Intel Cloud Orchestration Networking Design Document

Abstract

This document outlines the design considerations for the implementation of Open vSwitch and other networking technologies in the Cloud Integrated Advanced Orchestrator (Ciao), Intel Corporation's advanced cloud orchestration software. It describes the various techniques, structure, and technology choices that will be used in the execution of our project.

Date of Issue: December 2nd, 2016

Issuing Organization: Intel Corporation

Authorship: Matthew Johnson, Cody Malick, and Garrett Smith

Change History: First Draft, 12-02-2016

Contents

I. Introduction

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. This document outlines the steps, techniques, and methodology we will utilize to achieve each goal.

A. 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 vSwtich (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.

B. Scope

Ciao exists as a cloud orchestrator for cloud clusters. It is inherently Ciao exists as a cloud orchestrator for cloud clusters. It is inherently necessary for the separate nodes in the cloud cluster to be able to talk to each other. Without a reliable and secure software defined network Ciao would have little purpose. Utilization of Open VSwitch GRE tunnels allows Ciao to become more scalable and enables the inclusion of packet-acceleration technology such as the Data Plane Development Kit (DPDK).

C. Context

Our network mode will exist within Ciao, a cloud orchestrator designed to be fast and easy to deploy. Ciao is sectioned into three parts, each with distinctive purposes [1].

Controller	Responsible for policy choices around tenant workloads [1].
Scheduler	The Scheduler implements a ”push/pull” scheduling algorithm. In response to a controller approved workload instance arriving at the scheduler, it finds a first fit among cluster compute nodes currently requesting work [1].

Launcher The Launcher abstracts the specific launching details for the different workload types (eg: virtual machine, container, bare metal). Launcher reports compute node statistics to the scheduler and controller. It also reports per-instance statistics up to controller [1].

Our networking mode must facilitate the communication of packets between all three levels of Ciao, as well as individual compute and network nodes and the Compute Node Concentrator (CNCI) [2].

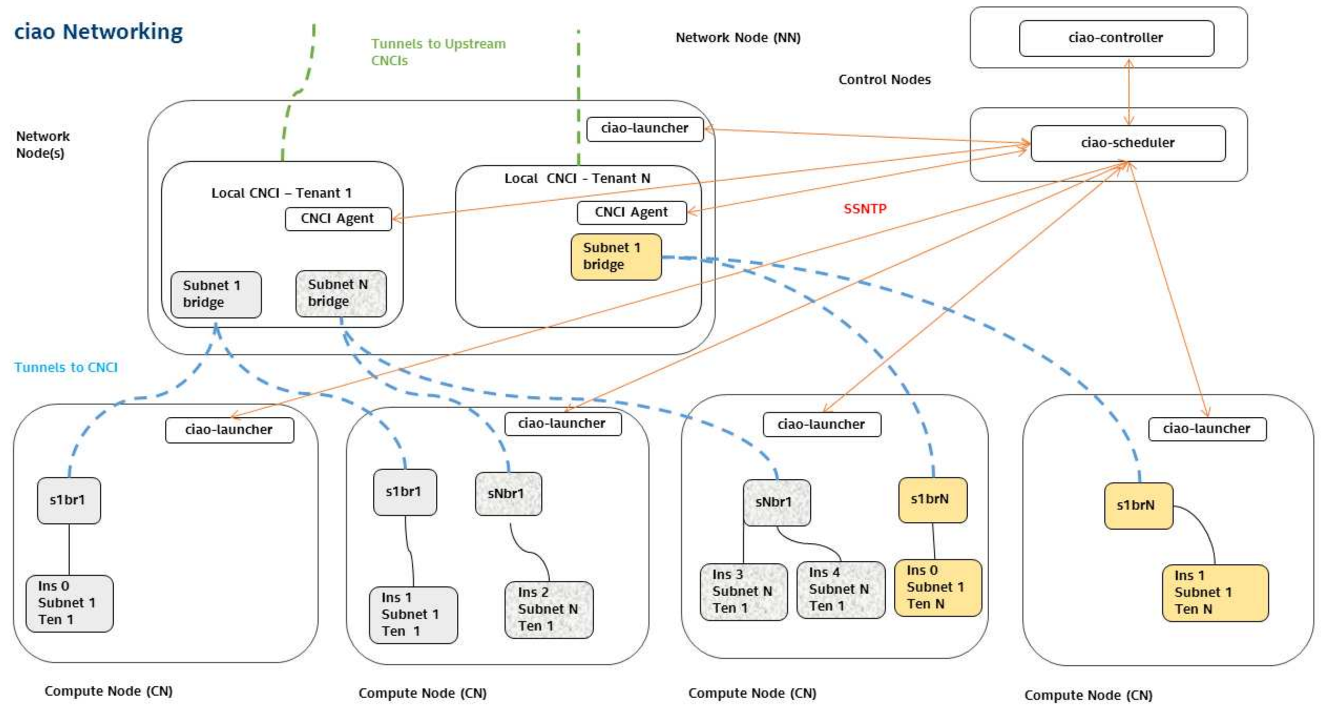
Compute Node A compute node typically runs VM and Container workloads for multiple tenants [2].

