CS 461 - Fall 2016 - Client Requirements Document

Matthew Johnson, Garrett Smith, Cody Malick

Cloud Orchestra

Abstract

This document outlines the requirements for the Cloud Orchestration Networking project sponsored by Intel Corporation. It formally defines the purpose, scope, description, function, use, constraints, and specific requirements of the project. Although there are no specific design decisions made, it will be used as a building block for the rest of the design, implementation, and testing process.

CONTENTS

1 Introduction

1.1 Purpose

Intel, while developing its Software Defined Network implementation into Ciao, has found a need for a more advanced form of network bride than the standard Linux bridge. The initial implementation using Linux bridges and GRE tunnels has worked well, but as further development was done on Ciao, the need for modern packet encapsulation and other innovative protocols were found to be needed. Implementing a network mode in Ciao utilizing Open vSwitch-created GRE tunnels would allow the network to utilize advanced networking techniques to increase performance, such as modern packet encapsulation methods. This addition would be used by those implementing Ciao in their own organizations or businesses to further increase speed and availability of features in their cloud.

1.2 Scope

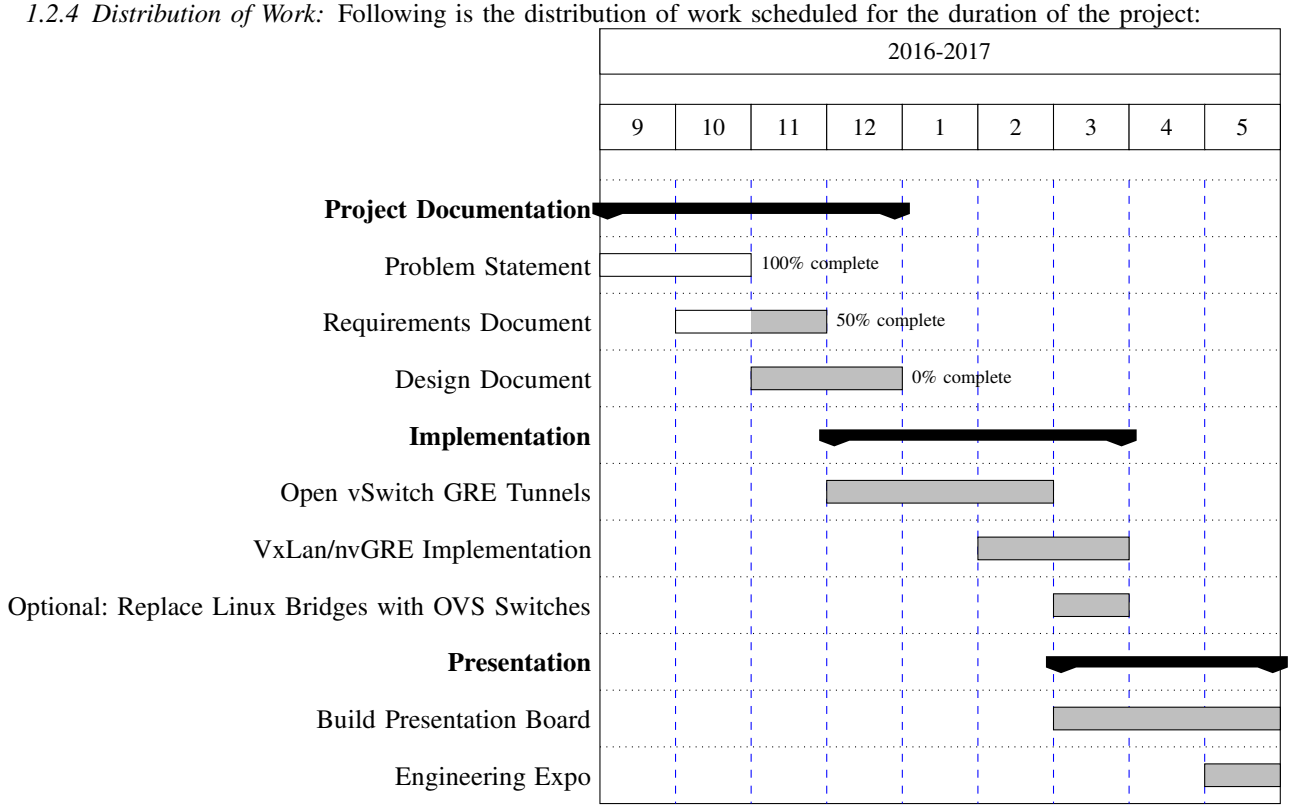
The scope of the Cloud Orchestration project encapsulates the following two main goals, followed by one additional stretch goal:

