Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Studiengang Computer Engineering
Fachbereich 1
Prof. Dr. Thomas Baar

Design and implementation of a connectivity manager for virtual scalable network environments

Bachelorarbeit
von

Manuel Bergler

01.12.2014 – 08.02.2015

Referent:	Prof. Dr. Thomas Baar
Korreferent:	Prof. Dr. Thomas Baar
Betreuer:	Benjamin Reichel M. Sc.

Manuel Bergler
Urbanstr. 26
10967 Berlin

Hiermit versichere ich, dass ich die von mir vorgelegte Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Berlin, den 08. Februar 2015

(Unterschrift)

Manuel Bergler

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

The demands on networks have changed dramatically in the past two decades, with an ever-growing number of people and devices relying on interconnected applications and services. The underlying infrastructure has been left mostly unchanged and is approaching its limits. In order to resolve this, Software Defined Networking (SDN) is going to be extending and replacing parts of traditional networking infrastructures. SDN separates the network into control and forwarding planes and therefore allows a more efficient orchestration and automation of network services.

The use of cloud-based services, with not only competitive pricing but also high-availability and fast network access, is taking over the traditional self-hosted data centers. The ease of administration and the deployment of new Virtual Machines (VMs) on the fly make it possible to create a Topology of Computers with no effort.

Network services have different requirements, depending on the type of data and their importance. The classification of network traffic can be done through Quality of Service (QoS). A new approach has to be made to enable the use of QoS in virtualized cloud infrastructures like OpenStack, to achieve controlled traffic from the deployment of Virtual Machines on.

1.2 Network Architecture

Today's traffic patterns, the rise of cloud computing and "big data" to only name a few examples, are exceeding the capacity of classic network architectures. With scalable computing and storage the common-place tree-structured network infrastructure with Ethernet switches are not efficient and manageable enough.

The increasing complexity of problems that have to be faced in networks and the need to control network traffic through software, are only a selection of the reasons why the Open Networking Foundation (ONF) developed an approach called Software-Defined Networking (SDN).

SDN is a leading-edge approach where the network control is separated from the forwarding functions. The centralized network intelligence allows programming the network, without a need to access the underlying infrastructure. Therefore a shift of today's networks to more flexibility, programmability and scalability is going to take place.

1.3 Objective

The primary objective of this work is the development of a network orchestrator which is able to apply Quality of Service to the network interfaces of Virtual Machines. These Virtual Machines are deployed with OpenStack Nova and connected to an OpenVSwitch, which uses OpenFlow. Another task of the Connectivity Manager is to select which OpenStack hypervisor new VMs should be running on, which takes different runtime parameters into account. The CM should be able to be applied in environments with scalable hypervisors and VMs.

1.4 Scope

The scope of this work includes a Connectivity Manager which will have a Connectivity Manager Agent running on the cloud controller within the OpenStack infrastructure, to provide access to the hypervisors of OpenStack Nova. These two components have to be implemented and integrated with the existing OpenVSwitches. As a reference for a cloud infrastructure, multimedia communications like the Nubomedia project will be used. The deployment of this cloud is then tested on different performance characteristics like network bandwidth, latency, CPU utilization and memory usage.

In virtualized cloud infrastructure like OpenStack, the placement of Virtual Machines (VMs) on a particular compute node can be decided on by comparing different run-time parameters. The network connectivity between those VMs has to be prioritized and classified into different classes, depending on the service that are running on it.

Currently there are a number of solutions for managing network connectivity between VMs. A comparison and their current limitations follows in the next section. The chosen approach is to extend the existing network control and management services with Quality of Service (QoS) capabilities. In support of the thesis the Connectivity Manager will be implemented and the differences in bandwidth usage will be shown in one use-case.

1.5 Overview

Chapter 1 begins with the motivation for this thesis and gives a brief introduction into the objectives and the scope.

Chapter 2 gives an overview of traditional network concepts and a introduction to SDN and its components. Furthermore the different services that make up OpenStack will be described.

Chapter 3 conceptualizes the state-of-the-art solutions that are currently available and evaluates their implementation and limitations.

Chapter 4 contains an analysis of requirements and an architectural overview of the Connectivity Manager. Moreover design aspects are introduced and illustrated according to their requirements.

Chapter 5 examines the implementation of the Connectivity Manager and Agent.

Chapter 6 evaluates the network performance tests on the basis of a particular use-case.

Chapter 7 summarizes the results of this work and gives an overview on possible future work.

Chapter 2

Fundamentals and related work

2.1 Software-Defined Networking

The origin of Software-Defined Networking (SDN) began already in 1995, however the first use cases were only developed in 2001 and the promotion of SDN only began with the foundation of the non-profit industry consortium Open Networking Foundation (ONF) in 2011. The ONF is dedicated to push and adapt open standards like the OpenFlow into the industry. In this following section a brief overview of the SDN architecture and concepts, including the OpenFlow protocol is given.

