Information Security

Open Networking

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur

eingereicht von

Daniel Mestan, BSc

Matrikel-Nr.: 1610619521

im Rahmen des
Studiengangs Information Security an der Fachhochschule St. Pölten

Betreuung
Betreuer: Dipl.-Ing Dipl.-Ing Christoph Lang-Muhr, BSc

St. Pölten,
01.06.2018

(Unterschrift Autor)

(Unterschrift Betreuer)

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich sonst keiner unerlaubten Hilfe bedient habe.
- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.
- diese Arbeit mit der vom Begutachter/von der Begutachterin beurteilten Arbeit übereinstimmt.

Der Studierende/Absolvent räumt der FH St. Pölten das Recht ein, die Diplomarbeit für Lehre- und Forschungstätigkeiten zu verwenden und damit zu werben (z.B. bei der Projektevernissage, in Publikationen, auf der Homepage), wobei der Absolvent als Urheber zu nennen ist. Jegliche kommerzielle Verwertung/Nutzung bedarf einer weiteren Vereinbarung zwischen dem Studierenden/Absolventen und der FH St. Pölten.

St. Pölten,
01.06.2018

_____
(Unterschrift Autor)

Zusammenfassung

Seit Jahrzehnten werden im Server-Bereich Bare-Metal Server, also von der Software entkoppelte Geräte, eingesetzt. Durch die Disaggregation der Hardware und Software ist es möglich, das Betriebssystem frei zu wählen und die Hardware an die Anforderungen der Applikationen anzupassen. Im Netzwerk-Bereich gab es diese Trennung bisher nicht, weshalb nur proprietäre Lösungen mit gebündelter Hard- und Software zum Einsatz kamen. Mit Open Networking wird es möglich, Bare-Metal Switches mit einem beliebigen Netzwerkbetriebssystem zu betreiben. Durch die Verwendung von Linux als Basis dieser Betriebssysteme kann die Management-Plane des Switches zusätzlich für andere Zwecke wie Datensammlung oder als Linux-DNS/DHCP-Server benutzt werden. Große IT-Unternehmen und Service-Provider setzen auf Open Networking, da das Management und die Orchestrierung von Netzwerkkomponenten ohne Automatisierung nicht möglich ist. Die Automatisierung wird durch Open Networking gefördert und durch Technologien wie API, DevOps und NETCONF bereitgestellt. Um das Netzwerkmanagement noch mehr zu vereinfachen, werden in Zukunft Software-Agents für eine einheitlichere Konfigurationsschnittstelle sorgen, welche von vielen Netzwerkbetriebssystemen unterstützt werden wird.

Abstract

For years so called bare-metal servers, devices that are decoupled from the software, have been used in the server area. Due to the disaggregation of the hardware and software, the operating system can be freely selected, and the hardware can be adapted to the requirements of the used applications. In the network area, there are only proprietary systems with no such separation of the hardware and software until now. With Open Networking it becomes possible to use bare-metal switches and choose any network operating system the customer wishes. By using Linux as the basis of the operating systems, the switches can also be used for other purposes such as data collection or as a Linux DNS / DHCP server. Large IT-players and service providers rely on Open Networking, the management of network components without automation is not possible anymore for their network. Automation is enabled through open networking and delivered through technologies such as API, DevOps, or NETCONF. To further simplify network management, software agents for a single configuration unit will be used on top of all kind network operating systems in the future.

1. Inhaltsverzeichnis

2. Abbildungsverzeichnis

3. Abkürzungsverzeichnis

AAA ..... Authentification, Authorization Accounting
AD ..... Active Directory
API ..... Application Programming Interface
ASIC ..... Anwendungsspezifische integrierte Schaltung
ASR ..... Aggregated Services Router
BIRD ..... Bird Internet Routing
BPDU ..... Bridge Protocol Data Unit
BRCTL ..... Bridge Control Utility
CLAG ..... Chassis Link Aggregation
CLI ..... Command Line Interface
CPS ..... Control Plane Services
CPU ..... Central Processing Unit
D1IM ..... Dell One Identity Manager
DevOps ..... Development IT-Operations
DIAMETER ..... Twice the RADIUS (Protocol)
DHCP ..... Dynamic Host Configuration Protocol
DOT1X/802.1x ..... IEEE 802.1x Port Based Authentification
DTP ..... Dynamic Trunking Protocol
EAP ..... Extensible Authentication Protocol
ELK ..... Elasticsearch Logstash Kibana
FBOSS ..... Facebook Operating System Software
GbE ..... Gigabit Ethernet
GNS3 ..... Graphical Network Simulator 3
GPIO ..... General Purpose Input Output
HTTP ..... Hypertext-Protocol
I2C ..... Inter-Integrated Circuit
IC ..... Integrated Circuit
ID ..... Identification
IDS ..... Intrusion Detection System
IETF ..... Internet Executive Task Force
IOS ..... Internetwork Operating System
IOS-XR ..... Internetwork Operating System XR
IP ..... Internet Protocol
IPv6 ..... Internet Protocol Version 6
IT ..... Information Technology
JSON ..... JavaScript Object Notation
JunOS ..... Juniper Operating System
L2 ..... Layer 2 (OSI-Model)
L3 ..... Layer 3 (OSI-Modell)
LACP ..... Link Aggregation Control Protocol
LAG ..... ink Aggregation
LDAP ..... Lightweight Directory Access Protocol
LED ..... Light Enabled Diode
MAC ..... Media Access Control
MCLAG ..... Multi Chassis Link Aggregation
MD5 ..... Message-Digest Algorithm 5
MLAG ..... Multi-Link Aggregation

MSTP ..... Multiple Spanning Tree Protocol
NCLU ..... Network Command Line Utility
NETCONF ..... Network Configuration Protocol
NFV ..... Network Functions Virtualization
NOS ..... Network Operating System
NSX ..... VMware NSX
NTP ..... Network Time Protocol
NX-OS ..... Nexus Operating System
OcNOS ..... Open Compute Network Operating System
OCP ..... Open Compute Project
ODM ..... Original Design Manufacturer
OEM ..... Original Equipment Manufacturer
OF-DPA ..... OpenFlow Data Plane Abstraction
ON ..... Open Networking
ONF ..... Open Networking Foundation
ONIE ..... Open Network Installation Environment
ONL ..... Open Network Linux
ONOS ..... Open Network Operating System
OpenNSL ..... Open Network Switch Library
OPX ..... Open Switch OPX
OSPF ..... Open Shortest Path First
OSPFv3 ..... Open Shortest Path First Version 3
OSPFv6 ..... Open Shortest Path First IPv6
OVS ..... Open vSwitch
PC ..... Personal Computer
PicOS ..... Programmable Internetworking and Communication Operating System
PSU ..... Power Supply Unit
PVSTP+ ..... Per VLAN Spanning Tree Protocol Plus
PXE ..... Preboot Execution Environment
RADIUS ..... Remote Authentification Dial-In User Service
REST-API ..... Representational State Transfer Application Programming Interface
RSTP ..... Rapid Spanning Tree Protocol
SAI ..... Switch Abstraction Interface
SDK ..... Software Development Kit
SDN ..... Software Defined Network
sFlow ..... sampled Flow
SFP ..... Small Form-factor Pluggable (Mini-GBIC)
SNMP ..... Simple Network Management Protocol
SNORT ..... Software Network Intrusion Detection & Prevention System
SONiC ..... Software for Open Networking in the Cloud
SPAN ..... Switched Port Analyzer
STP ..... Spanning Tree Protocol
TACACS+ ..... Terminal Access Controller Access Control System Plus
TFTP ..... Trivial File Transfer Protocol
VLAN ..... Virtual Local Area Network
Vtysh ..... Virtual Terminal Line Shell
VXLAN ..... Virtual Extensible Local Area Network
XML ..... Extensible Markup Language
XORP ..... Extensible Open Router Platform
YANG ..... Yet Another Next Generation

ZebOS.....Zebra Operating System
Zebra..... Zebra Routing Software Daemon
ZTP ..... Zero Touch Deployment

4. Einleitung

Zurzeit werden in herkömmlichen Netzwerken ausschließlich proprietäre Komponenten verwendet, welche nur mit der Software des Herstellers und dessen Management-Lösungen kompatibel sind. Das Netzwerkmanagement fällt oft schwer, da diese Systeme eigene Interfaces haben, die erst in Automatisierungstools integriert werden müssen. Um die Flexibilität und Administrierbarkeit dieser Systeme an die Anforderungen des Unternehmens anzupassen und die Wartung sowie den Betrieb der Infrastruktur zu erleichtern, soll Open Networking zur Anwendung kommen.

Open Networking ist die Portierung der Entkoppelung von Hardware und Software aus dem Server-Bereich in die Netzwerkinfrastruktur. Diese Art von Netzwerk-Systemen kommt zurzeit vor allem bei großen IT-Unternehmen und branchenführenden Service-Providern zum Einsatz. Die Technologie bietet den Einsatz einer beliebigen Netzwerkhardware und einer beliebigen Software, um die Infrastruktur und die verwendeten Komponenten an die Anforderungen flexibel anzupassen.

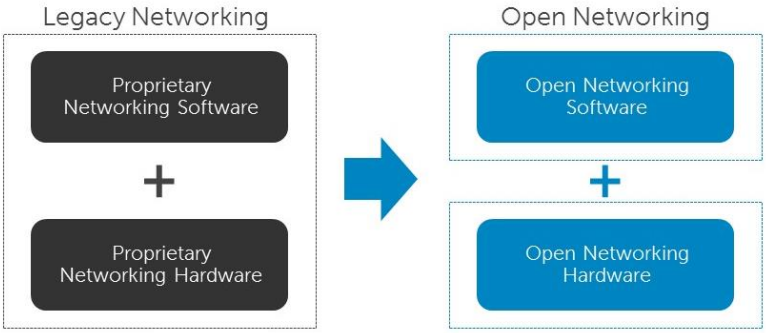

Abbildung 1 - Open Networking Konzept [80]

4.1. Rahmen und Aufbau der Arbeit

Diese Arbeit soll den Grundgedanken von Open Networking näherbringen und den Aufbau eines offenen Systems anhand einer Architektur vorstellen. Diese Architektur wird im Laufe der Arbeit definiert und nachfolgend die einzelnen Bestandteile beschrieben. Zusätzlich soll mithilfe von Begriffserklärungen sowie Vorstellungen der Entwicklungsgruppen und deren Projekte ein Grundwissen geschaffen werden. Der Kostenvorteil von Open Networking Komponenten wird ebenso wie die Netzwerkautomatisierung und Orchestrierung sowie das Zero-Touch-Provisioning erläutert. Der Fokus liegt dabei vorwiegend auf DevOps-Tools und dem ZTP, dessen Vorgänge detailliert erklärt werden.

Anschließend erfolgt eine Vorstellung von bekannten Netzwerkbetriebssystemen für Open Networking Hardware, welche vorwiegend mit dem Testsystem kompatibel sind und deshalb nicht als komplette Auflistung aller NOS dienen soll. Hier werden auch Open-Source-Lösungen und Software-Agents vorgestellt, die eher im nächsten Schritt von Open Networking Anwendung finden werden, jedoch auch in der aktuellen Architektur betrieben werden. Danach wird in einem Hands-On zunächst das Testsystem und die definierten Testszenarien vorgestellt, welche die Anwendung von Open Networking Komponenten in Produktionsumgebungen simulieren sollen. Nach der Vorstellung der Testszenarien werden die Testergebnisse und der Vorgang der Konfiguration vorgestellt und etwaige Probleme beschrieben beziehungsweise Lösungsvorschläge für diese gegeben.

Nach den Tests erfolgt die Beantwortung der Forschungsfragen anhand der erhaltenen Erkenntnisse. Im Anschluss bildet der Ausblick einen allgemeinen Überblick über die weitere Entwicklung von Open Networking und die zukünftigen Entwicklungen wie „Open Networking 2.0“ und die Auswirkungen auf geschlossene Systeme.

4.2. Formulierung der Fragestellungen

Die nachfolgenden Fragestellungen haben sich durch die genaue Analyse beziehungsweise Erörterung der Grundlagen und der Definition von Open Networking ergeben und werden im Rahmen dieser Arbeit beantwortet:

- **Welche Vor- und Nachteile bietet Open Networking und wo gibt es noch Aufholbedarf?**
- **Welche Auswirkungen hat Open Networking auf Netzwerkmanagement?**
- **Wie gut ist die Kompatibilität und Konnektivität zwischen traditionellen Netzwerkbetriebssystemen und Open Networking Netzwerkbetriebssystemen?**
- **Kann Open Networking geschlossene Systeme ablösen und ist ein Einsatz in Produktionssystemen möglich?**

4.3. Vorgehensweise

Für die Beantwortung dieser Fragestellungen werden zunächst die Theorie und die Grundlagen von Open Networking erklärt. Anschließend sollen mithilfe einer Architektur für offene Netzwerkinfrastrukturen die Vorteile dieser Technologie erörtert werden. Um die Auswirkungen auf das Netzwerkmanagement zu klären, wird das Zero-Touch-Provisioning sowie die Automatisierung der Software mit DevOps-Tools erklärt und bei der Vorstellung der Betriebssysteme der Fokus auf die Orchestrierungs- und Netzwerkmanagement-Möglichkeiten gelegt. Um Kompatibilitätsprobleme und die Möglichkeit eines Einsatzes von Open Networking in produktiven Systemen zu klären, werden Testszenarien entwickelt und definiert. Diese werden auf die vorhandenen Ressourcen angepasst, um damit möglichst praxisnahe Technologien und Konzepte zu verwenden. Anhand der Evaluierung der Testergebnisse und der Erkenntnisse der Marktanalyse sowie der Aufarbeitung der Theorie werden dann die Forschungsfragen beantwortet.

5. Open Networking

Dieses Kapitel beschreibt allgemeine sowie technische Grundlagen und Begriffserklärungen zu Open Networking. Ziel ist es, ein Verständnis der Thematik zu schaffen und grundlegende Fragen zu beantworten. Zuerst wird im Allgemeinen beschrieben, was hinter dem Begriff Open Networking steckt. Danach wird ein kurzer Ausblick auf die möglichen Gründe für eine Verwendung und die möglichen Vorteile gegeben. Anschließend wird das Thema SDN kurz erklärt und der Zusammenhang zwischen den beiden Begriffen erklärt. Außerdem werden die aktuellen Projektgruppen und Organisationen, die zurzeit an Open Networking arbeiten, vorgestellt. Anschließend erfolgt eine Vorstellung der für die Disaggregation notwendigen Technologien. Zum Schluss wird das Thema Orchestration und Netzwerkmanagement sowie die Konfigurationsmöglichkeiten von Open Networking-Systemen beschrieben.

5.1. Grundlagen

Unter dem Begriff Open Networking versteht man die herstellerunabhängige Nutzung von Netzwerkkomponenten und das Entkoppeln von Hardware, Software und Management. Durch Open Networking soll es möglich werden, die für einen Anwendungsfall notwendige und am besten geeignete Software auf einer beliebigen skalierbaren Hardware zu betreiben. Durch die Trennung der beiden Bereiche wird es möglich, neue Wege im Netzwerkbereich zu gehen und maßgeschneiderte Komponenten und eine beliebige Software einzusetzen. Im Server-Bereich hat sich diese Trennung schon vor langer Zeit etabliert und zeigt, welche Vorteile sie mit sich bringt. Anstatt Hardware und Software inklusive Support von einem Hersteller zu beziehen, wird es möglich, die gewünschte Software auf einem für die Anforderungen passenden Switch zu installieren. Durch diese Trennung wird es möglich, eine Alternative zu proprietären Netzwerkgeräten zu wählen und die neuesten Technologien einzusetzen. Dadurch können Kosteneffizienz und Erweiterbarkeit sowie das Netzwerkmanagement an die Bedürfnisse des Unternehmens angepasst werden. Gerade der Preis kann bei der Wahl von neuen Netzwerkkomponenten eine wichtige Rolle spielen, wodurch es zur Verwendung von Komponenten verschiedener Hersteller kommen kann. Dies kann zu Problemen führen, wenn ein Hersteller für die komplette Supply-Chain, von der Switch-Hardware über die Switch-Software bis zur Managementlösung, verantwortlich ist. So entsteht möglicherweise ein Wildwuchs an Managementlösungen für verschiedenste Hersteller. Beim Einsatz von unterschiedlichen Netzwerkkomponenten können aber vor allem mit diversen Schnittstellen und dem Netzwerkmanagement Kompatibilitätsprobleme auftreten. Unternehmen sind immer auf der Suche nach Netzwerklösungen, die in der Anschaffung und im Betrieb geringere Kosten ausweisen. Gerade deshalb arbeiten führende Unternehmen aus unterschiedlichen Branchen mit Switch-Herstellern zusammen und erarbeiten Alternativen zu diesem proprietären System. [1] [2]

5.1.1. Architektur

Durch diese Trennung von Hardware und Software ergeben sich für die Architektur der Netzwerkkomponenten Veränderungen der Schichten und Bestandteile. Eine traditionelle Infrastruktur besteht aus eigens für diesen Hersteller angefertigten applikationsspezifischen IC's (ASICs) im Switch, einem nur für diese Hardware passenden Betriebssystem, verschiedensten proprietären Protokollen und nur damit kompatiblen Architekturen und Management-Tools. Open Networking verfolgt hier einen anderen Ansatz, bei welchem sich die Bestandteile eines proprietären Systems zuordnen lassen. Ein Open Networking-System besteht aus einem „off the shelf"-ASIC, also einer genormten Switching-CPU. Dadurch ist es möglich, ähnlich wie bei Servern verschiedene Prozessorhersteller, in diesem Fall ASIC-Hersteller, zu wählen und im Switch zu verbauen. Aufbauend darauf muss eine mit offenen Standards gebaute Hardware verwendet werden, um eine Anpassung von Betriebssystemen einfacher zu gestalten. Das Betriebssystem bildet die nächste Schicht im Modell einer Open Networking-Infrastruktur und muss die verwendete Hardware und ASIC unterstützen. Betriebssysteme für Open

Networking-Komponenten bauen auf einer Linux-Basis auf, welche als Grundbaustein für ein Netzwerkbetriebssystem (NOS) gilt. Dieses besteht wiederum aus verschiedenen, je nach den individuellen Anforderungen, betriebenen Applikationen und Schnittstellen zur Hardware. Für das Netzwerkmanagement und die Automatisierung kann dann eine SDN-Lösung oder ein standardisiertes Management- und Automatisierungstool verwendet werden. [3]

Die untenstehende Abbildung veranschaulicht das Modell und basiert wie der obige Absatz auf einer Grafik in einem Blogeintrag von DellEMC zum Thema Open Networking. [3] Die einzelnen Schichten der Grafik werden nachfolgend genauer behandelt.