1.2.1 Open vSwitch GRE Tunnel: The first goal of the project is to switch the GRE tunnel implementation with the Open vSwitch created GRE tunnel. This will allow for newer packet encapsulation techniques to be used, as well as provide the option to test packet acceleration.

1.2.2 Test and Implement Best Performing Tunnel Implementation: Switch the tunneling implementation to VxLAN/n-vGRE based on performance measurements of VxLAN and nvGRE on data center network cards.

1.2.3 Stretch Goal, Replace Linux Bridge: The final objective and stretch goal of the project is to replace the Linux bridges with Open vSwitch instances.

1.2.4 Distribution of Work: Following is the distribution of work scheduled for the duration of the project:



1.3 Definitions, acronyms, and abbreviations

Bridge	Software or hardware that connects two or more network segments.
Cloud	A huge, amorphus network of servers somewhere [1].
Cloud Orchestration	An easy to deploy, secure, scalable cloud orchestration system which handles virtual machines, containers, and bare metal apps agnostically as generic workloads [2].
CNCI	Virtual Machines automatically configured by the ciao-controller, scheduled by the ciao-scheduler on a need basis, when tenant workloads are created [3].
Generic Routing Encapsulation (GRE)	Encapsulation of an arbitrary network layer protocol so it can be sent over another arbitrary network layer protocol [4].
Linux Bridge	Configurable software bridge built into the Linux kernel [5].
Network Node (NN)	A Network Node is used to aggregate network traffic for all tenants while still keeping individual tenant traffic isolated from all other the tenants using special virtual machines called Compute Node Concentrators (or CNCIs) [3].
nvGRE	Network Virtualization using Generic Routing Encapsulation [6].
Open vSwitch	Open source multilayer software switch with support for distribution across multiple physical devices [7].
OVS	Open vSwitch [7].
Packet Acceleration	Increasing the speed of the processing and transfer of network packets.
Packet Encapsulation	Attaching the headers for a network protocol to a packet so it can be transmitted using that protocol [8].
SSNTP	The Simple and Secure Node Transfer Protocol (SSNTP) is a custom, fully asynchronous and TLS based application layer protocol. All Cloud Integrated Advanced Orchestrator (CIAO) components communicate with each others over SSNTP [9].
Tunnel	Point to point network connection that encapsulates traffic between points [8].
VxLAN	Virtual Extensible Local Area Network [10].

1.4 References

[1] R. Munroe. (2011, jun) The cloud. [Online]. Available: http://xkcd.com/908/

[2] T. Pepper, S. Ortiz, and M. Simental. (2016, sep) Ciao project. [Online]. Available: https://github.com/01org/ciao/blob/master/README.md

[3] M. Castelino. (2016, may) Ciao networking. [Online]. Available: https://github.com/01org/ciao/blob/master/networking/README.md

[4] F. . T. Hanks, Li. (1994, oct) Generic routing encapsulation (gre). [Online]. Available: https://tools.ietf.org/html/rfc1701

[5] T. L. Foundation. (2016, nov) Bridge. [Online]. Available: https://wiki.linuxfoundation.org/networking/bridge

[6] M. Sridharan, A. Greenberg, N. Venkataramiah, K. Dudam, I. Ganga, and G. Lin. (2015, sep) Rfc7637. [Online]. Available: https://tools.ietf.org/html/rfc7637

[7] (2016, nov) Open vswitch. [Online]. Available: http://www.openvswitch.org/

[8] J. Kurose and K. Ross, *Computer Networking*, 6th ed. Pearson, 2012.

