ten was dedicated to working on the first poster draft. We got some feedback from Frank on our poster, then submitted it on time. Development wise, we found that Ciao uses a `bridge` data structure to build and track the state of the network. The `bridge` struct is only utilized with Linux generated bridges. We started work adding multiple mode capability to support the new code being written.

VII. 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 Ubuntu	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
7	Large scope change	Documents signed	Development progress, scope change defined
8	Good progress on OVS implementation	Approved scope change	Wrote functions to create OVS bridges and tunnels
9	Completed OVS command line functions	Adding network switches to Ciao data structures	Started Ciao integration of new network mode
10	Ciao implementation almost working, close to prototype	Poster created	Created poster, had it reviewed, and submitted

REFERENCES

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