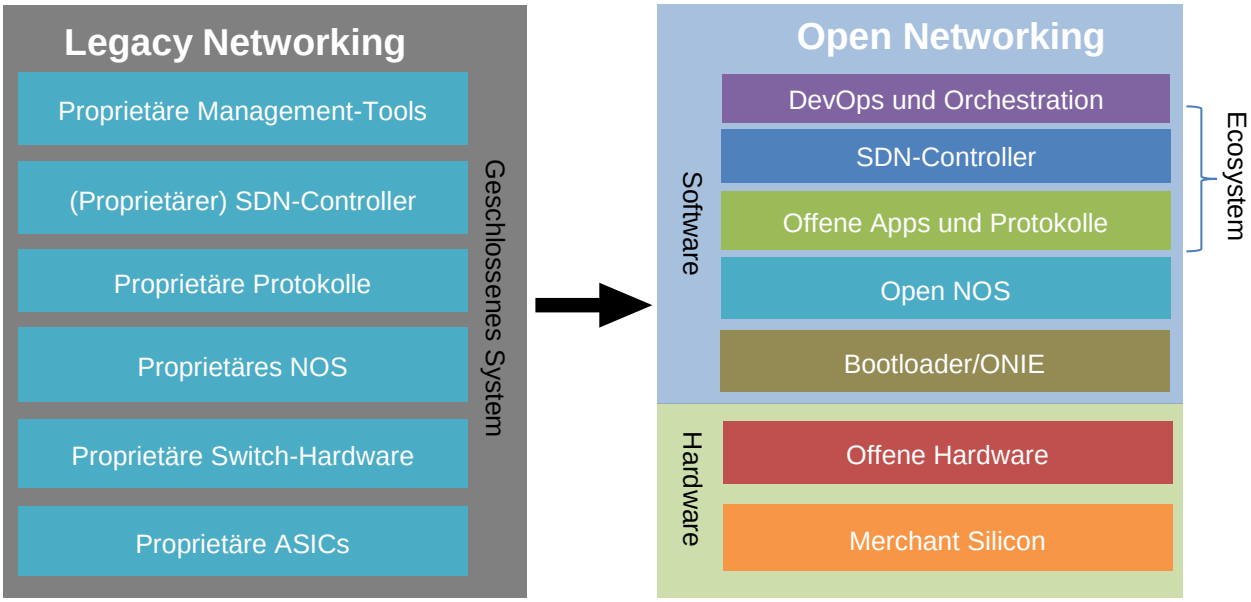

Abbildung 2 – Open Networking Architektur Vergleich

5.1.2. Open Networking Foundation

Ein wichtiges Ziel von Open Networking ist Innovation, weshalb aber auch neue Technologien erfunden werden müssen. Gerade für die im vorigen Modell vorgestellten Schichten ist es notwendig, Projekte und neue Möglichkeiten zu schaffen. Um diese Entwicklungen unter einem Dach zu vereinen und eine zentrale Steuerung und Koordination aller Projekte im Netzwerkbereich umzusetzen, hat das Open-Compute-Projekt (OCP) die Open Networking Foundation (ONF) gegründet. Sie ist eine Non-Profit-Organisation und hat das Ziel, die Entwicklung von Open Networking voranzutreiben und koordinierend bzw. unterstützend tätig zu sein. Die ONF besteht aus vielen branchenführenden Unternehmen und deren Forschungsabteilungen und ist auch für das Protokoll OpenFlow verantwortlich. Es gibt zu fast jeder Schicht der Architektur eines Open Networking-Systems Projekte, die von der ONF betreut werden. Ziel ist, ein allumfassendes Open-Source-Paket zu erstellen und damit die größtmögliche Flexibilität zu gewährleisten. Diese Projekte werden in der späteren Beschreibung der Architekturschichten vorgestellt. [4]

5.1.3. Begriffserklärung

Um einen Überblick über die Begriffe und Bezeichnungen der verschiedenen Switchtypen im Open Networking zu erhalten, wurde ein Blogeintrag von Jeff Doyle (Zeitschrift Network World) herangezogen. In diesem Beitrag wird versucht, in einfachen Worten die Terminologie rund um Open Switching zu erklären, weshalb er als Grundlage für die nachfolgende Erklärung verwendet wurde. [5]

Blackbox Switch

Diese Bezeichnung bedeutet, dass ein Switch als geschlossenes System gekauft wird. Sowohl die Hardware als auch die Software sind vom Switch-Hersteller vorgegeben und es besteht keine Möglichkeit, das Betriebssystem zu ändern. Meistens agiert der Hersteller hier als OEM, stellt also das Gerät in der eigenen Fabrik her und setzt auf proprietäre Schnittstellen. Die Software ist nicht von der Hardware trennbar und auch der Support muss über denselben Hersteller bezogen werden. Dadurch ist man an einen Hersteller gebunden und bezieht alle Services aus einer Hand. Dazu zählen meistens auch Management-Tools, welche nur mit der Software des Herstellers kompatibel sind. Oftmals sind sogar die Entwicklungen desselben Herstellers nicht untereinander kompatibel und so müssen für veraltete Systeme wieder andere Tools verwendet werden. So gesehen bietet ein Blackbox Switch wenig Flexibilität und erschwert die Schnittstellen und Kompatibilität zu anderen Produkten und Herstellern. Zu den bekanntesten Herstellern von Blackbox Switches zählen unter anderem alle traditionellen Netzwerkkomponentenhersteller wie Cisco, Dell, D-Link oder HP.

Bare-Metal Switch

Der Begriff kommt aus dem Server-Bereich, wo seit vielen Jahren schon, nur das pure Metall, also das Gerät ohne Software, gekauft wird. Man wählt die für einen Anwendungsfall passende Server-Hardware und wählt, abhängig vom Einsatzgebiet, das passende Betriebssystem und die Applikationen aus. Ziel dabei ist, eine möglichst gute Abstimmung aus Soft- und Hardware für den Anwendungsfall zu erhalten und dadurch kosteneffizient zu sein. Dieser Ansatz wird nun auch im Netzwerk-Bereich verfolgt, um diese Vorteile auch hier zu nutzen. Die meisten Hersteller von Bare-Metal Switches sind ODMs (Original Design Manufacturers) von Firmen wie Dell, Avaya, Netgear und anderen. Sie stellen also selbst entwickelte Produkte für andere Unternehmen her, welche ihr geschlossenes System darauf installieren und den Switch als Black-Box vertreiben. Hersteller von Bare-Metal Switches sind zum Beispiel Accton, Alpha Networks und Quanta QCT. Diese bieten denselben Switch, der für eine der oben genannten Firmen hergestellt wird, unter eigener Marke an und statten ihn mit einem Bootloader und einer Installationsumgebung für den Gebrauch als Bare-Metal Switch aus. Ein vorinstalliertes Betriebssystem ist auf einem Bare-Metal Switch nicht vorhanden, da man hier nur die Hardware kauft. Durch den fehlenden oder nur geringen Support bei dieser Art von Switch kann es jedoch für normal große Rechenzentren nicht immer nur von Vorteil sein, diese Lösung zu wählen. [5]

Whitebox Switch

Unter dieser Bezeichnung versteht man einen Bare-Metal Switch, der bereits mit einem Netzwerkbetriebssystem ausgestattet wurde und im Bundle verkauft wird. Der große Unterschied zu einer Black-Box besteht aber darin, dass die Hardware und die Software nicht gekoppelt sind und somit ein beliebiges NOS installiert werden kann. Ein Beispiel wäre Pica8, ein Hersteller, bei welchem die Switches direkt mit dem hauseigenen PicOS ausgeliefert werden. Diese Whitebox Switches haben aber ebenfalls einen Bootloader und eine Installationsumgebung vorinstalliert, mit deren Hilfe jedes andere unterstützte NOS installiert werden kann. [6] Auch Dell stellt Whitebox Switches her. Hierbei handelt es sich um die ON-Serie (Open Networking), welche mit dem hauseigenen OS10 ausgeliefert wird. Durch die Installationsumgebung im Bootloader der Switches dieser Serie ist es ebenfalls möglich, die Installation eines der vielen unterstützten NOS durchzuführen. [7] [8] Ein anderes Beispiel sind Switches der Firma Edgecore-Networks, wo Cumulus Network Linux vorinstalliert wird. Ein Whitebox Switch ist also ein Bare-Metal Switch, welcher mit einem vorinstallierten NOS verkauft wird und über die Möglichkeit verfügt, das Betriebssystem zu ersetzen. Dadurch ergibt sich auch oft der Vorteil, dass ein gut ausgebauter Support zur Hard- und Software zur Verfügung steht, weshalb Whitebox oder Britebox Switches für normal große Unternehmen die bessere Wahl sind. [5]

Britebox Switch

Der Begriff steht für „Branded Whitebox“ und ist im Grunde ein Bare-Metal Switch, der von einem ODM für andere Unternehmen hergestellt wird. Oft wird derselbe Switch vom Hersteller selbst vermarktet und als Bare-Metal Switch verkauft. Im Falle eines Britebox Switch wird jedoch die Hardware mit dem Logo der Marke, unter der dieser Switch verkauft wird, versehen und unter einem anderen Namen verkauft. Es handelt sich hierbei um denselben Switch, welcher deshalb auch die Möglichkeit bietet, das Betriebssystem zu wechseln. Keine Britebox Switches sind Geräte von Dells ON-Serie, wo Dell als OEM (Original Equipment Manufacturer) agiert und die Switches selbst herstellt. [5] [7]

Open Switch

Unter dieser Bezeichnung versteht man ein vom Open Compute Project (OCP) akzeptiertes Design einer Switch-Architektur. Der hardwareseitige Aufbau des Switches wird als Referenzdesign herangezogen und die verschiedenen, vom OCP und der ONF gesteuerten, Projekte darauf abgestimmt. Die meisten Bare-Metal Switches sind von der OCP als Open Switch bestätigt und mit den meisten Projekten kompatibel. [9] Auf der Wiki-Seite von Open Compute können alle Spezifikationen und Designs der Open Switches heruntergeladen werden. Dadurch ist es für andere Hersteller möglich, Switches zu bauen, die auf bereits etablierten Referenzdesigns basieren und bereits von vielen Projekten unterstützt werden. Dadurch wird eine gewisse Norm und Standardisierung in die Switching-Platform gebracht und die Anpassung der Software wird einfacher. Die nachfolgende Abbildung zeigt das Blockdiagramm von Facebooks Wedge 100 aus der vom OCP veröffentlichten Hardware-Spezifikation. [10]

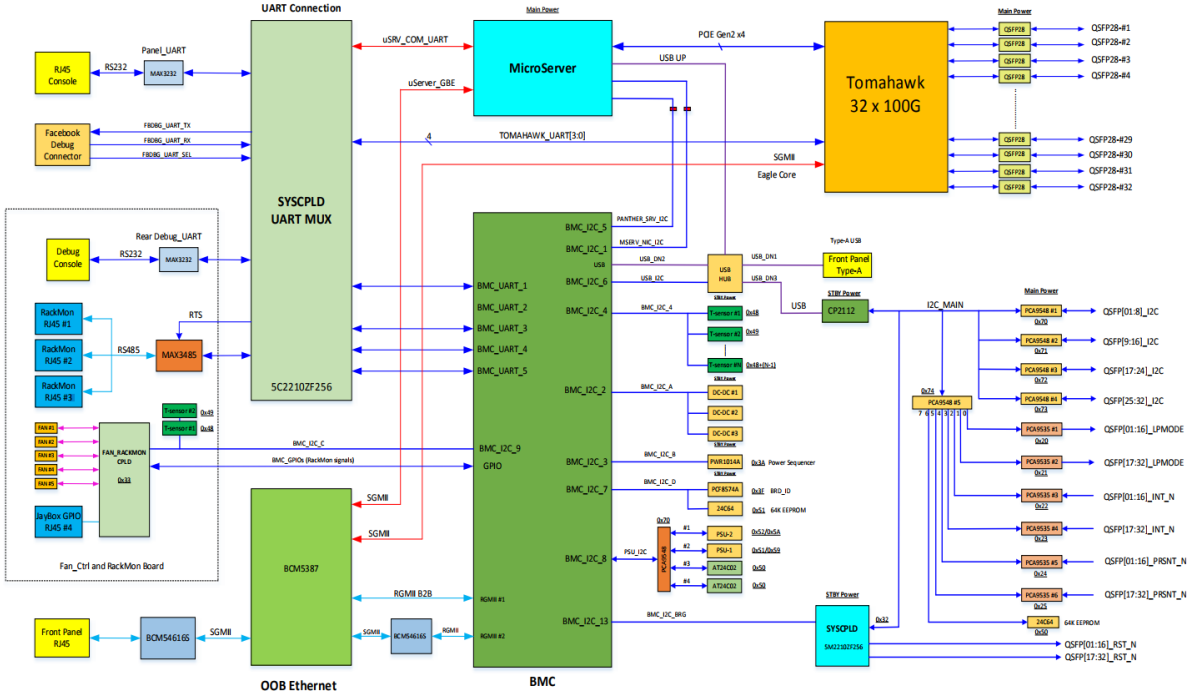

Abbildung 3 - Facebook Wedge 100 Block Diagramm [10, pp. 8 - Figure 1]

Das beste Beispiel für eine standardisierte Hardware ist der Open Switch Facebook Wedge 100, ein 32x100G Top-of-the-Rack-Switch. [11] Dieser Switch ist nicht die einzige von Facebook entwickele Hardwarekomponente. Für seine Rechenzentren entwickelt das Unternehmen eigene Switches und legt die Architektur und das Design mithilfe von OCP offen. Im Jahr 2015 hat Facebook zum ersten Mal einen Switch entworfen, den Facebook Wedge, einen Top-of-the-Rack-Switch mit 40G-Ports zum

Einsatz im eigenen Datacenter. Dieser Switch und das selbst entwickelte, darauf abgestimmte FBOSS haben sich in den Rechenzentren von Facebook etabliert und dazu geführt, dass hier weiterentwickelt wurde. [12] So wurde dann der Wedge 6-Pack, der erste modulare Open Switch entworfen, welcher auf der Basis des ersten Facebook Wedge basiert. [13] Mittlerweile hat sich um die Open Switches von Facebook ein Ecosystem und eine breite Nutzerbasis entwickelt. Nicht nur das Unternehmen selbst hat Interesse an den selbst entwickelten Switches, weshalb vor Kurzem eine Kooperation mit Edgecore-Networks eingegangen wurde. Diese zielt darauf ab mit der offenen Architektur den Wedge 100 Switch herzustellen und als Bare-Metal Switch oder als Whitebox Switch zu vertreiben. [11] [10]
Die nachfolgende Abbildung zeigt das Blockdiagramm von Facebooks Wedge 100 aus der vom OCP veröffentlichten Hardware-Spezifikation. [10]

5.1.4. Kostenfaktor

Der große Vorteil von Open Networking Switches ist der Preis, da hier nur noch die Hardware gekauft wird und weiters noch Lizenzgebühren für das Gerät selbst und das Betriebssystem entfallen. So rechnet zum Beispiel Cumulus Networks, ein Hersteller eines Netzwerkbetriebssystems, die Kosten für eine Switching-Infrastruktur für 200 Server in 10 Racks vor. Diese Topologie enthält 2 Spine-Switches mit 32x40G-Ports und 10 Leaf-Switches (Top of the Rack) mit jeweils 48x10G und 6x40G-Ports. Hier wird ein Quanta T3038-LY8-Switch als Leaf und ein Quanta T5032-LY6 als Spine-Switch verwendet. Um einen Vergleich zu ziehen, wurde mit der Preisliste von Cisco eine äquivalente Lösung mit Nexus 9396PX als Leaf-Switch und Nexus9332PQ als Spine-Switch gewählt. Zusätzlich wurde zum Vergleich noch eine Lösung mit Arista 7050SX-72 als Leaf-Switch und Arista 7050QX-32 als Spine-Switch errechnet. Der Preisvorteil bei der Benutzung von Bare-Metal Switches ist enorm, vor allem wenn man beachtet, dass es sich hier nur um den Preis der Hardware handelt. Nachfolgend ist die Berechnung von Cumulus zur Veranschaulichung abgebildet. [14]

BARE METAL VS LEGACY SWITCH COMPARISON			
CAPEX for 200 nodes in 10 racks	**Bare Metal Switch List Price***	**Cisco List Price****	**Arista List Price*****
Leaves & Spines	$66,980	$432,000	$439,028
Cables & Interconnect	$29,199	$323,600	$202,000
Total	$96,179	$755,600	$641,028

Abbildung 4 - Bare metal vs legacy switch comparison [14, p. 3]

Die Kostenersparnis steigt mit der Größe des Netzwerks an, weshalb es gerade für große IT-Unternehmen von Vorteil sein kann, einen Umstieg auf Open Networking Switches, sei es Bare-Metal, Whitebox oder Britebox, durchzuführen. Zu beachten ist jedoch, dass in dieser Rechnung keine Lizenzgebühren und Anschaffungskosten für die Software enthalten sind. Hier geht es ausschließlich um die verwendete Switching-Hardware und die Verkabelung. Jedoch sind gerade die Lizenzkosten bei traditionellen Netzwerkherstellern wie Cisco oft höher als die Kosten der eigentlichen Hardware. Fakt ist, dass durch den Kauf von Open Networking Switches ein klarer Preisvorteil entsteht, der auch nach dem Kauf der Lizenzen bzw. Software und im Betrieb weiterhin bestehen bleibt.