[9] S. Ortiz, J. Andersen, and D. Lespiau. (2016, sep) Simple and secure node transfer protocol. [Online]. Available: https://github.com/01org/ciao/blob/master/ssntp/README.md

[10] M. Mahalingam. (2014, aug) Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks. [Online]. Available: https://tools.ietf.org/html/rfc7348

[11] (2016, apr) Ciao network topology. [Online]. Available: https://github.com/01org/ciao/blob/master/networking/documentation/ciao-networking.png

1.5 Overview

The following section describes more details about the product, including product perspective, specific requirements, functionality requirements, and any assumptions or dependecies used. The section is organized in the following fashion:

1) Overall Description
2) Product Perspective
3) Product Functions
4) User Characteristics
5) Constraints
6) Reliability
7) Security
8) Specific Requirements

2 Overall Description

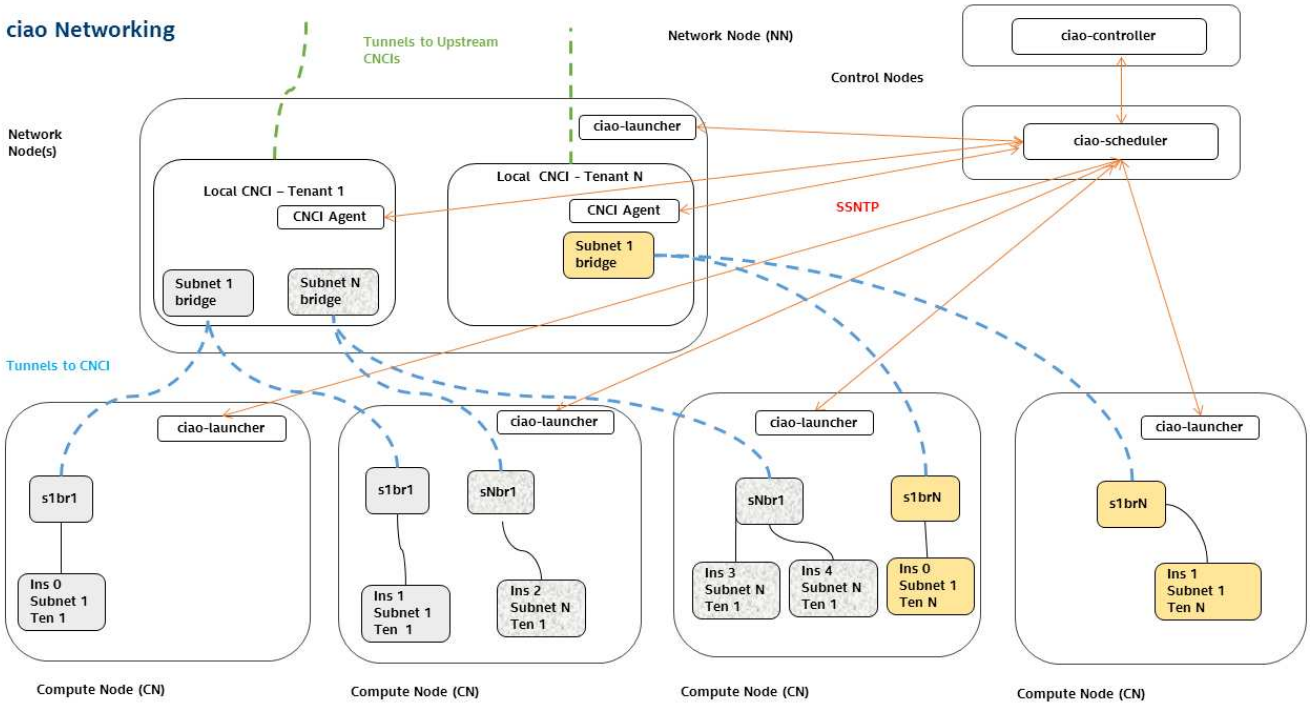
2.1 Product Perspective

The SDN we implement will be utilized by Ciao to transfer packets between compute nodes and control nodes on a cloud cluster. For this purpose the mode must be fully integrated into the Ciao infrastructure and must behave similarly to what is already in place.