2.1.1 Motivation

Today's internet is part of the modern society, be it for private users, enterprises or vital infrastructure services. Networks are required to evolve in order to address the challenges that are entailed with new applications, services and a growing number of end-users.

With a more detailed view on the challenges of current networks one comes to see the following limitations:

- **Inability to scale**: With the expansion of data centers, networks must grow too. Configuring and managing these additional network devices comes at a high administrative effort. With the virtualization of data centers network traffic patterns becomes more and more dynamic and unpredictable. With multi-tenancy a further complication is introduced, because different end-users and services need different network performance and might require traffic steering. Such scaling and network management cannot be done with a manual configuration of the underlying infrastructure.
- **Complexity**: In the past decades new networking protocols have been adapted by the industry. To add or move any device, multiple existing switches, routes, firewalls must be touched in order to manage protocol-based mechanisms on a device-level. With the virtualization of servers the amount of interfaces that need network connectivity and the distribution of applications over a number of virtual machines (VMs) are another demand that the current fairly static networks cannot dynamically adapt to.

- **Inconsistent policies**: For IT to apply a network- or data center-wide policy a lot of devices and mechanisms may need to be reconfigured. Virtual Machines are created and rebuilt within no time, but if for example access or security needs to be updated, the benefits of this dynamic are subverted(?).
- **Vendor dependence**: Standards are needed to match the requirements of the markets with the capabilities of networks and enable network operators to customize the network to specific environments.

Traditionally decisions about traffic flowing through the network are made directly by each network device, because the control logic and forwarding hardware are tightly coupled.

2.1.1.1 Classical switches & routers

Packet forwarding (data plane) and routing decisions (control plane) in classical switching and routing are both within one device. In figure .. the main components that are depicted have the following functions:

1. The **forwarding path** typically handles data path operations for each packet. It generally consists of Application-Specific Integrated Circuits (ASIC), network-processors or general-purpose processors that forwards frames and packets at wire speed (line-rate). Their lookup functions can be further increased with memory resources like Content Addressable Memory (CAM) or Ternary Content Addressable Memory (TCAM) to contain the forwarding information.
2. The elements in the **control plane** are based on general-purpose processors that provide services like routing and signaling protocols, including ARP, MAC Learning and forwarding tables.

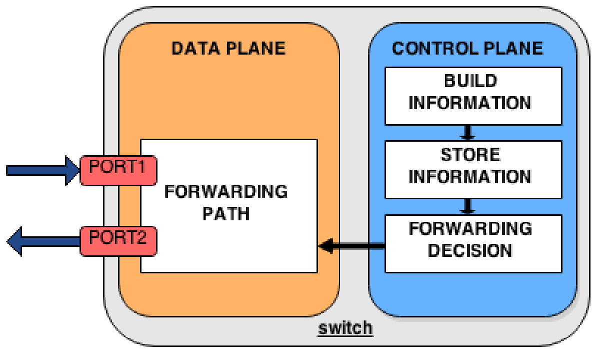

Figure 2.1: "Classical" switch components

A switch consists of multiple ports for incoming and outgoing data. Internal forwarding tables classify the packets and forward them to one or many specific ports. It does so by collecting MAC addresses and storing their corresponding port in specific tables. Layer 2 switches also support the segregation into virtual LANs (VLAN), which enables the network operator to logically isolate networks that share a single switch.

Routers forward packets on the Network layer (Layer 3) and routing-decisions are made based on IP addresses. They contain a routing table where paths to neighbour networks are stored, so that packets can be forwarded to their destination IP address. Other features that can be configured with routers are Quality of Service (QoS), Network Address Translation (NAT) and packet filtering.

The main differences between the classical architecture and SDN will be further described in the coming sections.

2.1.2 Software-Defined Networking concept

SDN represents a new dynamic, manageable, cost-effective and adaptable architecture that is built to serve the dynamic infrastructures that are needed as a backbone for today's data centers. Opposed to the traditional approach, network control and forwarding functions are decoupled and thus can be programmed and divided into different applications and network services. The work of the Open Networking Foundation laid out the OpenFlow protocol as the base for modern SDN solutions.

2.1.3 SDN architecture

SDN separates the architecture into three distinct layers that communicate with each other through different APIs. In figure .. this separation is shown.

- **Infrastructure Layer:** here all physical and virtual devices (e.g. switches and routers) that are capable of the OpenFlow Protocol provide forwarding mechanisms on different Network Layers.
- **Control Layer:** represents the 'network intelligence' and collects global view of the network, by communicating with the switching elements through the so called Southbound API.
- **Application Layer:** consists of business applications that allow the network operator to extend the SDN controller on an abstracted level, without being tied to the actual details of the implementation of the infrastructure. This communication with the Control Layer