5.2. Hardware

Pica8, ein Hersteller von Switches und PicOS beschreibt in einem Whitepaper, welche Komponenten und Schnittstellen notwendig sind, um ein Betriebssystem überhaupt erst auf ein neues Gerät zu portieren. Die Grundvoraussetzungen sind laut Pica8 ein offenes ASIC und eine Möglichkeit zur Steuerung des Chassis. [15]

5.2.1. Merchant Silicon

Eine Anwendungsspezifische integrierte Schaltung (ASIC) ist ein für eine bestimmte Anwendung entworfener integrierter Schaltkreis (IC). Diese Chips werden meist mit vorgefertigten Logiken ausgestattet, die dann je nach Aufgabengebiet und Kundenanforderungen miteinander verknüpft werden. Dadurch sinken die Planungs- und Herstellungskosten und die Komplexität der Schaltung nimmt ab. Außerdem müssen bestimmte Logiken und Schaltungen, die für den Anwendungsfall nicht benötigt werden, nicht den Weg auf den Chip finden. Dadurch wird der Chip den Anforderungen des Kunden gemäß maßgeschneidert und ist für die jeweilige Applikation schneller als herkömmliche ICs, die für ein breites Feld an Aufgaben entworfen wurden. In den meisten Fällen erfolgt die Fertigung von ASICs in kleinen Serien, da sie normalerweise nur auf Kundenwunsch hergestellt werden. [16]
Diese Art von integrierten Schaltkreisen kommt seit einigen Jahren auch in Datacenter-Switches zum Einsatz, da es in diesem Bereich auf einen hohen Durchsatz des Netzwerkverkehrs ankommt. Moderne Enterprise Switches verwenden eine CPU für die Control-Plane, also für das Management und die Konfiguration des Geräts und des Umfelds, während die ASIC das Herz der Switching-Engine ist. [15, p. 4] So ist die CPU zum Beispiel für das Software-Handling von Netzwerkprotokollen wie Spanning-Tree oder OSPF verantwortlich. Die ASIC selbst ist kein kompletter Switch und kann die Aufgaben der CPU nicht übernehmen Um die große Bandbreite und den immensen Datendurchsatz zu garantieren, sind normale CPUs, welche beim Entwurf für viele Anwendungen optimiert wurden, nicht geeignet. Hier kommen ASICs zum Einsatz, welche für diese Zwecke entworfen wurden. Diese sind für die Switching-Logik verantwortlich und für diese Aufgabe besser geeignet, da die Optimierung für diesen Anwendungsfall vorgenommen wurde. [17] [18, pp. 3,4]

Bei traditionellen Blackbox Switches kommt meist eine proprietäre ASIC zum Einsatz, welche auf die Wünsche des Switch-Herstellers angepasst wurde. Die genaue Funktionsweise, die Schnittstellen (APIs) und der Bauplan sind nur dem Hersteller bekannt, weshalb es auch nicht möglich ist, auf Blackbox Switches eine nicht-proprietäre Software zu installieren. Die API der ASIC bzw. das SDK dient dazu, die ASIC zu konfigurieren und ihre Funktion zu gewährleisten. Ein VLAN wird zum Beispiel mithilfe der Schnittstelle zur ASIC angelegt und konfiguriert. [15]

5.2.2. Standardisierte Platform

Der Grundstein für den Betrieb offener Software auf einem Switch sind neben einem merchant ASIC und dessen Schnittstellen auch standardisierte Switch-Plattformen. Unter der Switching-Plattform wird das Switch-Chassis inklusive aller festen und modularen Einheiten, abgesehen vom ASIC, verstanden. Dabei handelt es sich unter anderem um die Sensoren, Lüftereinschübe, Netzteile (PSU), das Mainboard, die Switchports, SFP-Module und LEDs. Um eine Steuerung dieser Komponenten zu ermöglichen, muss das Chassis über offene und standardisierte Schnittstellen angesprochen werden. Bei allen Arten von Open Networking Switches wurden diese Schnittstellen bereits beim Design auf die Software angepasst und bieten so eine optimale Unterstützung der offenen Switch-Software. Ohne die Möglichkeit der Steuerung der Plattform wäre ein Betrieb des Switches nicht möglich, da diese essentiellen Bestandteile nicht angesprochen werden können und ein Schaden an der Hardware verursacht werden würde. Bei Blackbox Switches gibt es keine Dokumentation und Möglichkeit, die Schnittstellen der Plattform zu benutzen und eine Disaggregation von Hardware und Software ist nicht

möglich. Dies gilt auch dann, wenn in einem Switch ein ASIC mit offenen Schnittstellen verbaut wurde, aber die Plattform ein geschlossenes System ist. Da in allen Open Networking Switches eine klassische x86-CPU verbaut ist, welche für die Kontroll-Einheit (Control-Plane) verantwortlich ist, dient Linux als Grundlage der Software. Dies liegt an der guten Kompatibilität von Linux und den vorhandenen Treibern.

5.3. Software

5.3.1. Bootloader und ONIE

Das Open Network Install Environment (ONIE) ist eine offene Installationsumgebung für Open Networking Hardware, durch welche es möglich wird, ein beliebiges Betriebssystem auf einem offenen Switch zu installieren. Diese Installationsumgebung ist ein Projekt des Open Compute Projects (OCP) und wurde 2012 von Cumulus Networks entwickelt. Es handelt sich dabei um ein minimalistisches Betriebssystem, welches als Firmware auf Open Networking Switches zum Einsatz kommt. Es dient dazu, ein beliebiges NOS auf einem Switch zu installieren und kann auch für die automatische Provisionierung verwendet werden. [19] ONIE hat sich als Firmware und für die Installation von NOS durchgesetzt und ist der Industriestandard für Open Networking Switches. Moderne Switches bieten wie oben beschrieben eine eigene CPU für das Management der Plattform und die Peripherie, welche unabhängig vom ASIC arbeitet und nicht für das Switching zuständig ist. ONIE läuft auf diesem Management-Subsystem und bietet eine minimalistische Linux-Umgebung für die Provisionierung und Installation der Software, wie man sie aus der Server-Umgebung kennt (PXE) und besteht aus einer Kombination von Bootloader und Linux-Kernel. Für die Beschreibung der Architektur und der Abläufe im ONIE wurde die Dokumentation vom OpenComputeProject (OCP), dem Entwickler von ONIE, verwendet. [20]

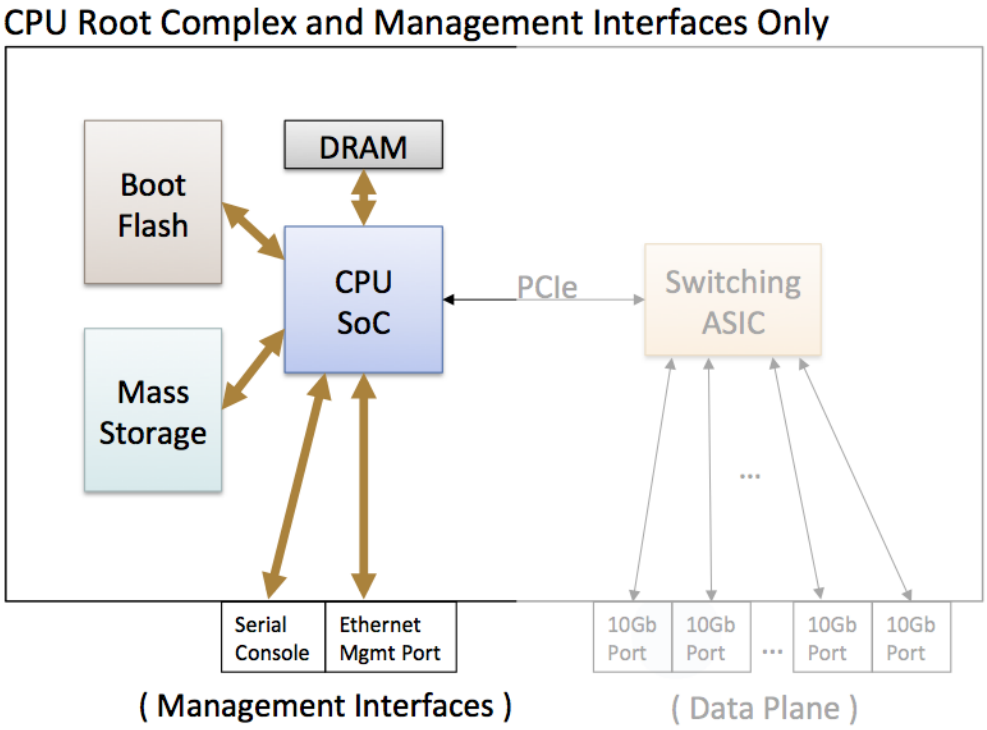

Abbildung 5 - Switch Hardware Design [20]

ONIE verwendet nur die Plattform-CPU, welche für die Verwaltung aller Komponenten der Plattform zuständig ist, und bietet keinen Zugriff auf die Switching ASIC, welche für die Steuerung der Data-Plane verantwortlich ist. Die vorherige Abbildung hebt die wichtigsten von ONIE verwendeten Komponenten hervor und zeigt, welche Schnittstellen verwendet werden. [20]

Der Ablauf beim ersten Start eines Open Networking Switches mit installiertem ONIE erfolgt immer nach einem definierten Prozess. Zuerst wird der Bootloader des Switches geladen, welcher herstellerabhängig, aber an offene Standards gebunden, ist. Der Bootloader lädt das ONIE-Image vom Flash-Speicher des Switches startet den Ladevorgang. Danach erfolgt die Initialisierung von ONIE und der Plattform inklusive des Management-Ports, wodurch eine Netzwerkkommunikation möglich wird. Nun steht ein Zugriff über die Kommandozeile zur Verfügung, über die einzelne Vorgänge gesteuert werden können. Im Normalfall sucht das ONIE nach verfügbaren Abbildern von Netzwerkbetriebssystemen und startet gegebenenfalls die Installation des Images. Der Installationsprozess des NOS wird anschließend gestartet und das Betriebssystem auf den Massenspeicher des Switches gespeichert. Abschließend speichert ONIE die Konfiguration und ändert den Bootvorgang so ab, dass das installierte Betriebssystem beim nächsten Neustart des Switches gestartet wird. [21] [22]

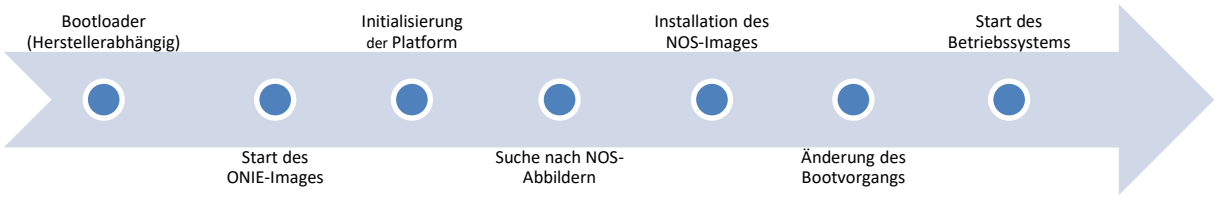

Abbildung 6 - Erster Systemstart ONIE

Nach einer erfolgreichen Installation eines NOS wird ONIE nicht vom System entfernt, sondern verbleibt im Flash-Speicher des Switches und wird nur beim Systemstart übersprungen. Durch die Änderung des Bootvorgangs ignoriert der Switch das ONIE-Image und startet direkt das installierte Betriebssystem. Um wieder in das ONIE zu starten, gibt es in allen NOS die Möglichkeit, durch eine vorher definierte API den Bootvorgang wieder so abzuändern, dass die Installationsumgebung gestartet wird und sich das System wie bei einem initialen Start verhält. Außerdem bietet ONIE die Möglichkeit, noch vor dem Start des Betriebssystems in das ONIE zu wechseln und den Start des NOS zu verhindern. In dieser frühen Phase des Systemstarts bietet das ONIE folgende Möglichkeiten [22]:

- Reinstall NOS – Neuinstallation des Betriebssystems
- Uninstall NOS – Deinstallation aller installierter NOS, Wipe aller Daten am Switch, außer ONIE
- Rescue – Neustart des Switches und direkter Boot in das ONIE
- Update – Installation einer neuen Version des ONIE

Die Installationsumgebung bietet verschiedene Varianten zur Installation eines NOS-Abbildes. Laut ONIE-Dokumentation gibt es unter anderem folgende Möglichkeiten zur Lokalisation und Installation eines Abbildes [20]:

- Statisch im Bootloader festgelegtes Abbild
- Angeschlossenes Speichermedium, z.B.: USB-Stick
- DHCPv4/DHCPv6

- IPv4 und IPv6 Nachbarn
- mDNS / DNS-SD
- PXE-like TFTP
- http

Die nachfolgende Grafik stammt aus der ONIE-Dokumentation der OCP und stellt die einzelnen Lokalisationsmöglichkeiten und den Ablauf in einem Entscheidungsbaum dar. [20]

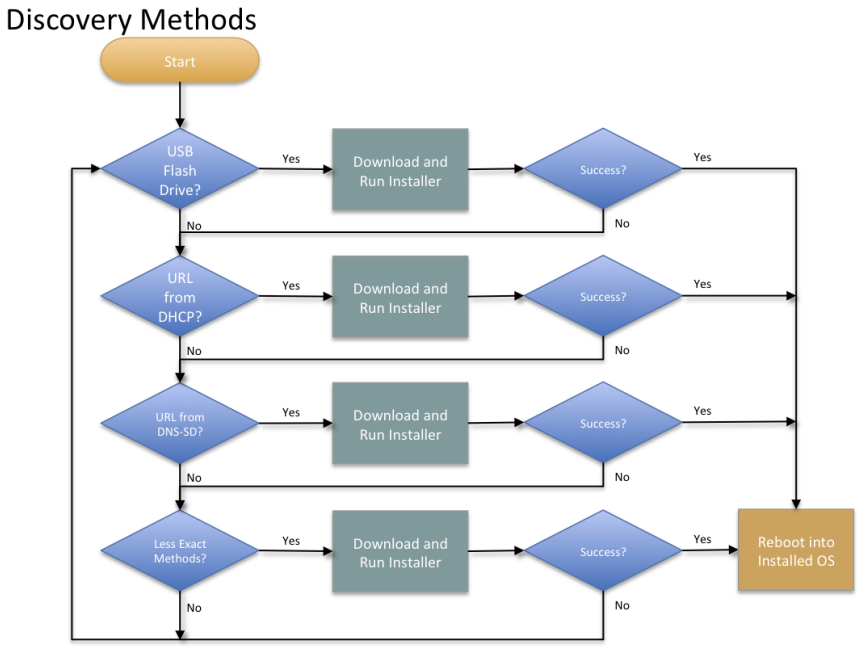

Abbildung 7 - ONIE Discovery Entscheidungsbaum [20]

OCP empfiehlt für den Image-Download und die Installation einen http-Server als Datenquelle zu verwenden, da dieses Protokoll auch für die Übertragung großer Images geeignet ist. Sobald ONIE automatisch ein Abbild eines Betriebssystems findet, wird der Installationsprozess gestartet. Dieser Vorgang kann auch per Hand gestoppt und selbst vorgenommen werden, was bei der Installation einer speziellen Version für Testzwecke hilfreich ist. Der Autodiscovery-Modus lässt sich mit dem Befehl „onie-discovery-stop“ deaktivieren, während man die manuelle Installation eines NOS-Abbilds mit „onie-nos-install <IMAGENAME>“ starten kann. [20] [21]

5.3.2. Open-NOS

Mithilfe des ONIE kann ein beliebiges Netzwerkbetriebssystem, welches mit einem Open Switch kompatibel ist, installiert werden. Da die Hardware eines Switches nicht mit der eines Servers vergleichbar ist, können Betriebssysteme, die für Server oder ähnliche Hardware erstellt wurden, nicht ohne Probleme für Switches verwendet werden. Die Plattform eines Switches enthält verschiedene Bestandteile wie GPIOs, I2C-Buse, Sensoren und LEDs, die zwar jenen in Servern ähnlich sind, meist aber erweitert wurden und deshalb von einem normalen Betriebssystem für Server nicht angesprochen werden können. Aus diesem Grund ist es notwendig, eine grundlegende Basis für ein Netzwerkbetriebssystem zu schaffen und die notwendigen Treiber zu programmieren. Dieser Vorgang der Portierung ist bei jedem Switch unterschiedlich und kann deshalb auf jeder Hardware zu unterschiedlichen Problemen führen. Gerade deshalb werden vom OCP Switch-Referenz-Designs veröffentlicht und für viele Bare-Metal-Switches verwendet. Aufgrund der guten Kompatibilität setzen alle Hersteller von NOS auf Linux als Basisbetriebssystem für ihre Software-Lösungen und verwenden

einen nicht modifizierten Linux-Kernel als Grundbaustein. Die Portierung auf einen neuen Switch und Anpassung der Treiber ist danach immer noch ein aufwendiges Unterfangen, weshalb verschiedene NOS-Hersteller Arbeitsgruppen bilden und Projekte, die die Erweiterung der Kompatibilität als Ziel haben, starten. Einige Betriebssysteme werden im nachfolgenden Kapitel näher vorgestellt, während Open Network Linux (ONL) vom Open Compute Project an dieser Stelle genauer beschrieben wird, da es als Grundlage für ein NOS dienen soll und somit eine Sonderstellung einnimmt.

ONL ist ein Framework, mit dessen Hilfe man die Control-Plane, also die Steuerungseinheit des Switches, selbst programmieren kann. Das Framework wurde von Cumulus Networks und Big Switch Networks, die beide Teil des Open Compute Projects sind, vorgestellt und dient als Grundlage für alle weiteren Bausteine eines NOS. Es enthält deshalb nur einen beispielhaften Programmcode für die Paket-Weiterleitung und die Programmierung der Schnittstelle zum ASIC. Die Ziele von ONL sind, eine Entwicklungsplattform für Open Networking Entwickler und das OCP zu sein und als Grundlage kommerzieller Betriebssysteme für offene Hardware zu dienen. Die verantwortlichen Unternehmen, Cumulus Networks und Big Switch Networks, benutzen für ihre NOS ONL als Basis. Um das Paket-Forwarding der ASIC zu programmieren, ist ein SDK notwendig, welche verschiedene Hersteller als Open-Source zur Verfügung stellen. Wichtig bei der Entwicklung eigener Packet-Forwarding-Applikationen ist die Auswahl des richtigen SDKs für das im Switch verbaute ASIC. Beispiele für ASIC-SDKs sind das OF-DPA (OpenFlow – Data Plane Abstraction) [23] und OpenNSL (Open Network Switch Layer) [24] von Broadcom, deren Code auf Github veröffentlicht wurde. Der Betrieb von ONL allein als Netzwerkbetriebssystem ist nicht möglich, da es sich hier um kein vollwertiges NOS, sondern um eine Entwicklungsumgebung oder ein Basis-Betriebssystem handelt. [25]
Einen ähnlichen Ansatz verfolgt Dell mit OpenSwitch OPX Base, wobei es sich hier aber um ein vollwertiges Netzwerkbetriebssystem handelt. Dieses wird im nachfolgenden Kapitel genauer beschrieben.