Network Node 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 (CNCIs) [2].

Compute Node Concentrator (CNCI)
CNCIs are Virtual Machines automatically configured by the ciao-controller, scheduled by the ciao-scheduler on a need basis, when tenant workloads are created [2].

Specifically, the Ciao network components must communicate securely using the Simple and Secure Node Transfer Protocol (SSNTP). The network node aggregates traffic between compute nodes while keeping the tenant traffic isolated from other tenants in the cluster. Network nodes achieve this with CNCIs. A graphic of the lowest-level of this network configuration shows their relation to each other.

Fig. 1. Ciao Network Topology [3]



D. Summary

We will implement an Open vSwitch Generic Routing Encapsulation (OVS-GRE) mode in Ciao in order to leverage DPDK and other software defined networking technology innovations which are dependent on OVS. This document will outline our design strategy, design views, and design viewpoints for each component of our solution.

II. References

[1] T. Pepper, S. Ortiz, M. Ryan *et al.* (2016, sep) Ciao readme. [Online]. Available: https://github.com/01org/ciao/blob/master/README.md

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

[3] (2016, apr) Ciao network topology. [Online]. Available: https://github.com/01org/ciao/blob/master/networking/documentation/ciao-networking.png

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

[5] DPDK. What it is. [Online]. Available: http://dpdk.org/

[6] D. Thaler. (2000, nov) Multipath issues in unicast and multicast next-hop selection. [Online]. Available: https://tools.ietf.org/html/rfc2991

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

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

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

[10] (2016, nov) Open vswitch. [Online]. Available: http://www.openvswitch.org/

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

[12] 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

[13] 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

[14] E. B. Pfaff, B. David. (2013, dec) The open vswitch database management protocol. [Online]. Available: https://tools.ietf.org/html/rfc7047

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

[16] ——. (2016, dec) play_with_ovs.go. [Online]. Available: https://github.com/socketplane/libovsdb/blob/5113f8fb4d9d374417ab4ce35424fbea1aad7272/example/play_wit

[17] S. Yegulalp. (2015, jun) What's the go language really good for? [Online]. Available: http://www.infoworld.com/article/2928602/google-go/whats-the-go-language-really-good-for.html

III. Glossary

Bridge	Software or hardware that connects two or more network segments.
Ciao	Ciao is a cloud orchestrator that provides an easy to deploy, secure, scalable cloud orchestration system which handles virtual machines, containers, and bare metal apps agnostically as generic

	workloads. Implemented in the Go language, it separates logic into ”controller”, ”scheduler” and ”launcher” components which communicate over the ”Simple and Secure Node Transfer Protocol (SSNTP)” [1].
Cloud	A huge, amorphous network of servers somewhere [4].
Cloud Orchestration	A networking tool designed to aid in the deployment of multiple virtual machines, containers, or bare-metal applications [1].
Compute Node Concentrator (CNCI)	Virtual Machines automatically configured by the ciao-controller, scheduled by the ciao-scheduler on a need basis, when tenant workloads are created [2].
Data Plane Developement Kit (DPDK)	DPDK is a set of libraries and drivers for fast packet processing. It was designed to run on any processors. The first supported CPU was Intel x86 and it is now extended to IBM Power 8, EZchip TILE-Gx and ARM. It runs mostly in Linux userland [5].
Equal Cost Multipath Routing (ECMP)	Equal cost multipath routing is a routing strategy in which next path routing for a packet can occur along one of several equal-cost paths to the destination [6].
Generic Routing Encapsulation (GRE)	Encapsulation of an arbitrary network layer protocol so it can be sent over another arbitrary network layer protocol [7].
Linux Bridge	Configurable software bridge built into the Linux kernel [8].
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 (CNCIs) [2].
nvGRE	Network Virtualization using Generic Routing Encapsulation [9].
Open vSwitch	Open source multilayer software switch with support for distribution across multiple physical devices [10].
OVS	Open vSwitch [10].
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 [11].

SSNTP	The Simple and Secure Node Transfer Protocol (SSNTP) is a custom, fully asynchronous and TLS based application layer protocol. All Ciao components communicate with each others over SSNTP [12].
Tunnel	Point to point network connection that encapsulates traffic between points [11].
VxLAN	Virtual Extensible Local Area Network [13].

IV. Body

A. Design Stakeholders

The stakeholders are Intel, and the members of the Oregon State University capstone group working on the project, Matthew Johnson, Cody Malick and Garrett Smith. Additional obvious stakeholders are the existing and future users of Ciao.

B. Design Concerns

The design concerns are the operating systems that need to be supported, the implementation of Open vSwitch created GRE tunnels, the implementation of VxLAN tunnels, the implementation of nvGRE tunnels, gathering performance metrics for the tunneling implementations, replacing the Linux bridges used by Ciao with Open vSwitch switch instances, logging, and testing. The capstone group members are stakeholders interested in all of the design concerns because they will be implementing all of the concerns. Intel is interested in all of the design concerns because they are the clients and they will be using Ciao.