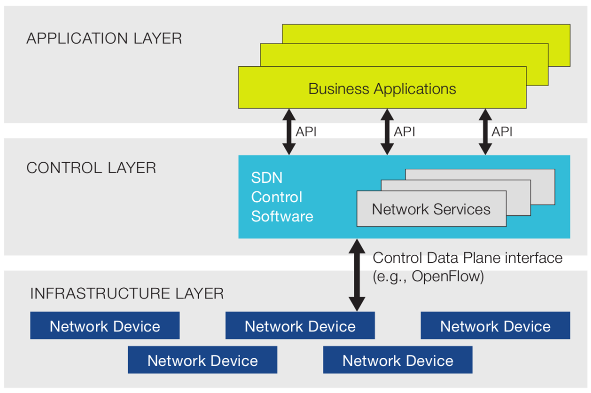

Figure 2.2: Software-Defined Network architecture

2.1.4 OpenFlow

With OpenFlow the Open Networking Foundation defined the first standard communications interface between the SDN architecture's control and forwarding layers. It enables manipulation and direct access to the forwarding plane of physical as well as virtual (hypervisor-based) network devices such as switches and routers.

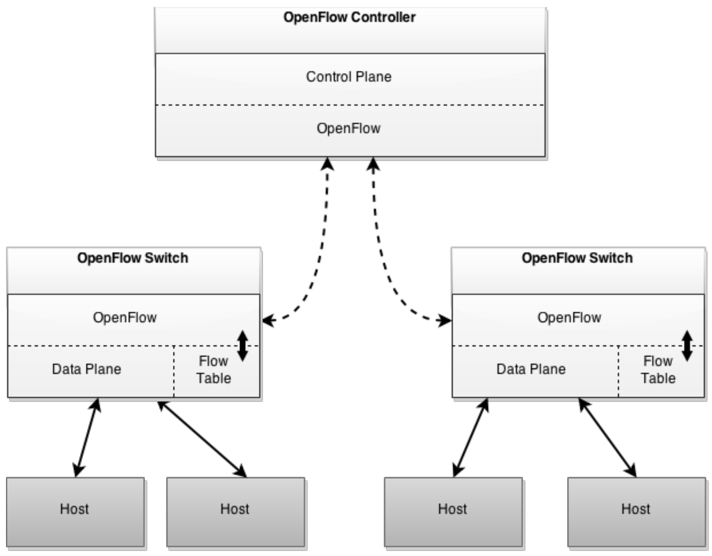

Figure 2.3: OpenFlow Network Architecture

OpenFlow first of all stands for the communications protocol that is used by SDN controllers to fetch information and configure switches. Additionally it is a switch specification that defines its minimum capabilities in order to support OpenFlow.

Most of the OpenFlow-enabled switches and controllers currently still only support the OpenFlow version 1.0 (released in December 2009). The newest version at this date is 1.4, however this explanation of OpenFlow will be focussed on version 1.3 since that is the most recent specification which is supported by OpenVSwitch.

The

Generally the switches are backwards-compatible down to version 1.0. In the following description the focus lies on the required features of all OpenFlow capable devices, however it has to be mentioned that there is also a set of optional features.

Flow Tables

2.1.4.1 OpenFlow Controller

2.1.4.2 OpenFlow Switch

2.1.4.3 OpenFlow Software Switch

OpenVSwitch

2.2 Cloud computing infrastructures

2.2.1 OpenStack

2.2.2 Devstack

2.2.3 OpenStack Nova

2.2.4 OpenStack Heat

2.2.5 OpenStack Neutron

2.3 Conclusion

Chapter 3

Requirements

3.1 Functional requirements

3.2 Non-functional requirements

Chapter 4

State of the art

4.1 Requirements

4.1.1 Functional requirements

4.1.2 Non-functional requirements

4.2 Existing S-O-T-A solutions

4.2.1 OpenDaylight SDN controller

4.2.2 Ryu SDN controller

4.2.3 OpenStack Neutron - QoS Extension

Chapter 5

Design

5.1 Architecture overview

5.2 Design of Connectivity Manager

5.3 Design of Connectivity Manager Agent

5.4 Conclusion

Chapter 6

Implementation

6.1 Environment

6.1.0.1 Software development approach

6.1.0.2 Project structure

6.2 Connectivity Manager components and operations

6.2.0.3 Selection of best-matching Hypervisor

6.2.0.4 Enabling QoS for VM

6.3 OpenStack Neutron configuration

6.4 Conclusion

Chapter 7

Evaluation

7.1 Feature analysis

7.2 Nubomedia use-case

7.3 Conclusion

Chapter 8

Conclusion

8.1 Summary

8.2 Problems encountered

8.3 Future work

Appendix A

List of source codes

Appendix B

Glossar

SDN $\rightarrow$ Software-Defined Networking.

Bibliography

[Abdel-Aziz 71] Y. I. Abdel-Aziz and H. M. Karara, „Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry“, in: Symposium on Close-Range Photogrammetry, issue 11, pp. 1–18, University of Illinois at Urbana-Champaign, 1971.

[AutTech 07] Firma Automation Technology GmbH in 22946 Trittau, Produktübersicht, Downloads und Datenblätter. URL: `http://www.automationtechnology.de`