Because this software will be a component of a larger system, it must follow the design of that larger system and be fully integrated. Because Ciao is implemented in the Go programming language, so too must this SDN implementation be written in Go. The networking mode must route packets between ports on different nodes using networking protocols.

The SDN that is implemented must support the following structure of the cloud in Ciao:

Fig. 1. Ciao Network Topology [11]



Ciao has several functional components that must be worked with:

libsnnet Provides networking APIs to the ciao-launcher to create tenant specific network interfaces on compute nodes and CNCI specific network interfaces on a network node.

ciao-cnci-agent A SSNTP client which connects to the ciao-scheduler and runs within a CNCI VM and configures tenant network connectivity by interacting with the ciao-controller and ciao-launchers using the ciao-scheduler. The ciao-cnci-agent can also be run on physical nodes if desired.

docker-plugin Provides unified networking between VM and Docker workloads.

2.2 Product Functions

Ciao networking must support the following functionality.

1) Secure isolated overlay network - The networking implementation must provide each tenant with a secure isolated overlay network without the need for any configuration by the tenant and minimal configuration by the data center operator.
2) Auto discovery of compute/network nodes - Auto discover roles of nodes when they are attached to the network.
3) Diverse and scalable - Support large number of tenants with large or small number of workloads.
4) Work on different platforms - Operate on any Linux distribution by limiting the number of dependencies on user-space tools and leveraging Linux kernel interfaces whenever possible.
5) Migrate workloads - Provide the ability to migrate workloads from a Compute Node on demand or when a CN crashes without tenant intervention. Provide the ability to migrate CNCIs on demand or when a Network Node crashes.
6) Encrypt traffic - Provide the ability to transparently encrypt all tenant traffic even within the data center network.

7) Create GRE tunnels with Open VSwitch
8) Support for tenant and workload level security rules
9) Support for tenant and workload level NAT rules (inbound and outbound)

2.3 User characteristics

The users of this mode in Ciao will be educated and technically-minded data center administrators. They will be familiar with cloud technologies and networks, but software-defined and hardware-defined.

Despite this, however, due to Ciao's goal of minimum configuration the user may have a lower knowledge base than, by contrast, the normal network administrator for an openstack cluster.

2.4 Constraints

2.4.1 High-order language requirements: As mentioned above, it is necessary to write this mode in the Go programming language in order to integrate with the rest of Ciao.

2.4.2 Reliability: The networking mode must be completely reliable and include the ability to migrate workloads when a compute or network node crashes with little-to-no interruption to the network capabilities for the tenants and must do so without tenant intervention.

2.4.3 Security: Traffic must be fully and transparently encrypted even within the data center network. It must utilize Simple and Secure Node Transfer Protocol (SSNTP) to ensure security of the traffic between the nodes.

3 Specific Requirements

The main requirement is that Open VSwitch is used to create the GRE tunnels in the SDN implementation. A further goal is to switch the tunneling implementation to VxLAN or nvGRE based on performance measurements of each. The result of those performance metrics will dictate which is used.

A stretch goal is to replace the currently-implemented linux bridges with Open VSwitch switch instances.

The specific requirements for this project are as follows:

1) Documentation
   a) Problem statement
   b) Requirements document
   c) Design document
2) Implementation
   a) Open vSwitch GRE tunnels
   b) VxLAN/nvGRE implementation
   c) Optional: replace Linux bridges with OVS switches
3) Presentation
   a) Build presentation board
   b) Present at Engineering Expo