C. Context Design Viewpoint

The context design viewpoint used here is from the IEEE 1016-2009 standard [**?**].

1) Context Design Concerns: The design concerns for the context viewpoint are related to overall compatibility both with the platform the software is running on as well as the software the networking mode is running in. It must also be compatible with existing networking protocols as used within Ciao already. Other major concerns include security and performance of the software defined network.

There is one primary use case initiated by a compute node sending a data packet to another compute node within the SDN.

Actor	Compute Node 1
Precondition	Compute Node 1 attempts to send data over network to Compute Node 2.
Postcondition	Compute Node 2 receives data sent by Compute Node 1
Main Path	1) Compute Node 1 encapsulates packet. 2) Compute Node 1 sends packet to Network Node. 3) Network Node routes packet via relevant CNCI to Compute Node 2. 4) Compute Node 2 receives packet and de-encapsulates.

2) Design Entities: The users of the networking mode we are implementing are the current and future users of Ciao, system administrators of small to medium enterprises running private clouds.

The actors of the networking mode are the components of Ciao networking, the compute nodes, network nodes, and CNCIs. These nodes handle the encapsulation, routing, and de-encapsulation of packets.

3) Design Relationships: Compute nodes both send and receive packets to other compute nodes. Network nodes aggregate this tenant network traffic while CNCIs within the network nodes keep tenant traffic isolated from each other.

4) Design Constraints: Design of the solution must integrate fully with Ciao as well as the Linux family of operating systems it runs on. This constraint demands specific technology choices as described below in the Context Design View section.

D. Context Design View

Compatibility with the rest of the Ciao system is paramount and drives many design decisions. The network implementation will be done in the Go programming

Because compatibility with the reset of the Ciao system is paramount, our software defined network will be written in the Go programming language and fully integrated in to Ciao. The Go programming language was selected for several reasons, including the efficiency of the language regarding both speed and memory, the concurrency capabilities, and the ease of implementation. Go was compared against C and Python as alternatives, and prevailed in every criteria except for availability of the language.

This network mode will be written as a standalone networking mode for Ciao as an additional option to the standard Linux bridges available now. For this reason it must be fully integrated with the Ciao networking framework as it currently exists [2].

Since Ciao targets the Linux family of operating systems, our networking solution must also support Linux operating systems.

E. Interface Design Viewpoint

One of the main project goals is switching the GRE tunneling implementation from standard GRE tunnels to Open vSwitch generated GRE tunnels. The implementation of this object will be through designing and building an interface for Open vSwitch to hook into Ciao.

1) Design Concerns: Ciao will need to implement an interface for Open vSwitch in order to utilize the new feature set that OVS supplies. In order to access the OVS Management Database, we will need to access the Open vSwitch Database Management Protocol [14]. The following functions of the protocol will need to be created: [14]

Function Name	Parameters	Description
Insert	*table* : required *row* : required *id* : optional	The operation inserts specified value into table at row. If no id is specified, then a new unique id is generated.
Select	*table* : required *where* : required *columns* : optional	Searches *table* for rows that match all the conditions specified in *where*. If *columns* is not specified, all columns from the table are returned.
Update	*table* : required *where* : required *row* : required	Updates specified rows in a table. It searches *table* for rows that match all conditions specified in *where*.
Delete	*table* : required *where* : required	Operation deletes all the rows from *table* that match all the conditions specified in *where*

2) Design Elements: The primary method of interaction with Open vSwitch is through the Configuration Management Database, which provides a programmatic interface for updating and changing OVS on the fly. This will be the primary interface for our implementation.

F. Interface Design View

In order to interact with the interface using Golang, the implementation will use the libovsdb library[15]. Libovsdb provides a direct interface for Golang to access and modify the OVS Database configurations. The following is an example function call to call the above functions:[16]

Listing 1. Example insert operation in the OVS Database

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

Using the above code, slightly altered for each required operation, Golang can be used to insert, select, update, and delete using a standard format.

G. Interaction Design Viewpoint

H. Interaction Design View

I. Resource Design Viewpoint

J. Resource Design View

K. Design Rationale

Our design decisions were based largely around the goal of maintaining compatibility with Ciao and the platforms that Ciao runs on, as well as improving the capabilities and performance of Ciao's SDN. Implementation will be done in Go in order to integrate with the larger system and because the Go programming language is platform independent [17].

Open vSwitch was chosen because of the various features offered. OVS has support for packet acceleration frameworks such as DPDK and advanced SDN tunneling protocols such as nvGRE and VxLAN which would allow leverage of ECMP routing and potentially increase network performance.

V. Signatures

________________________ Robert Nesius, Engineering Manager

________________________ Matthew Johnson

________________________ Garrett Smith

________________________ Cody Malick