5.3.3. Ecosystem

Auf einem NOS werden verschiedene Open-Source- oder Closed-Source-Tools ausgeführt, die den netzwerktechnischen Betrieb des Switches und die Verwendung verschiedener Netzwerkprotokolle ermöglichen. Hersteller von kommerziellen NOS verwenden für ihre Projekte oft eine Mischung aus offener und nicht-öffentlich-zugänglicher Software. Es gibt bereits eine große Anzahl an Open-Source-Tools für Switches und große Sammlungen von Apps, die zusammenspielen. Dabei muss es sich nicht immer um ausschließliche Netzwerktools handeln, da die hohe Flexibilität und die Möglichkeit der Installation von Linux-Tools am Switch große Vorteile von Open Networking sind.

Die meisten Betriebssysteme bieten bereits eine eigene Sammlung von Apps an, also ein Ecosystem, an welches man aber nicht gebunden ist. Aufgrund des offenen Designs und der Verwendung von Linux als Betriebssystem kann jederzeit eine andere Anwendung oder ein anderes Projekt für den jeweiligen Anwendungsfall verwendet werden. Die Applikationen eines Open Networking Ecosystems lassen sich in verschiedene Kategorien unterteilen. Die Kategorien werden nachfolgend mit Beispielen (Tools) erklärt und beschrieben, wobei das Cumulus Networks Ecosystem als Grundlage dient. [26]

Routing / L3-Stack

Der Layer3-Stack dient dazu, verschiedene Routingprotokolle auf dem Switch zu betreiben und eine Konfiguration zu ermöglichen. Durch den Einsatz einer oder mehrerer Routing-Dienste wird die Layer3-Funktionalität des Switches und ein Betrieb als Router ermöglicht. Diese Applikationen wurden ursprünglich für Linux- oder BSD-Server entwickelt, können aber durch die Grundidee von Open Networking ebenfalls auf einem Switch installiert werden. Bekannte Routing-Dienste, die auch auf Switches zum Einsatz kommen, sind Quagga, BIRD (Internet Routing Daemon) oder FRRouting. Alle

L3-Stacks sind Open-Source-Projekte, werden laufend weiterentwickelt und verwenden die gleiche Kernel-Schnittstelle (Zebra). FRRouting ist ein Projekt der Linux Foundation und kommt unter anderem in den Betriebssystemen von BigSwitch Networks und Cumulus Networks zum Einsatz.

Monitoring / Logging

Um das Monitoring des Switches zu ermöglichen, kommen verschiedene Technologien und Programme zum Einsatz. Da in den meisten Fällen Linux als Grundlage dient, wird für Logging Syslog verwendet. Um SNMP-Abfragen zu ermöglichen, steht entweder snmpd oder net-snmp zur Verfügung. Für das Monitoring des Netzwerkverkehrs und der Anomalieerkennung können sFlow oder NetFlow verwendet werden. Alle gängigen Open Networking Betriebssysteme bieten eine Möglichkeit von NetFlow- oder sFlow-Daten. Der Vorteil liegt darin, dass nicht der komplette Datenverkehr aufgezeichnet wird, sondern lediglich Metadaten, wie Sender, Empfänger, Protokoll und Port, an den Flow-Collector gesendet werden müssen. In manchen Fällen kann aber für eine Analyse des Netzwerkverkehrs auch ein SPAN- oder Mirror-Port notwendig sein, dies ist zum Beispiel für die Implementierung von Intrusion Detection Systemen notwendig. Theoretisch könnte sogar SNORT, eine Open-Source IDS-Software auf dem Switch installiert werden. Manche Open Networking Betriebssysteme betreiben mit ihrem Ecosystem aber auch Möglichkeiten zur direkten Integration in einen ELK-Stack oder Splunk, für ein zentrales Logging-System.

Security / Authentifikation

Für die Sicherheitsfeatures und die Benutzer/Computer-Authentifikation können auch verschiedene Programme verwendet werden. Eine direkte Einbindung von LDAP, Active Directory oder RADIUS zur Benutzerauthentifizierung wird mit der Installation von Zusatzpaketen wie ldap-utils oder tacplus-client ermöglicht. Das Management der User erfolgt in den meisten Fällen direkt unter Linux, wodurch auch die Anbindung von AAA-Protokollen einfach ist. Für die Authentifizierung der Benutzer oder Computer am Switchport kann durch die Verwendung von Hostapd und wpasupplicant RADIUS verwendet werden. Zusätzlich gibt es noch verschiedene Tools und Programme zur Anbindung an Identity Management und SingleSingOn-Systeme wie den Dell One Identity Manager (D1IM).

Netzwerkvirtualisierung (SDN)

Software Defined Networks werden immer wichtiger und finden auch im Open Networking Einsatz. Viele Betriebssystemhersteller bieten eigene SDN-Controller und SDN-Lösungen für ihre Betriebssysteme an, die manchmal sogar mit Systemen anderer Hersteller kompatibel sind. Als Controller für SDN-Fabrics können aber auch Openstack (NFV) oder VMware vSphere bzw. VMware NSX dienen. Es gibt aber auch Open-Source Projekte für SDN-Controller, wie zum Beispiel ONOS (Open Network Operating System) von der Open Networking Foundation oder OpenDaylight, ein Projekt der Linux Foundation. Die Wahl des SDN-Controllers muss mit Blick auf die Kompatibilität der vorhandenen Netzwerkinfrastruktur erfolgen. Da viele NOS-Hersteller eigene Lösungen anbieten, gibt es hier noch Aufholbedarf, auch wenn spezielle SDN-Controller als Teil des Ecosystems angeboten und beworben werden.

5.4. Netzwerkautomatisierung und Zero-Touch-Provisioning

Tools und Technologien zum Netzwerkmanagement und zur Automatisierung würden zwar eigentlich zum Ecosystem dazugehören, werden aber hier in einem eigenen Kapitel behandelt, da Orchestration, Automatisierung und Zero-Touch-Provisioning ein sehr wichtiger Bestandteil der Open Networking Architektur sind. Viele der nachfolgend vorgestellten Betriebssysteme bieten eine REST-API an und können so konfiguriert, gemonitort und gesteuert werden. Auch die Automatisierung mit DevOps-Prozessen und Tools wie Ansible, Chef oder Puppet findet im Open Networking eine Anwendung.

Die Effizienzsteigerung und Optimierung eines Netzwerks benötigt ab einer gewissen Größe ein Maß an Automatisierung, welche bereits beim Provisionieren und der initialen Konfiguration der Komponente anfängt. Zero-Touch-Provisioning erlaubt es, eine Netzwerk-Komponente direkt in ein bestehendes Umfeld zu installieren und mithilfe von DHCP oder TFTP die richtige Software und Konfiguration zu deployen. Cumulus Networks hat in einem Artikel verschiedene Szenarien, in welchen eine Automatisierung und ZTP im Netzwerkbereich Sinn macht. Bei der Änderung einer Konfiguration, welche auf vielen Geräten gesetzt werden muss, zum Beispiel die Änderung eines NTP- oder Syslog-Servers, wird oft darauf vertraut, dass der Netzwerkadministrator nicht auf einzelne Komponenten vergessen hat. Die manuelle Änderung solcher globalen Konfigurationen und das Ausrollen dieser nimmt oft einige Stunden in Anspruch. Um diesen Vorgang zu automatisieren, können DevOps-Tools ihre Stärken ausspielen und den Vorgang, mit einem verlässlichen Ergebnis, auf einige Sekunden verkürzen. Der Prozess des zeitnahen Austausches eines defekten Switches, „Hot Swapping“, wird durch die Verwendung von ZTP und DevOps erleichtert, da ein kaputtes Gerät schnell und einfach mit einem frisch provisionierten und konfigurierten Gerät ersetzt werden kann. Gerade bei hochverfügbaren Netzwerken mit hohen Bandbreiten ist ein rascher Austausch von defekten Switches notwendig. Außerdem ist die Verwendung von DevOps für das Konfigurationsmanagement hilfreich, da solche Tools Möglichkeiten bieten, die vorgenommenen Veränderungen an der Konfiguration zu speichern. [27]

Für Open Networking haben Technologien wie ZTP und DevOps eine besondere Bedeutung. Bisher war man für die Konfiguration von Netzwerkkomponenten auf Plugins und Erweiterungen für Automatisierungstools angewiesen, welche mit der CLI eines Switches oder Routers umgehen können und diese richtig interpretieren. Durch die Verwendung von Linux als Basis-Betriebssystem wird diese Abhängigkeit entfernt und die Geräte können, sofern die NOS die Konfiguration über die Linux-Command-Line unterstützen, wie ein Linux-Server behandelt werden. Dadurch ist es auch für Linux-Administratoren und DevOps-Administratoren einfacher, die Einbindung von Netzwerkkomponenten durchzuführen und sie schlüpfen in die Rolle eines Netzwerkadministrators bzw. fördern eine engere Kooperation. Auch das Deployment von Applikationen für das Monitoring oder anderer Kategorien eines Ecosystems kann mithilfe von DevOps ausgerollt und installiert werden. Einige NOS für Open Switches besitzen eine eigene CLI, wodurch auch hier DevOps-Module für die einfachere Automatisierung verwendet werden sollten.

Die automatische Konfiguration muss aber nicht unbedingt mit der CLI erfolgen, da mit NETCONF, YANG und REST-APIs weitere Schnittstellen für die Konfiguration zur Verfügung stehen. NETCONF ist ein Netzwerkmanagement-Protokoll zur Konfiguration von Netzwerkkomponenten und wurde von der IETF im RFC 6241 [28] standardisiert. Mit diesem Protokoll kann die Konfiguration eines Netzwerkgeräts empfangen, geschrieben, editiert, gesperrt oder entsperrt werden. Das verwendete Format ist dabei XML oder JSON, wobei die Modellierung der Konfiguration mit YANG-Models erfolgt. YANG (Yet Another Next Generation) ist eine Datenmodellierungssprache, welche für die von NETCONF gesendete und empfangene Konfiguration dient und ebenfalls von der IETF im RFC 6020 [29] entworfen wurde. Aus den mit YANG geschriebenen Modellen werden JSON oder XML-Dateien erstellt, welche für die spätere Konfiguration mit NETCONF verwendet werden. Die Modellierungssprache ist vor allem für Programmierer der Programmiersprache C oder C++ einfach zu lernen, da eine ähnliche Syntax verwendet wird. Die Verwendung von REST-APIs stellt ebenfalls eine Möglichkeit für die Netzwerkautomatisierung dar, wobei

diese, wie auch NETCONF, vom NOS-Hersteller erstellt werden sollte. Es ist zwar möglich, auf offenen Systemen eigene APIs zu schreiben, zum Beispiel mit OpenAPI, jedoch ist der Aufwand extrem groß. REST-APIs mit für Menschen lesbaren Bezeichnungen sind für viele Netzwerkadministratoren die einfachste Möglichkeit zur Automatisierung und zeichnen sich durch die einfache Implementierung in Runbooks oder Playbooks für DevOps-Applikationen aus. [29] [27]

Nachfolgend wird der ZTP-Prozess am Beispiel einer Neuinstallation und Provisionierung eines Netzwerkswitches Schritt für Schritt erklärt, um einen Überblick über den Ablauf zu erhalten.
Der Prozess ist vergleichbar mit der automatischen Installation von SCCM-Systemen, wo auch eine Provisionierung vorgenommen wird. ZTP automatisiert die Installation des Betriebssystems und von anderen Abhängigkeiten, wie Sicherheitspatches und zusätzlicher Software, des verwendeten Einsatzgebiets. [30]

Bisher war ein ZTP von Netzwerkkomponenten nicht oder nur eingeschränkt möglich, da es keine Umgebung für das automatische Installieren eines Images gab. Die Geräte mussten zuerst initial konfiguriert werden (z.B..: DHCP), damit anschließend überhaupt erst eine automatische Provisionierung vorgenommen werden kann. Am besten lässt sich der ZTP-Prozess von ONIE mit einer Schritt-für-Schritt-Anleitung erklären. Das ONIE bietet durch das vorhin beschriebene Autodiscovery die Möglichkeit, einen Open Networking Switch direkt mit einem NOS-Image auszustatten. Dabei werden im Subnet IP-Helper-Adressen vergeben, welche dem ONIE die benötigten Informationen, wie die Namen des NOS-Images und der initialen Konfigurationsdatei sowie deren Downloadpfad, mitteilen. Anschließend startet ONIE den Download der Dateien, startet automatisiert die Installationsroutine des NOS-Images und konfiguriert es anschließend mit der initialen Konfiguration, welche für weitere Schritte benötigt wird. [30] [31]

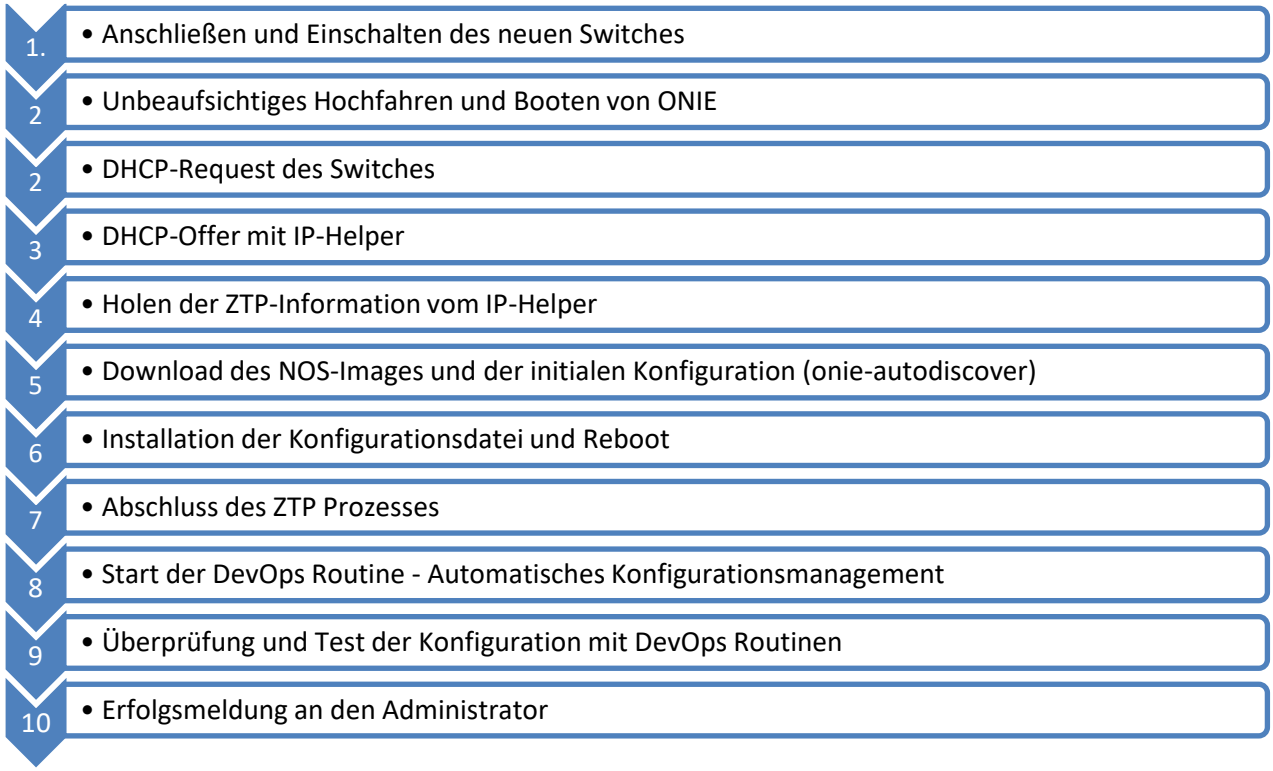

Abbildung 8 - Zero Touch Provisioning Prozess

Nach dem Abschluss des ZTP-Prozesses kann mit der Netzwerkautomatisierung und der Konfiguration des Switches mithilfe von DevOps-Tools, wie Ansible, Puppet oder Chef begonnen werden. Hier muss der Netzwerkadministrator zuvor Runbooks für die automatische Konfiguration des Switches erstellt haben,

welche eine der oben beschriebenen Schnittstellen verwendet. Da viele NOS bereits Module für ihre CLIs zur Verfügung stellen, wird in diesem Beispiel von einer automatischen Konfiguration mittels CLI ausgegangen. Für den Automatisierungsprozess werden die für diesen Switch passenden Konfigurationen anhand der Zuweisung zu einem Runbook vorgenommen und die Parametrisierung des Switches gestartet. Nach Abschluss der DevOps Routine erfolgt meist eine Erfolgsmeldung an den Administrator und es sollten direkt danach Tests und Überprüfungen des zuvor provisionierten Switches gestartet werden. Nach Abschluss aller Prozesse und Routinen ist der Switch fertig provisioniert und bereit für den Betrieb in der Produktion. Der komplette Automatisierungsvorgang erfolgt dabei, bei guter Vorkonfiguration, automatisch und ohne Eingriffe des Administrators. [27] [32]

Der Provisionierungsprozess deckt alle Schritte, vom initialen Einspielen des Betriebssystems bis zur Überprüfung der Konfiguration, ab und sorgt für eine höhere Effizienz beim Tausch der Hardware. Wenn von einer Verwendung von DevOps im Netzwerkbereich die Rede ist, wird auch oft die Bezeichnung „NetDevOps“ oder „NetOps“ synonym dazu verwendet, um den Fokus auf die Netzwerkautomatisierung zu richten. Vor allem im Open Networking Bereich ist die Verwendung des ZTP-Prozesses mithilfe von ONIE und die Konfiguration des Switches bzw. die Installation von Programmen des Ecosystems mit DevOps-Routinen weit fortgeschritten und bietet einen großen Vorteil gegenüber proprietären Lösungen bzw. Blackbox Switches. [33] [32]

Abbildung 9 - Carl and Captain Cloud Discuss NetDevOps [33]

Im Traditional Networking kommt meist eine Mischung aus DevOps-Routinen und proprietären Management-Tools zum ZTP zum Einsatz, was die Abläufe beim Hardwaretausch noch zusätzlich erschwert und aufgrund von Inkompatibilitäten oft eine Konfiguration per Hand erforderlich macht. Aber auch Hersteller von Blackbox Switches und Netzwerklösungen haben bereits eine breite Community, welche sich auf die Netzwerkautomatisierung und das Zero Touch Provisioning konzentriert, darunter auch Cisco mit seinem DevNet. Die Verwendung von ZTP und der Automatisierung im Netzwerk wird vor allem in Zeiten von Software Defined Networks und Netzwerkvirtualisierung immer wichtiger und sollte deshalb forciert werden.

6. Network Operating Systems (NOS)

Dieses Kapitel gibt einen Überblick über verschiedene Netzwerkbetriebssysteme für Switches, wobei der Fokus auf Open Networking liegt, während Blackbox-Lösungen nur kurz angeschnitten werden. Dadurch sollen die verschiedenen Vor- und Nachteile beziehungsweise die Hervorhebungsmerkmale des Betriebssystems nähergebracht werden. Dabei soll der Fokus auf den Security-Eigenschaften, den Möglichkeiten zur Automatisierung und der Kompatibilität zu Software-Defined-Networks liegen. Es werden sowohl Open-Source Betriebssysteme als auch Closed-Source Lösungen präsentiert und die verantwortlichen Firmen sowie Teile des Ecosystems vorgestellt. Diese Auflistung ist nicht komplett und bietet deshalb keinen Anspruch auf Vollständigkeit. Es wurde nur darauf geachtet, einige Betriebssysteme vorzustellen, die auch mit dem für die anschließenden Testfälle kompatibel sind und um die Unterschiede zu Blackbox-Betriebssystemen hervorzuheben.

6.1. Traditionelle NOS

Die bekanntesten Netzwerkbetriebssysteme sind jene der großen Hersteller von Netzwerkkomponenten, Cisco, Juniper, HP, etc. Diese Hersteller bevorzugen den Verkauf von Blackbox Switches und vertreiben die Software nur in Verbindung mit ihren Komponenten und einer Lizenzgebühr. Auch wenn die Weiterentwicklung dieser Betriebssysteme voranschreitet und der Fokus auf kosteneffiziente Technologien und Orchestration wandert, bleibt das große Problem von traditionellen NOS der administrative Aufwand und die Inkompatibilität zu zentralen Managementlösungen. Jeder Hersteller von Blackbox Switches ist daran interessiert, eigene Management-Softwaresysteme für die vertriebene Hard- und Software zu vertreiben, was ebenfalls mit Lizenzgebühren verbunden ist. Ein Vorteil solcher Systeme ist, dass es seit Jahren keine großen Änderungen gegeben hat, da die Hersteller bei diesen Betriebssystemen mit Neuerungen sehr konservativ umgehen. Durch die riesige Verbreitung von Blackbox Switches gibt es auch zahlreiche Zertifizierungen für die Benutzung dieser Systeme und eine verbreitete Community. Für viele Firmen ist es auch wünschenswert, ein System aus einer Hand zu beziehen und dieses auf bereits existierendes Know-How der Mitarbeiter anzupassen. Auch können gerade die Mitarbeiter der Netzwerkabteilung und deren Einstellung zu neuen Technologien eine Rolle bei der Wahl der Netzwerklösung spielen. Das IOS von Cisco ist das beste Beispiel für ein Blackbox System, für welches es bekannterweise Unmengen Zertifizierungen und Managementlösungen gibt. Das große Problem bei geschlossenen Ecosystemen ist, dass die Kompatibilität zwischen den Netzwerkkomponenten und den Managementlösungen teilweise stark vom aktuellen Release abhängt. Sobald eine neue Generation von Netzwerkkomponenten veröffentlicht wird, ist oftmals eine neue Netzwerkmanagementlösung oder ein kostenpflichtiges Update des bestehenden Systems notwendig.

Trotzdem haben Hersteller von Blackbox-Systemen die aktuellen Entwicklungen rund um Open Networking erkannt und eigene offene oder teilweise geöffnete Betriebssysteme für ihre Komponenten veröffentlicht. Für den Einsatz im Datacenter bietet Cisco bereits seit einiger Zeit das NX-OS, welches auf allen Switches der Nexus-Reihe läuft und ebenfalls über viele offene Schnittstellen verfügt. Der Fokus von NX-OS liegt ebenfalls bei der Verwendung von Linux als Grundsystem, Kompatibilität mit offenen Schnittstellen und der Trennung von Data- und Control-Plane, um Performanceeinbußen der Switching-ASIC bei hoher Systemlast am Management-System entgegenzuwirken. Der Hersteller hat mit dem DevNet eine große Community für die Automatisierung und das Netzwerkmanagement dieser Systeme aufgebaut, wo auch Laborübungen und Beispiele für die Automatisierung zu finden sind. [34]
Außerdem hat Cisco mit IOS-XE eine neue Version seines Switch-Betriebssystems IOS vorgestellt, welches ebenfalls als Dienst auf einem Linux Betriebssystem läuft und auch im Campus-Bereich die Trennung von Data-Plane und Control-Plane vornimmt. Weiters wird der Fokus auf Automatisierung beziehungsweise die Kompatibilität mit DevOps-Tools gelegt und die Konfiguration zusätzlich über APIs, YANG-Models und NETCONF ermöglicht. Damit wählt Cisco eine ähnliche Lösung wie Hersteller von Open Networking

Betriebssystemen, ermöglicht die Installation aber nur auf hauseigenen Blackbox-Switches. Dieses Betriebssystem ist mit den neuen Campus-Switches der Catalyst-Reihe von Cisco und der neuen Generation der ASR-Edge-Router kompatibel. [35]

Auch der Hersteller Juniper hat in seinem etablierten Betriebssystem JunOS neue Funktionen eingeführt, welche auf die Automatisierung und Netzwerkprogrammierung ausgerichtet sind. So gibt es hier die Möglichkeit, Systemparameter und Statuswerte sowie Telemetriedaten über eine API abzufragen und diese zu verarbeiten. Die Trennung des Management-Systems vom Switching-System wurde ebenfalls integriert und auch Schnittstellen für SDN wurden definiert. Durch die Unterstützung von Open-Standards wie NETCONF, YANG-Models und APIs wird auch hier die Netzwerkautomatisierung klar weiterentwickelt und vorangetrieben. Leider handelt es sich bei JunOS ebenfalls um ein geschlossenes System, welches nur auf Netzwerkkomponenten der Firma Juniper lauffähig ist. [36]

Ein Vorreiter bei der Verwendung von offenen Schnittstellen ist der Netzwerkhersteller Dell, welcher auch Whitebox bzw. Britebox Switches vertreibt. Bereits in OS9 wurden erste Maßnahmen zur Netzwerkautomatisierung getätigt und offene Schnittstellen wie NETCONF eingeführt. Jedoch erst mit OS10 wurde Dell zu einem Vorreiter im Open Networking, da hier vollständig auf eine offene Architektur gesetzt wird. Das OS10 gibt es in drei verschiedenen Varianten, wobei zwei davon trotzdem als traditionelle NOS bezeichnet werden können, da sie nur mit Switches von Dell kompatibel sind. Dabei handelt es sich um die OS10 Enterprise Edition und die OS10 Open Edition, welche beide ein offenes Betriebssystem als Basis verwenden. Diese Basis ist gleichzeitig die dritte Version des NOS, OpenSwitch OPX, welches später vorgestellt wird.
Die Enterprise Edition ist mit Eigenentwicklungen von Dell ausgestattet und mit NX-OS vergleichbar, da der Fokus hier auf offenen Schnittstellen und nicht auf offener Hardware liegt. Der Unterschied liegt jedoch darin, dass diese Version nicht nur Zugang zur CLI, sondern auch zur Linux Shell bietet, wodurch die Automatisierung mit DevOps-Tools um einiges erleichtert wird. Diese Version von OS10 beinhaltet viele Möglichkeiten für die Orchestration und die Konfiguration, wie die CLI, NETCONF, YANG oder verschiedene APIs. Außerdem wird mit dieser Version der Enterprise Support und ein Lizenzmodell angeboten. Die OpenEdition von OS10 bietet dieselben Services wie Enterprise Support, verwendet jedoch für alle Applikationen des Ecosystems Open-Source-Tools, um eine offene Basis bereitzustellen. Dadurch gibt es noch mehr Möglichkeiten für NetDevOps-Technologien, da für viele dieser Tools bereits Module existieren. Die Architektur wird nachfolgend bei der Vorstellung von OpenSwitch OPX vorgenommen, da alle drei Betriebssysteme dieselbe Basis verwenden. [37]

6.2. OpenSwitch OPX

Die Community-Version von Dell OS10 ist das Open-Source-Projekt OpenSwitch OPX, dessen Entwicklung ursprünglich von HP vorangetrieben wurde. Nach dem Ausstieg von HP übernahm Dell zusammen mit verschiedenen ASIC-Herstellern und dem Bare-Metal-Switch-Hersteller Accton die Leitung des Projekts, das von der Linux Foundation verwaltet wird. OPX verfolgt einen neuen Ansatz für die Architektur eines NOS und verwendet systemintern für alle Abläufe der Konfiguration eine API. Dabei handelt es sich um die sogenannte Control-Plane-Services-API (CPS-API), welche die Steuerung aller Layer2 und Layer3-Funktionen sowie der Steuerung der Data-Plane übernimmt und als zentrale Schnittstelle im Betriebssystem fungiert. Das Alleinstellungsmerkmal von OPX ist, dass es keine CLI besitzt und die komplette Konfiguration des Switches entweder über Linux-Kommandos oder über die CPS-API oder mithilfe von YANG-Models vorgenommen werden muss. Das Ziel, welches damit verfolgt wird, ist, alle Abläufe der Konfiguration über Scripts und APIs vorzunehmen und somit die Fehlerquote bei der Konfiguration zu senken. Gerade bei großen Unternehmen wie Telekom-Betreibern ist es aufgrund der Größe und der dadurch schlechten Administrierbarkeit des Netzwerks notwendig, die Automatisierung stark zu forcieren. Auch aufgrund der Tatsache, dass es unter OPX keine CLI gibt, richtet sich dieses Betriebssystem vor allem an Linux-Administratoren, die mit DevOps- und Scripting-Know-How die Konfiguration vornehmen. OPX selbst ohne

zusätzliche Applikationen und Teile des Ecosystems ist für den Einsatz in der Produktion nicht geeignet. Um aus diesem System ein vollwertiges NOS zu bauen, muss sich jeder Anwender sein eigenes Ecosystem zusammenstellen und verwalten. Durch die offene Basis und den Zugriff auf die Linux-Shell inklusive Administrator-Berechtigungen ist dies ohne Probleme möglich. [37] [38]

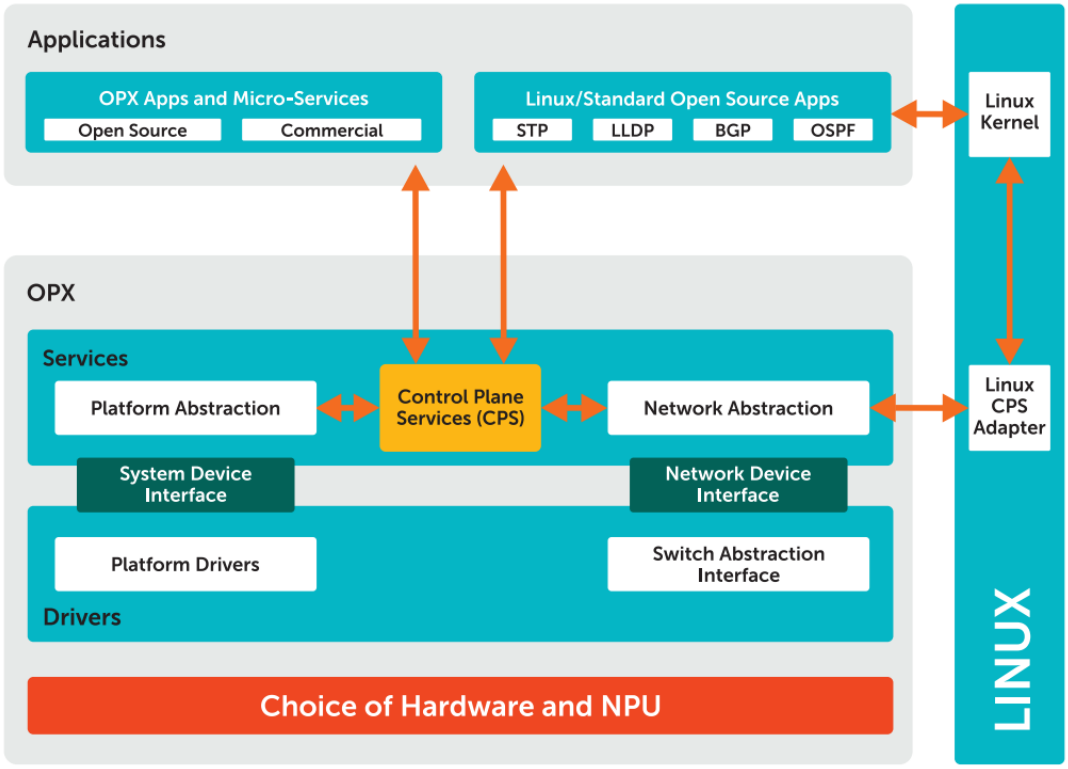

Abbildung 10 - OpenSwitch OPX Architektur [38]

Als Grundlage für das Betriebssystem verwendet OPX, wie auch OpenNetworkLinux, einen nicht modifizierten Linux-Kernel. Dieser ist für die Control-Plane und das Management-Interface des Switches verantwortlich. Für die Kommunikation und Steuerung der ASIC, und damit die Data-Plane, wird das vom Open Compute Project geleitete Tool Switch Abstraction Interface (SAI) verwendet, welches mithilfe von APIs einen herstellerunabhängigen Weg der Konfiguration der ASIC bereitstellen soll. Diese Hardwareschnittstelle wird vom Network Device Interface gesteuert, welche zusammen mit dem Network-Abstraction-Dienst auf Statusänderungen des Linux-Netlink-Managers reagiert und diese Änderungen an die ASIC weiterleitet. Die Kontrolle der Plattform, also des Switch-Chassis, wird durch das System-Device-Interface vorgenommen, während die notwendigen Treiber als Grundlage dafür dienen. Die beiden Dienste, Platform Abstraction Service und Network Abstraction Service, werden von der CPS-API gesteuert und kontrollieren mit ihren Schnittstellen die Switch-Hardware. Außerdem agiert die CPS-API als Schnittstelle für alle Applikationen des Ecosystems, die derzeit implementiert werden. Die Steuerung dieser Schnittstelle und damit die Konfiguration des Switches erfolgt mit dem Linux CPS Adapter und Libraries für die Programmiersprachen C und Python. [38]

Zurzeit sind 14 Switches mit OPX kompatibel, wobei es sich dabei fast ausschließlich um Switches von Dell mit einer Broadcom-ASIC handelt. Die einzige Ausnahme ist ein Edgecore-Switch, welcher eine Cavium ASIC verwendet. Durch die vielen offenen Schnittstellen und die grundlegend offene Architektur sowie die standardmäßige Verwendung von Open-Source-Projekten für wichtige Systemschnittstellen, ist OpenSwitch OPX ein Betriebssystem, das als Vorzeigebetriebssystem für den Grundgedanken von Open Networking dient. Da es aber noch immer mitten in der Entwicklung steckt, muss abgewartet werden, ob OPX bereit für

den Einsatz in der Produktion ist. Während Dell mit OS10 vorzeigt, welche Möglichkeiten diese offene Architektur bietet, muss das Ecosystem von OPX noch um viele Applikationen erweitert werden. Um die Grundidee von OPX beizubehalten, ist es notwendig, hier weiterhin auf Open-Source-Tools und Open-Source-Projekte zu setzen. Die Verwendung der CPS-API als zentrale Schnittstelle für die Konfiguration des Switches ist einzigartig in diesem Umfeld und wird weiterentwickelt werden. Noch ist diese Schnittstelle nicht fertig, sie wird aber laufend von der Community weiterentwickelt und zeigt, welche Vorteile sie bietet. Für den ein oder anderen Netzwerkadministrator wird es wünschenswert sein, eine CLI zu implementieren., jedoch liegt der Fokus von OPX klar auf der zentralen CPS-API, weshalb dieses Alleinstellungsmerkmal wohl beibehalten werden wird. Dell hat mit diesen Betriebssystemen eine klare Vorreiterrolle im Open Networking eingenommen und sorgt dafür, dass es eine Alternative zu OpenNetworkLinux als Grundlage für Netzwerkbetriebssysteme gibt. Auch wenn die Kompatibilitätsliste von OPX noch nicht so groß ist wie jene von ONL, ist die Arbeit von Dell und der Community rund um OpenSwitch hervorzuheben und weiterhin zu beobachten.

6.3. Cumulus Linux

Das führende Open Networking Unternehmen, Cumulus Networks, wurde 2010 gegründet, hat seinen Sitz in Mountain View, California, und vertreibt das wohl bekannteste aller Open Networking Betriebssysteme, Cumulus Network Linux. Nach drei Jahren der Entwicklung und einem Schattendasein erregte Cumulus Networks 2013 Aufsehen, da es sein offenes Netzwerkbetriebssystem Cumulus Network Linux vorstellte. Ziel des Unternehmens ist es, eine offene Netzwerkinfrastruktur für Unternehmen bereitzustellen und die Flexibilität, Effizienz und Automatisierung der Netzwerkadministration voranzutreiben. Es wird damit geworben, dass Kunden ihre Rechenzentren so wie Google und Facebook betreiben können, nämlich kostengünstig und hochautomatisiert. Mit Cumulus Network Linux vertreibt das Unternehmen ein Betriebssystem, das eine sehr große Kompatibilität zu verschiedenen Whitebox und Bare Metal Switches hat und bereits von einigen großen Unternehmen genutzt wird. Im Frühjahr 2013 hat das Unternehmen ONIE, die Installationsumgebung für Open Networking Switches,
vorgestellt, welche ursprünglich nur für die Installation des eigenen Betriebssystems geplant war. Mit ONIE trägt das Unternehmen die Verantwortung für ein großes Projekt des OCP-Networking-Teams, dessen Vorsitz auch von Cumulus eingenommen wird. Außerdem treibt Cumulus zusammen mit Big Switch Networks die Entwicklung von OpenNetworkLinux voran, welches auch als Grundlage für Cumulus Network Linux dient und deshalb auf eine große Liste an kompatiblen Geräten zurückgreifen kann. Das Unternehmen ist bei vielen Projekten im Open Networking beteiligt und ist daran interessiert, die Entwicklung dieser Technologie voranzutreiben. [39]

Das Unternehmen bewirbt Cumulus Network Linux als Grundlage für Web-Scale-Networking, also ein hochskalierbares und hochautomatisiertes Netzwerk. [40] Das System basiert auf OpenNetworkLinux und damit auf Debian Jessie. Die Kompatibilitätsliste beinhaltet mehr als 70 Switches, womit die Wahl der Netzwerkhardware an die Kundenanforderungen angepasst werden kann. Als zentrale Schnittstelle im System fungiert der von Cumulus Networks entwickelte Dienst „switchd“, der als Schaltzentrale zwischen dem User-Space, dem Kernel-Space und dem Switching-ASIC fungiert. Für den Layer3-Stack wird Quagga, beziehungsweise in neueren Versionen FRRouting, verwendet. Das Betriebssystem verfügt über eine optionale HTTP-API, welche nachinstalliert werden muss, bietet aber keine Möglichkeit der Konfiguration über NETCONF und YANG-Models. Für die Konfiguration auf der Kommandozeile stehen eine eigens programmierte CLI, das NCLU, und die Linux-Shell zur Verfügung. Das Unternehmen bewirbt auch die Möglichkeiten, die durch die Verwendung des Linux-Betriebssystems bestehen. So gibt es eine Anleitung, wie Docker unter Cumulus Linux installiert wird und wie diese Container direkt im Control-Plane-System des Switches gestartet werden können. Es gibt jedoch auch Hinweise darauf, dass von der Nutzung CPU-intensiver Applikationen sowohl in Docker-Containern als auch am Betriebssystem selbst Abstand genommen werden soll, da sonst Performanceeinbußen zu befürchten sind. Im Grunde ist es jedoch möglich, jede Linux-Applikation auf dem Switch zu installieren und zu betreiben. [41]

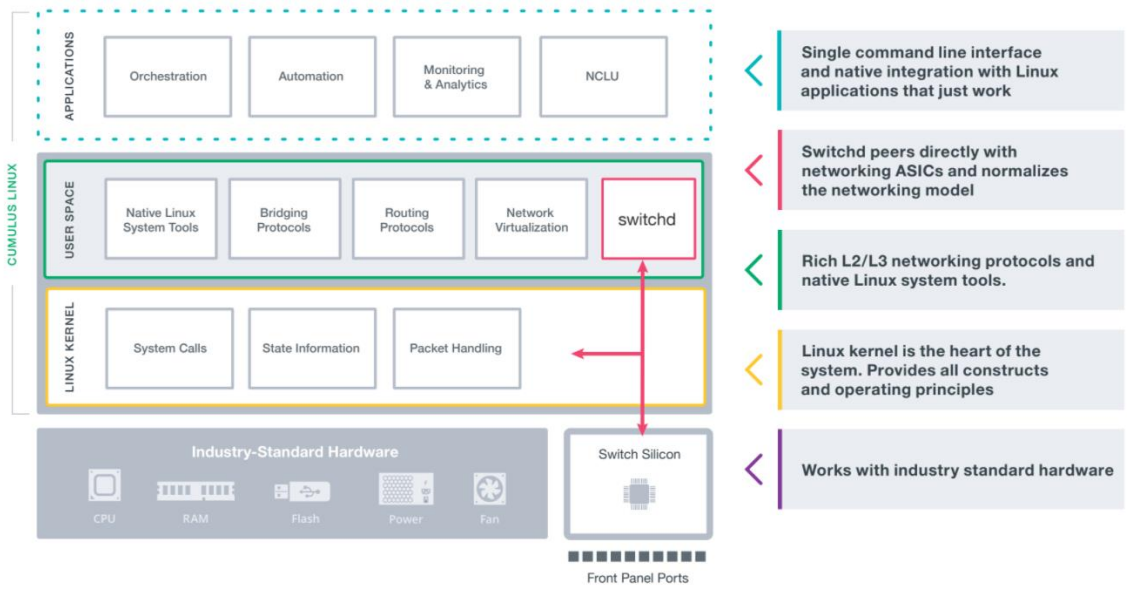

Abbildung 11 - Cumulus Network Linux Architektur [42, p. 3]

Auch die Orchestration mit OpenStack beziehungsweise die Interaktion mit OpenStack Neutron ist über Schnittstellen möglich, wodurch eine dynamische Provisionierung von VLANs und VXLANs möglich wird. Ein wichtiges Feature von Cumulus Network Linux ist die Netzwerkautomatisierung, welche auch groß beworben wird. Cumulus betreibt Github-Repositories für Module für verschiedene Automatisierungsframeworks, wie Ansible, Puppet und Chef. Der Support für diese Repositories wurde jedoch erst im Jänner 2018 eingestellt, da das Unternehmen die Meinung vertritt, dass diese Module gar nicht notwendig sind, weil es sich bei Cumulus um ein vollwertiges Linux-Betriebssystem handelt und deshalb keine proprietären Module notwendig sind. [43] Nähere Informationen über das Lizenzmodell sind nicht vorhanden, Cumulus Network Linux wird jedoch als kostengünstige Lösung beworben, weshalb die Lizenzkosten im Vergleich mit traditionellen NOS relativ vermutlich günstig sein werden. [39]

Die Dokumentation des Betriebssystems ist vollständig und detailliert aufgebaut und wird laufend erweitert. Es gibt viele Beispiele für mögliche Konfigurationen und sogar die Möglichkeit zur Virtualisierung des Betriebssystems für Testzwecke. Rund um das Betriebssystem hat sich ein großes Ecosystem entwickelt, welches die Unterstützung vieler Drittanbieter-Applikationen ermöglicht und die Kompatibilitätsliste der Hardware laufend erweitert. Da die Automatisierung über DevOps-Tools mit Linux-Modulen möglich ist, gibt es ausreichend Dokumentation, obwohl eigene Module für diese Frameworks entwickelt wurden, die ebenfalls ausführlich dokumentiert sind. Cumulus bietet verschiedene White-Papers über das Thema Open Networking, liefert Best-Practice-Guides für die Konfiguration und Absicherung des Betriebssystems, wie zum Beispiel den „Securing Cumulus Linux“-Best-Practice-Guide [44], und leistet große Beiträge zum Open Compute Project.

6.4. Ipinfusion OcNOS

Das Unternehmen IP Infusion ist seit 20 Jahren im Netzwerkbereich tätig, führte jedoch ein Schattendasein, da die vertriebene Software nur für OEM-Kunden zur Verfügung stand. 1999 wurde IP Infusion vom Entwickler der Open-Source-Routing-Software Zebra mit dem Ziel gegründet, eine Enterprise-Version dieses L3-Stacks zu vertreiben. Zebra wurde laufend weiterentwickelt und ist heute bei vielen NOS unter den Namen Quagga und FRRouting im Einsatz. Das Hauptgeschäftsfeld von IP Infusion ist weiterhin der Vertrieb der Enterprise-Version von Zebra, um die ein komplettes Netzwerkbetriebssystem entwickelt wurde. Das ZebOS wird von vielen Netzwerkkomponentenherstellern als Grundlage oder komplettes Betriebssystem verwendet. So setzen zum Beispiel F5, Citrix und Netgear bei vielen der eigenen

Komponenten auf das NOS aus dem Hause IP Infusion und vertreiben das Betriebssystem unter ihrer eigenen Marke. Durch die Architektur des Betriebssystems kann die CLI an die Wünsche des OEM-Partners angepasst werden, wodurch es für den Kunden oft nicht möglich ist festzustellen, dass ZebOS im Hintergrund als Grundlage dient. Der große Vorteil für OEM-Partner ist, dass es sich für diese oft nicht auszahlt, ein eigenes Betriebssystem inklusive Support für manche Produkte herzustellen. Deshalb wird auf ZebOS gesetzt, das von IP Infusion an die Hardware angepasst wird und dann auch der Support von IP Infusion bereitgestellt wird. [45]

Mit OcNOS (Open Compute Network Operating System) wurde im Jahr 2014 ein Netzwerkbetriebssystem für Whitebox und Bare Metal Switches veröffentlicht, welches auf ZebOS und damit einen Linux-Kernel basiert. Durch die jahrelange Erfahrung im Entwurf und der Erstellung von ZebOS hat IP Infusion viel Know-How aufgebaut und in die Entwicklung von OcNOS einfließen lassen. Es wird auch damit geworben, dass aufgrund der internen Architektur und der Verwendung von APIs als zentrale Schnittstelle die CLI von OcNOS an die Kundenwünsche angepasst werden kann, wenn dies notwendig und für einen Kunden erstrebenswert ist. Vermutlich aufgrund der jahrelangen Erfahrung verfolgt das Unternehmen rund um das Betriebssystem eine Lizensierung, wie sie von Anbietern traditioneller NOS bekannt ist und vertreibt OcNOS in drei Produktschienen mit verschiedenen Features. Die Version „OCNOS-ENT-IPBASE“ beinhaltet die Lizenzen für eine Verwendung des Kompletten Layer2 und Layer3-Stack, inklusive dynamischer Routingprotokolle, für Enterprise Switches mit TenGigabitEthernet-Ports. Für Switches mit Port-Geschwindigkeiten von über 10 Gigabit ist die „OCNOS-DC-IPBASE“-Version notwendig, welche dieselben Funktionen bietet. Die dritte Version „OCNOS-DC-MPLS“ bietet zusätzlich noch die Möglichkeit, MPLS beziehungsweise MPLS-TP zu betreiben, ist jedoch nicht mit allen Switches kompatibel. Das Betriebssystem bietet verschiedene Möglichkeiten zur Konfiguration und Parametrisierung beziehungsweise dem Monitoring der Hardware. So steht den Administratoren eine betriebsbereite und, dank ZebOS, ausgereifte CLI zur Verfügung. Daneben gibt es noch die Möglichkeit, die Konfiguration über NETCONF bzw. YANG-Models und eine REST-API durchzuführen. Außerdem gibt es die Möglichkeit, direkt Zugriff auf die Linux-Shell zu erhalten und direkt mit dem darunterliegenden Linux-Betriebssystem zu arbeiten. Diese Methode der Konfiguration wird jedoch nicht vom Hersteller unterstützt, weshalb es dafür auch keinen Support gibt. [46] [47]

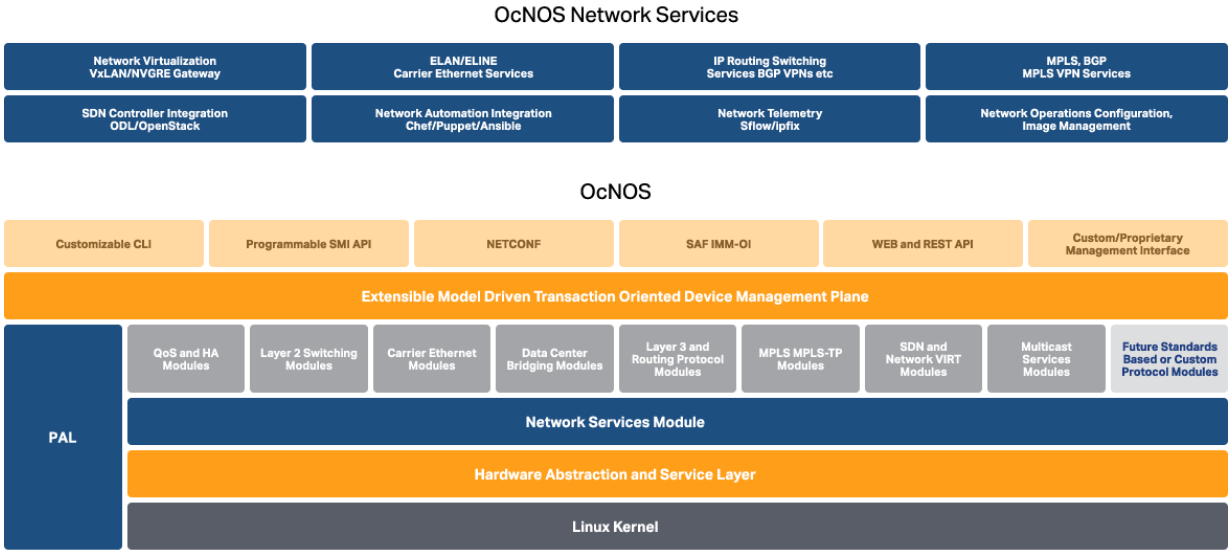

Abbildung 12 - OcNOS Architektur [46]

Die Dokumentation von OcNOS ist nicht frei zugänglich und nur in Verbindung mit einer Lizenz und einer Registrierung im Kundenportal erhältlich. Dafür ist diese aber sehr detailliert und aufgeschlüsselt beschrieben, wodurch die Konfiguration, neben der Ähnlichkeit zur Cisco-CLI, sehr einfach ausfällt. Es gibt auch einen User-Guide zur Konfiguration des Betriebssystems mithilfe von DevOps-Tools wie Ansible oder

die Konfiguration mit NETCONF. Hierfür sind jedoch Module, die vom Hersteller in Github-Repositories zur Verfügung gestellt werden, notwendig. Während diese Arbeit geschrieben wurde, hat IP Infusion die Produktbeschreibung von OcNOS überarbeitet und mehr Details über die Architektur und die Funktionen des NOS veröffentlicht. Die Kompatibilitätsliste beinhaltet einige Switches von Dell, Agema, Edge Core und IM, welche allesamt auf einer Broadcom-ASIC beruhen. Der gute Enterprise-Support und die Liste der Funktionen des Betriebssystems kann sich sehen lassen und ist vollständig , wobei anzumerken ist, dass das komplette Ecosystem aus der Hand von IP Infusion stammt. Durch diese Tatsache erweckt es den Anschein, dass OcNOS ein Enterprise-Betriebssystem unter den Open Networking NOS sein soll und deshalb eher einem traditionellen NOS ähnelt und kein typisches Open Networking NOS ist.

6.5. Pica8 PicOS

Das Unternehmen Pica8 wurde 2009 gegründet, hat seinen Firmensitz in Palo Alto, California, und ist seit der Gründung als Hersteller von Netzwerkbetriebssystemen tätig. Zu dieser Zeit hieß das Betriebssystem noch XORPlus, angelehnt an die Basis für das Betriebssystem, XORP (eXtensible Open Router Platform) [48], ein Layer3-Stack vergleichbar mit Quagga. Zu Beginn verkaufte das Unternehmen Switches zum Discount-Preis, indem es sie zusammen mit dem eigenen Betriebssystem PicOS als Britebox Switches verkaufte. Damals bestand noch keine Möglichkeit, das NOS zu ändern, weil PicOS noch als proprietäres System galt und auf dem(auf den??) als Pica8-Switch vermarkteten Switches verkauft wurde, obwohl Pica8 als Vorreiter und Marktführer von Open Networking Lösungen galt. Nach und nach wurde das NOS erweitert und so führte das Unternehmen auch den Support für den OpenFlow-Stack „Indigo“ von Big Switch Networks ein, welcher als alternativer SDN-Stack auf den Systemen zum Einsatz kam. Im Jahr 2012 erlangte Pica8 Bekanntheit durch die Veröffentlichung ihrer SDN-Referenz-Architektur für Cloud-Anbieter, welche von einigen großen Service-Providern wie Baidu oder NTT Communications eingeführt wurde. Außerdem wurde das Betriebssystem der hauseigenen OEM-Switches um einen Modus für das Management mithilfe von Open vSwitch erweitert, womit PicOS in zwei Varianten betrieben werden kann, der SDN-Edition und der Enterprise-Edition. Pica8 veröffentlicht laufend neue Whitepapers zum Thema Open Networking und beschreibt auch die Vorgänge und Anforderungen bei der Portierung ihrer Software auf neue Hardware. [49]

Nach der Veröffentlichung des ONIE-Installers hat das Unternehmen entschieden, das Betriebssystem auch für Komponenten anderer Hersteller zu optimieren. Dadurch wurde die Hardware-Unterstützung von PicOS laufend erweitert, sodass heute eine Vielzahl an Switches unterstützt werden. Dabei handelt es sich vornehmlich um die hauseigenen OEM-Switches, Bare-Metal Switches der Firma Edgecore und Whitebox beziehungsweise Britebox Switches verschiedener Hersteller wie HP oder Dell. Als Grundlage für das Betriebssystem dient ein nicht-modifiziertes Debian Linux, wodurch die Möglichkeit der Nutzung von Drittanbieter-Tools besteht. Während andere NOS für die Kommunikation mit der ASIC auf das Switch Abstraction Interface setzen, verwendet Pica8 mit der proprietären vASIC eine eigene Lösung. Für die Layer2-Funktionen verwendet Pica8 hauseigene Entwicklungen, während für den Layer3-Stack XORP und BIRD dienen. Für die SDN-Version von PicOS wird für die Kommunikation zwischen der vASIC und Open vSwitch die proprietäre Lösung PicOS IPC verwendet. Für die Konfiguration des Betriebssystems steht grundsätzlich die CLI zur Verfügung, welche an die JunOS-CLI erinnert. Daneben gibt es noch die Möglichkeit den Switch über eine REST-API mit JSON-Daten zu konfigurieren, wobei die CLI ebenfalls mit dieser API arbeitet. Dadurch ist es möglich, dass alle Konfigurationen, die mit der CLI durchgeführt werden können, auch mit der API vorgenommen werden können. Da es sich hierbei um ein Linux-Betriebssystem handelt, kann die gesamte Konfiguration ebenfalls über Linux vorgenommen werden, wodurch auch eine optimale Interaktion und Automatisierung mit DevOps-Tools möglich ist. Zusätzlich bietet PicOS noch die Möglichkeit eines CrossFlow Modus, in welchem ein gleichzeitiger Betrieb von SDN und traditionellem Routing und Switching möglich wird. Dadurch kann man OpenFlow auf einzelnen Ports aktivieren und somit zwischen beiden Modi beliebig umschalten. [50] [15]

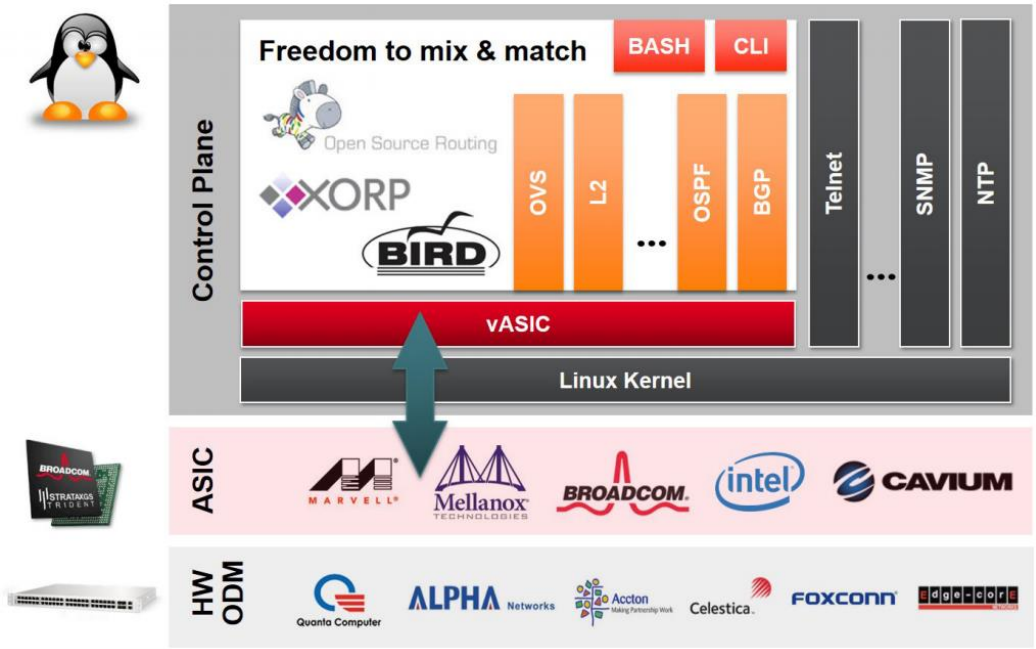

Abbildung 13 - PicOS Architektur [50, p. 2]

Auf der Unternehmensseite gibt es einen Link zur aktuellen Dokumentation, welche frei zugänglich und gut strukturiert ist. Es gibt Konfigurations-Anleitungen für beide Betriebsmodi, SDN-Modus und Enterprise-Modus, als auch Anleitungen für das Beheben bestimmter Problemfälle und die Konfiguration bestimmter Testszenarien. [51] Da Linux als Host-Betriebssystem verwendet wird, kann mithilfe von DevOps-Tools die Automatisierung der Konfiguration ohne Zusatzmodule vorgenommen werden. Trotzdem bietet Pica8 Module für Ansible, Puppet, Chef und Salt an, welche jedoch nicht mehr weiterentwickelt werden, da auf die Standardmodule dieser DevOps-Tools gesetzt wird. Pica8 bietet einige Best-Practice-Guides und Lösungsvorschläge für verschiedene Szenarien an, welche mit der eigenen Software beschrieben werden und für jeden verfügbar sind. Das Unternehmen vertreibt weiterhin eigene Switches, wird aber in Zukunft trotzdem die Kompatibilität zu anderen Switches und ASICs erweitern, um mehr Kunden zu erreichen.

6.6. SONiC

Das Akronym SONiC steht für „Software for Open Networking in the Cloud“ und wurde ursprünglich von Microsoft für die Netzwerkinfrastruktur hinter der Azure-Cloud entworfen. Microsoft hat den Source-Code vom Vorgänger-Projekt Azure Cloud Switch veröffentlicht und zusammen mit der Community zu SONiC weiterentwickelt. Aktuell verwendet Microsoft SONiC als Grundlage für das Produktionssystem ihrer eigenen Cloud-Dienste und betreibt Community-Support und die Weiterentwicklung auf Github. SONiC ist kein normales Betriebssystem wie die vorher genannten NOS, da es sich hierbei um eine Sammlung von Open-Source-Tools handelt, die für ein voll funktionsfähiges Layer3-Gerät notwendig sind. Der Grundgedanke hinter SONiC ist, dass die Verwendung verschiedenster Hardware und Software als Infrastruktur für die Cloud oft dazu führt, dass der Administrationsaufwand exponentiell steigt. Um dieser Entwicklung entgegenzuwirken, wurde das Betriebssystem erfunden, welches dafür sorgt, dass alle notwendigen Features und Konfigurationen mit einer einheitlichen Deployment-Methode und Syntax vorgenommen werden können.
Die Architektur von SONiC basiert zu großen Teilen auf dem Switch Abstraction Interface (SAI) des OCP, welches ursprünglich ebenfalls von Microsoft entwickelt wurde. Durch das SAI ist es möglich, eine einheitliche API für Switching-ASICs zu verwenden und dadurch die Kompatibilität zur Hardware zu erhöhen. Die Konfiguration der Software funktioniert mithilfe einer einzigen Konfigurationsdatenbank, einer redisDB und Konfigurationsdateien im JSON-Format. Die Datenbank besteht aus mehreren Tabellen, in

welchen verschiedene Netzwerkparameter geändert werden können. So gibt es für die Switchport-Konfiguration die Tabelle „PORT“, oder für Layer3-Konfigurationen die „INTERFACE“-Tabelle. Die Konfigurationen werden bei einem Systemstart aus der JSON-Datei unter dem Pfad „/etc/sonic/config_db.json“ ausgelesen und in die Datenbank aufgenommen. Alle Applikationen speichern ihre Änderungen an der Konfiguration in diese Datenbank, wobei ein Zurückschreiben der Konfiguration in die Datei erst nach dem Kommando „config save“ erfolgt.

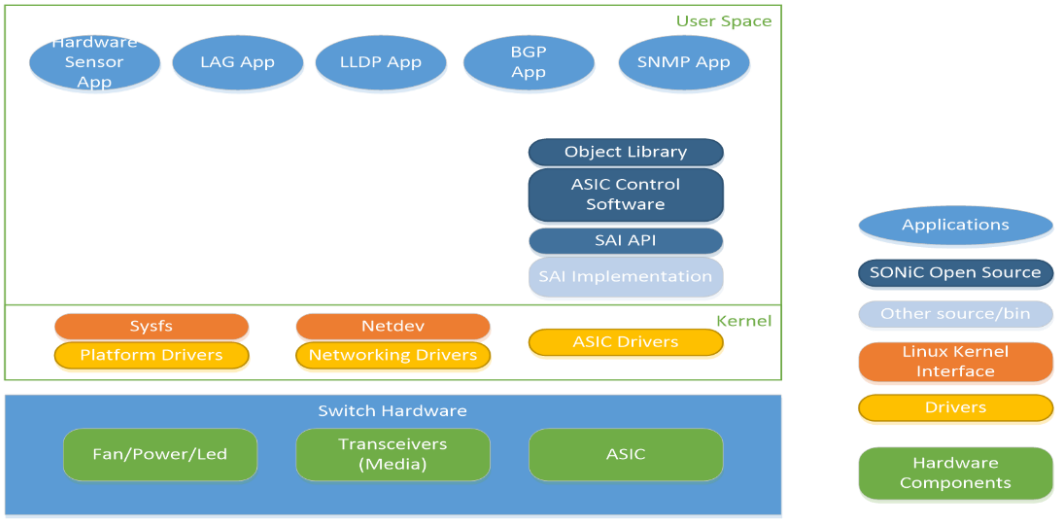

Abbildung 14 - SONiC Architektur [52]

Da es sich bei SONiC um ein Open-Source-Projekt handelt, dessen Dokumentation ebenfalls von der Community vorgenommen wird, ist diese noch nicht vollständig und wird laufend aktualisiert. Die grundlegende Konfiguration wird zwar beschrieben, jedoch ist eine vollständige Referenz über alle möglichen Parameter nicht vorhanden, was die Konfiguration anfangs erschwert. Trotzdem gibt es eine Referenz über alle vorhandenen „Show“-Befehle zum Auslesen der aktuellen Systemparameter und einen Troubleshooting-Guide, in welchem Lösungen für bekannte Probleme beschrieben werden. Viele Hersteller von offenen als auch traditionellen Netzwerkbetriebssystemen bemühen sich um die Implementierung von SONiC auf ihren Betriebssystemen und treiben damit die mögliche Orchestrierung verschiedenster Software und Hardware-Typen in der Cloud-Infrastruktur voran. Unter anderem bemüht sich Dell um eine Integration in OS10 und Big Switch Networks um eine Implementierung in das Open-Source-Projekt OpenNetworkLinux. Sogar Cisco hat Microsoft Bemühungen zur Portierung von SAI und SONiC auf das NX-OS der Nexus 9500 und Nexus 9300 Switch-Plattformen zugesprochen, was auf eine spannende Zukunft dieser beiden Projekte hindeutet. [53]

6.7. Facebook FBOSS

Facebook hat für seine eigenen Datacenter-Switches Wedge und Wedge 100 das FBOSS NOS entwickelt, welches jedoch kein vollwertiges Betriebssystem ist, sondern nur eine Sammlung von Applikationen, die auf einem Linux-Betriebssystem laufen und die für Facebook notwendigen Netzwerkfeatures unterstützt. Das Projekt wurde von Facebook auf Github veröffentlicht und wird seitdem auch aktiv von der Community weiterentwickelt. Durch die modulare Architektur ist FBOSS sowie auch SONiC nicht an das Betriebssystem gebunden und kann von Entwicklern auf andere Distributionen und NOS portiert werden. Außerdem ist es möglich und von Facebook auch gewünscht, FBOSS oder Teile davon als Grundlage für weiterführende Projekte zu verwenden und somit die Entwicklung neuer Open-Source-Projekte zu ermöglichen. Das Ziel von Facebook ist es, für FBOSS eine große Liste an kompatibler Hard- und Software zu erhalten und mithilfe der Community eine große Vielfalt an lauffähiger Software auf Basis von FBOSS zu erstellen, weshalb die Weiterentwicklung der Toolsammlung auch vom Open Compute Project koordiniert wird. [54]

Da Facebook die selbstentwickelten Switches und FBOSS im Produktionssystem in den eigenen Rechenzentren im Einsatz hat, wurde auch die Architektur auf die Gegebenheiten und Anforderungen von Facebook angepasst. Die Netzwerk-Topologie des Unternehmens und FBOSS sind auf ein einfaches Feature-Set abgestimmt, weshalb FBOSS nur die für Facebook notwendigen Netzwerkfunktionen beherrscht. Da das Unternehmen viele Layer2-Features wie Spanning-Tree oder Layer2-Overlays nicht benutzt, wurden sie auch nicht in den FBOSS-Agent integriert, aber von der Community entwickelt. Für die Kommunikation mit der Switching-ASIC verwendet Facebook die OpenNSL-API von Broadcom [24], weshalb der Agent zurzeit nur mit Switches, deren ASIC von dieser API unterstützt wird, kompatibel ist. Im Grunde lässt sich die Architektur von FBOSS in drei Bereiche gliedern. Der FBOSS-Agent-Dienst ist für die Kommunikation mit der ASIC über OpenNSL-APIs zuständig und wird mit einem JSON-File, welches sich unter dem Pfad „fboss/agent/configs“ befindet, konfiguriert. Beispielkonfigurationen befinden sich im Github-Repository von FBOSS. Der Routing-Dienst ist für die Forwarding-Table im ASIC zuständig und wird vom FBOSS-Agent gesteuert. Ursprünglich beinhaltet FBOSS keinen Layer3-Stack, der mit FBOSS kommunizieren könnte, da die Lösung, welche von Facebook verwendet wird, laut eigenen Aussagen für die Community nicht hilfreich wäre. Facebook nennt als Grund dafür die Tatsache, dass der hauseigene Layer3-Stack auf die eigenen Bedürfnisse angepasst wurde. Auf Github wurde jedoch schon ein Lösungsvorschlag angenommen, durch welchen eine Interaktion zwischen Layer3-Stacks wie Quagga und FBOSS möglich wird. Ähnlich wie bei OPX könnten Routen über den Linux Netlink Manager an FBOSS weitergeleitet werden. Der dritte Bereich sind Management-Tools, welche ebenfalls nicht veröffentlicht wurden, um Statusmeldungen für das Monitoring, die Generation von Konfigurations-Dateien und das Debugging zu ermöglichen. [54] [55]

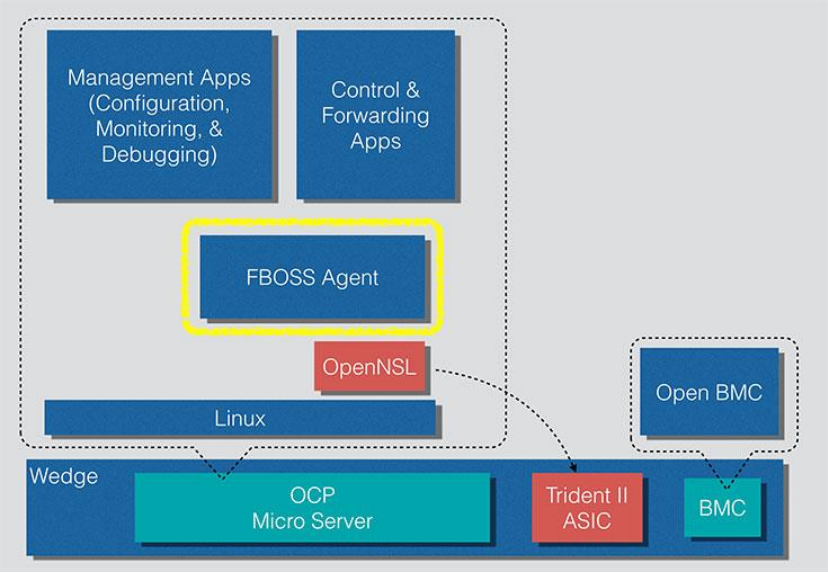

Abbildung 15 - FBOSS Architektur [54]

Da FBOSS zwar bereits im Rechenzentrum von Facebook im Einsatz ist, jedoch nur teilweise veröffentlicht wurde, ist das Betriebssystem im aktuellen Zustand nicht für Produktivsysteme geeignet. Auch die Dokumentation ist selbst für Testzwecke nicht verwendbar, da man nur Beispielkonfigurationen ohne Kommandoreferenz oder Kommentaren einsehen kann. Derzeit ist das Projekt noch in der Entwicklungsphase und wird es auch noch einige Zeit bleiben, da der Layer3-Stack und die Management-Tools erst entwickelt werden müssen. Der Ansatz, welcher von FBOSS verfolgt wird, ähnelt jenem von SONiC, wobei die Liste der kompatiblen Hardware derzeit noch relativ klein ausfällt und das Projekt noch nicht so fortgeschritten ist wie jenes von Microsoft.

7. Hands-on

7.1. Vorstellung der Testumgebung und Testszenarien

Um die wichtigsten Unterschiede der Syntax, Bedienung und die verschiedenen Möglichkeiten der ausgewählten NOS aufzuzeigen, wurden Tests mit Open Networking-Hardware und einigen, mit diesem Switch kompatiblen, NOS durchgeführt. Bei der Auswahl der Hardware wurde darauf geachtet, dass möglichst viele Open Networking Betriebssysteme kompatibel sind. Um diese Tests durchzuführen, wurden Teststellungen der Hersteller der Betriebssysteme zur Verfügung gestellt. Für einige der nachfolgend vorgestellten Testszenarien werden zwei Switches benötigt. Hier kann zwar die Konfiguration vorgenommen werden, abschließende Tests fehlen aber in diesen Fällen aufgrund der fehlenden Hardware.

7.1.1. Testumgebung

Open Networking Hardware und Software

Als Open Networking Switch wurde ein Dell S4048-ON ausgewählt, welcher von der Fachhochschule als Teststellung zur Verfügung gestellt wurde. Der Dell Switch ist ein 10/40GbE top-of-the-rack Switch mit Open Networking Funktion der ON-Serie. Dieser Switch hat bereits das Open Network Install Environment installiert und ist mit vielen verschiedenen Betriebssystemen kompatibel. Die in dieser Testumgebung verwendeten Betriebssysteme wurden von den jeweiligen Herstellern zur Verfügung gestellt. Leider haben nicht alle angefragten Hersteller eine Testversion zur Verfügung gestellt, wodurch es zur Auswahl der folgenden NOS gekommen ist:

- Cumulus Network Linux
- Pica8 PicOS
- Ipinfusion OcNOS

Der Datacenter-Switch wurde von Dell wie alle anderen Komponenten mit der Endung „–ON“ für Open Networking entworfen und auf die Verwendung im SDN optimiert. Software Defined Networking ist nicht in den nachfolgenden Testszenarien enthalten. Er bietet 48 SFP+ (10 GbE) und 6 QSFP+ (40 GbE) Ports, welche jedoch in der Testumgebung nur mit Gigabit-Ethernet-RJ45-SFPs ausgestattet und an Gigabit-Ethernet-Ports eines PCs angeschlossen wurden. [7]

Zusätzliche Hardware und virtuelle Hardware

Für Kompatibilitätstests wurde ein Cisco Catalyst 2960G 24-Port Switch und ein mittels GNS3 virtualisierter Cisco 3745 Router verwendet.

7.1.2. Testszenarien

VLAN und Security

Um den Einsatz von Open Networking im Campus-Bereich zu evaluieren, wurden drei grundlegende und simple Konzepte aus dem Themengebiet VLAN und Security ausgewählt. Diese Beispiele finden sich in fast jedem Campus-Netzwerk und sogar in älteren Datacenter-Netzwerken, auch in abgewandelter Form, wieder. Um Open Networking im Büronetz einsetzen zu können, müssen alle drei Szenarien umsetzbar sein, da ansonsten grundlegende Funktionen und Sicherheitsfeatures verloren gehen und deshalb ein Einsatz in diesem Netzwerksegment nicht möglich wäre. Alle drei Beispiele bauen aufeinander auf und bilden zusammen eine sichere Campus-Umgebung.

VLAN Bridge

Eine der grundlegendsten und einfachsten Konfigurationen ist das Erstellen von VLANs auf einem Switch. Um dieses sehr simple Szenario etwas zu erweitern, wurde - sofern eine Virtualisierung möglich war - ein zusätzlicher Switch über einen Trunk-Port an die bestehende VLAN Bridge angebunden. Pakete, welche auf einem Access-Port, der nur ein VLAN übertragen kann, ankommen, müssen nicht mit einem VLAN-Tag versehen sein, da diese dann intern im Switch getaggt werden. Über einen Trunk werden mehrere verschiedene VLANs übertragen, weshalb sie hier mit einem Tag versehen werden müssen.

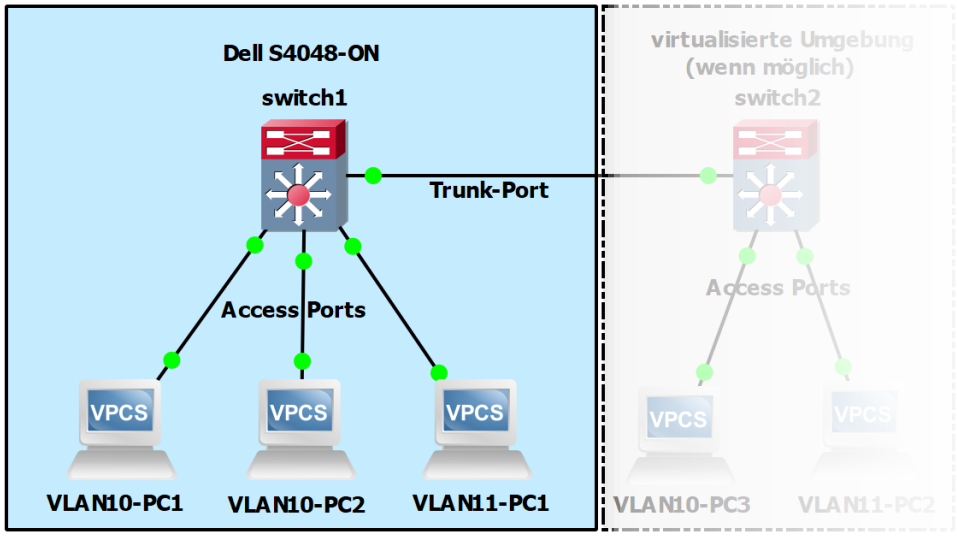

Abbildung 16 - VLAN und Security - VLAN Bridge

In diesem Beispiel führt der Trunk zu einem zweiten Switch, welcher jedoch nur dann konfiguriert wurde, wenn eine Möglichkeit zur Virtualisierung gegeben war. Ansonsten werden die Konfigurationen beider Switches angegeben, jedoch nur der Switch1 konfiguriert. Abschließend wurden Ping-Tests zwischen den verschiedenen virtuellen PCs in unterschiedlichen und gleichen VLANs durchgeführt, um die Funktionalität zu überprüfen.

Absicherung der Switchports nach CCNA Security Chapter 6 Lab A

Alle Switchports im Access-Layer sollten entsprechend abgesichert werden, um Attacken auf den Spanning-Tree und den Switch selbst zu unterbinden. Da diese Sicherheitsfeatures nicht überall einzusetzen sind, müssen sie selbst konfiguriert werden. Im CCNA Security gibt es eine Laborübung zur Absicherung von Layer 2 Switches, welches hier als Best-Practice und Vorlage für die Konfiguration und Absicherung der Switchports dient. Keine der in dieser Übung angegebenen Sicherheitsfeatures sind proprietär, weshalb diese Beispielkonfiguration auch auf andere Hersteller und Betriebssysteme anwendbar ist. Zur Durchführung der Evaluierung, wurden alle angegebenen Sicherheitsfeatures, soweit möglich, auf allen Access- und Trunk-Ports konfiguriert und überprüft.

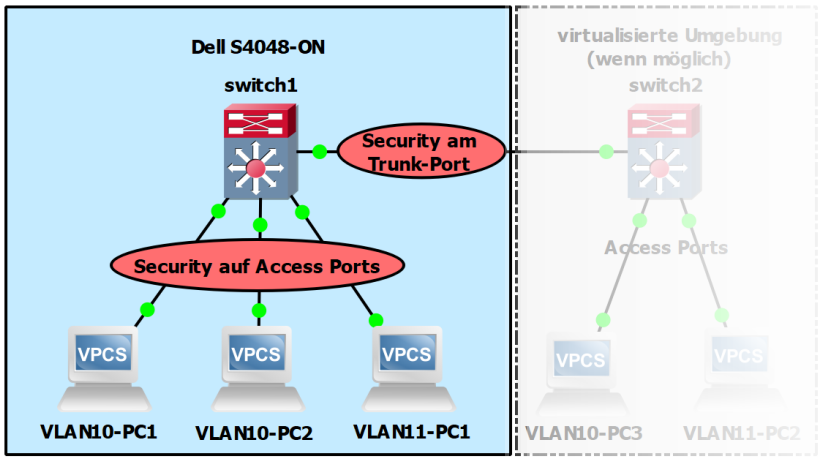

Abbildung 17 - VLAN und Security - Absicherung der Switchports

In Kapitel 6 Laborübung A, Securing Layer 2 Switches, geht Cisco von einer bereits bestehenden Konfiguration von zwei Switches, miteinander verbunden durch einen Trunk-Port, und von bereits konfigurierten Access-Ports aus.

Zur Absicherung des Trunk-Ports zwischen den beiden Switches und zum Schutz vor VLAN-Hopping empfiehlt Cisco folgende Maßnahmen, welche für die Tests zusätzlich noch adaptiert wurden:

- Nur wirklich benötigte Trunk-Ports auch als solche definieren und alle anderen Ports als Access-Ports konfigurieren.
- Das native-vlan, also das VLAN, auf welchem der nicht getaggte Netzwerkverkehr am Trunk übertragen wird, auf ein beliebiges anderes VLAN ändern. In diesem Beispiel wurde dafür das VLAN 99 gewählt.
- Auf den benötigten Trunk-Ports das DTP (Dynamic Trunking Protocol) deaktivieren.
- Nur die Übertragung der wirklich benötigten VLANs auf dem Trunk-Port erlauben.
- Storm-Control für Broadcast-Pakete am Trunk-Port auf 50% setzen und damit alle Broadcasts, die über 50% des Netzwerkverkehrs verursachen, verwerfen.

Die von Cisco empfohlenen Sicherheitsmaßnahmen auf den Access-Ports des Switches dienen vor allem zum Schutz vor Spanning-Tree-Attacken, Spoofing und zur einfachen MAC-Authentifizierung:

- Alle Switchports, die nicht benötigt werden, in den „shutdown“ Status setzen.
- Trunking auf allen Access-Ports deaktivieren und stattdessen den Portmodus auf Access setzen.
- PortFast (auch PortEdge, AdminEdge) auf allen Access-Ports konfigurieren, damit der Port schneller in den Status „up“ wechselt und der Switch diesen Port nicht in den Spanning-Tree aufnimmt.
- Absicherung aller Access-Ports mit dem BPDU-Guard, wodurch ungewollte Netzwerkerweiterungen durch Switches auf Access-Ports automatisch ein Deaktivieren des Ports verursachen (err-disable).
- Optional die Konfiguration von Port-Security und das Erlauben von bestimmten MAC-Adressen auf Access-Ports.

Abschließend wurden Tests mit einem Switch durchgeführt, welcher auf einem Access-Port angeschlossen wurde, um die Funktion des BPDU-Guards zu testen.

802.1x Authentifizierung mit lokalem RADIUS-Server

In Campus-Netzwerken ist 802.1x bereits der Standard zur sicheren Benutzer- und Computerauthentifizierung. Die Authentifizierung erfolgt meist mittels Computerzertifikat und/oder Benutzerdaten aus dem Active-Directory. Die eigentliche Authentifizierung erfolgt am Authentifizierungsserver, welcher die übermittelten Daten prüft und diese Information an den Switch weiterleitet, welcher den Access-Port dann autorisiert oder deaktiviert. Dadurch wird sichergestellt, dass nur berechtigte Geräte Zugang in das Firmennetzwerk erhalten. Durch eine automatische VLAN-Zuweisung erfolgt zusätzlich noch die richtige Zuordnung von Client in das richtige VLAN. Zur Authentifizierung wird leider immer noch TACACS+ oder RADIUS eingesetzt, welche den Datenaustausch unverschlüsselt und über UDP durchführen.

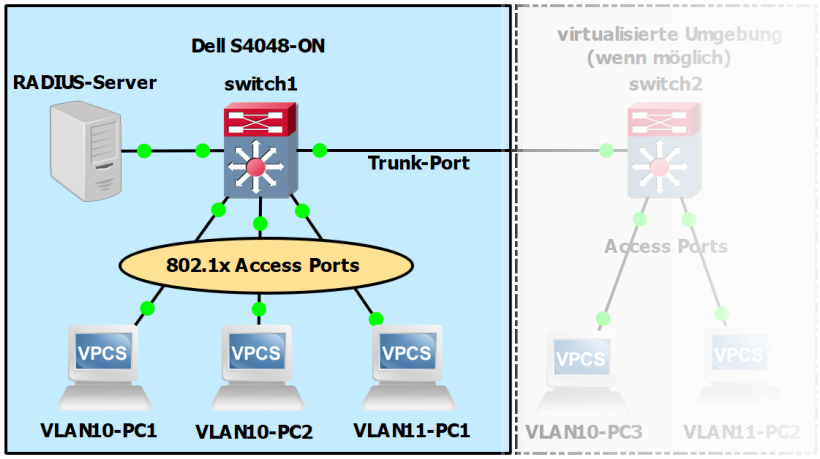

Abbildung 18 - VLAN und Security - 802.1x

In diesem Testszenario wurde FreeRadius auf einem Debian-Server installiert und eine lokale Benutzerdatenbank anstatt einer Active-Directory-Anbindung konfiguriert. Zusätzlich wurden den Benutzern noch VLAN-IDs für die dynamische VLAN-Zuweisung zugeordnet.

```
root@debian#
apt-get install freeradius-utils freeradius-common
vim /etc/freeradius/users.conf
>    #VLAN 10 Client Configuration
>    v10user Cleartext-Password := "secret10"
>         Service-Type = Framed-User,
>         Tunnel-Type = VLAN,
>         Tunnel-Medium-Type = "IEEE-802",
>         Tunnel-Private-Group-ID = 10
>    #VLAN 11 Client Configuration
>    v11user Cleartext-Password := "secret11"
>         Service-Type = Framed-User,
>         Tunnel-Type = VLAN,
>         Tunnel-Medium-Type = "IEEE-802",
>         Tunnel-Private-Group-ID = 11
```

Dot1x ist oft eine grundlegende Anforderung an ein sicheres Firmennetzwerk und muss von einem NOS zwingend unterstützt werden, damit dieses überhaupt erst im Campus-Bereich eingesetzt werden kann.

LAG / MCLAG

Da Open Networking vor allem im Datacenter zum Einsatz kommt und mehr auf dieses hochverfügbare Netzwerksegment als auf herkömmliche Büronetze abzielt, muss auch die Konfiguration von typischen Datacenter-Szenarien überprüft werden. Um größere, typische Netze aufzubauen, fehlte leider die notwendige Hardware, deshalb wurden nur sehr simple Datacenter-Anwendungsbeispiele definiert.

Diese beiden Konzepte bilden aber die grundlegende Basis für alle weiteren modernen Lösungen, wie zum Beispiel Spine/Leaf oder VPC. Redundanz, Ausfallsicherheit und Aggregation sind im Datacenter unabdingbar und werden auch von Open Networking-Lösungen unterstützt, da diese vor allem auf den Einsatz im Rechenzentrum abzielen. Die Konfiguration wurde, wenn dies möglich war, mithilfe eines virtuellen Switches als Gegenstück zur echten Hardware vorgenommen. Wo dies nicht möglich war, wurde zwar die Konfiguration des vorhandenen Geräts durchgeführt, die des Pendants wurde dann lediglich erklärt.

Link Aggregation Group (LAG)

Die Zusammenfassung von mehreren physikalischen Verbindungen zu einer einzigen Verbindung zwecks Ausfallsicherheit und Bandbreitenerhöhung wird vorwiegend, aber nicht ausschließlich, im Datacenter eingesetzt. Dieses Konzept ist unter vielen verschiedenen Namen, wie PortChannel, EtherChannel, Bond, Ling Aggregation (LAG) usw. bekannt. Als grundlegende Technik muss jedes NOS eine Konfiguration ermöglichen, um Ausfallsicherheit zu gewährleisten. Auch hier wurde der zweite Switch virtualisiert oder, wenn dies nicht möglich war, zumindest dessen Konfiguration erklärt und beschrieben.

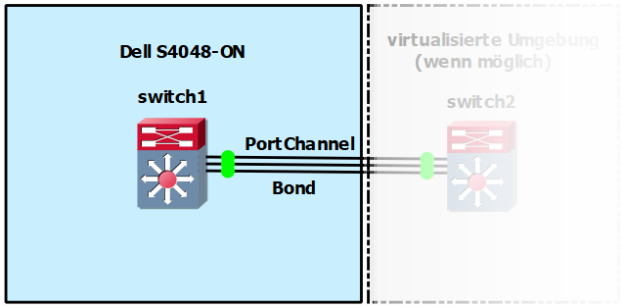

Abbildung 19 - LAG/MCLAG - Link Aggregation

MultiChassis-LAG

Dieses Konzept einer Anbindung eines Servers oder Switches wird von jedem Hersteller anders benannt. Unter einer Multi Chassis Link Aggregation Group (MCLAG) versteht man grundlegend auch virtual PortChannel, MCLAG, Distributed Trunking oder CrossStack EtherChannel. Dabei handelt es sich um eine Anbindung eines Geräts über mehrere Verbindungen zu physikalisch eigenständigen Switch-Chassis. Diese kommunizieren wiederum über einen peer-link, welcher mittels Link Aggregation Control Protocol (LACP) gesteuert wird, untereinander. Die beiden Verbindungen werden als ein LAG angesehen und somit als eine einzelne physikalische Verbindung betrachtet. Falls eine der beiden Verbindungen ausfällt, läuft der Datenverkehr trotzdem noch über die andere Verbindung weiter.

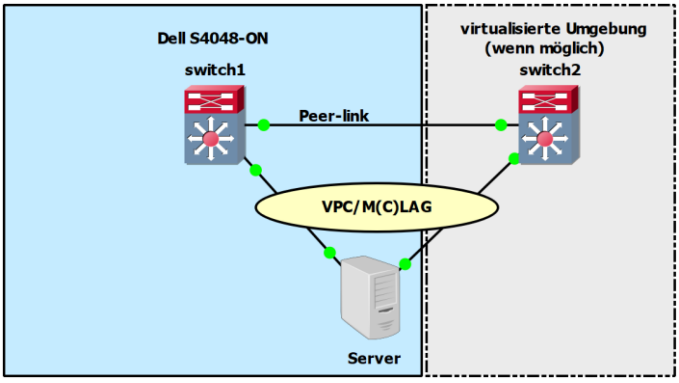

Abbildung 20 - LAG/MCLAG - MultiChassis LAG

Ein MCLAG kann auch aus LAGs zu den Switches bestehen, wodurch noch einmal zusätzlich Ausfallsicherheit gewonnen werden kann. Dadurch, dass sich beide Switches eine MAC-Adresse teilen und die MCLAG als LAG funktioniert, kann im Vorhinein nicht vorausgesagt werden, welchen Weg das Paket nimmt.

Im Datacenter-Umfeld ist diese Technologie bei der Anbindung von Switches und hochverfügbaren Servern notwendig, um Ausfallsicherheit und Redundanz zu gewährleisten. Mithilfe von MCLAG lassen sich komplexe und kritische Netzwerkinfrastrukturen abbilden. Gerade deshalb muss es mithilfe von Open Networking möglich sein, diese Technologie umzusetzen und zu betreiben.
Auch bei diesem Beispiel wurde die Konfiguration auf dem vorhandenen Open Networking-Switch vorgenommen.

Kompatibilität zu traditionellen Systemen (Cisco)

Da sich kein produktives Netzwerk von einem Tag auf den anderen komplett umbauen lässt und die Technologie Open Networking selbst sich gerade in der Transition Phase befindet, ist eine Kompatibilität zu traditionellen Netzwerkkomponenten von unterschiedlichen Herstellern notwendig. Eine Migration von einem traditionellen System auf eine Open Networking-Lösung und ein anderes NOS bedarf guter Planung und Schnittstellen. Wahrscheinlich wird zuerst eine kleine Testinfrastruktur in Betrieb genommen, welche dann in das bestehende Netzwerk eingegliedert werden muss. Hier kann es, bei verschiedenen Herstellern, zu Problemen kommen. Diese Use-Cases sollen die Kompatibilität von Layer2- und Layer3-Verbindungen zwischen einem Cisco-Switch bzw. einem Cisco Router und dem Open Networking-Switch mit verschiedenen Betriebssystemen testen.

VLAN Bridge, Spanning-Tree und shared VLANs

Für den Einsatz von Open Networking und die Kompatibilität im Campus-Bereich, wo zumeist noch VLAN Bridges und der Spanning-Tree mit geteilten VLANs zum Einsatz kommt, muss eine Verbindung über Trunks zwischen Cisco und Open Networking-Switch hergestellt werden. Dieser Trunk wird wie im CCNASecurity Best-Practice mithilfe von verschiedenen Sicherheitsfeatures soweit wie möglich abgesichert.

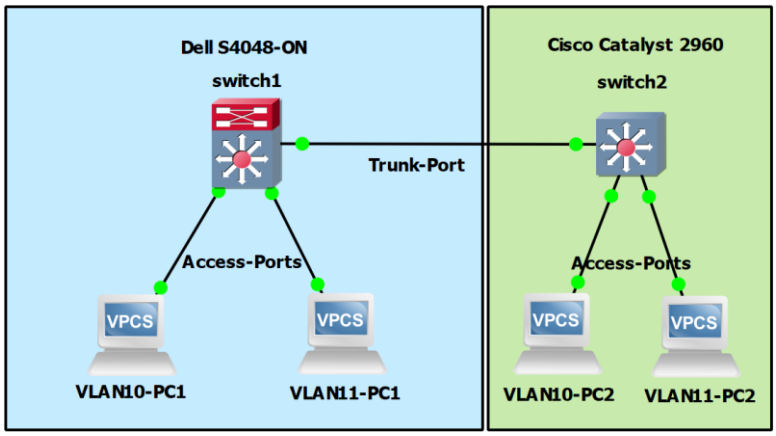

Abbildung 21 - Kompatibilität zu Cisco - VLAN Bridge

Die Herausforderung dabei ist, dass der Spanning-Tree zwischen den beiden Systemen funktioniert und Verbindungen zulässt. Abschließend wurden zum Test der Konfiguration und Kompatibilität Verbindungsversuche (Pings) von und zu VLAN-Clients auf beiden Switches durchgeführt.

Auf dem Cisco 2960 Switch wurde folgende Konfiguration vorgenommen:

```
Switch2#
#VLANs erstellen
vlan 10
vlan 11
#IP-Adressen für VLANs anlegen
interface vlan 10
ip address 10.0.10.2 255.255.255.0
interface vlan 11
ip address 10.0.11.2 255.255.255.0

#Trunk und Sicherheitsfeatures konfigurieren
interface GigabitEthernet0/1
 switchport mode trunk
 switchport trunk allowed vlan 10,11
 switchport trunk native vlan 99
 switchport nonegotiate
 storm-control broadcast level 50
#Access-Ports konfigurieren
interface GigabitEthernet0/10
 switchport mode access
 switchport access vlan 10
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet0/11
 switchport mode access
 switchport access vlan 11
 spanning-tree portfast
 spanning-tree bpduguard enable
```

Layer3 OSPF-Routing

Um eine Implementierung im Datacenter zu testen, könnte eine Layer3-Verbindung zum Einsatz kommen. Diese wird entweder mittels statischer Route oder mithilfe von dynamischen Routing-Protokollen realisiert. Open Shortest Path (OSPF) ist ein herstellerunabhängiges Routing-Protokoll, das für solche Schnittstellen gut geeignet sein könnte. Bei OSPF werden verschiedene Netzbereiche in Areas eingeteilt. Die Router haben jeweils eindeutige IDs, welche an IP-Adressen angelehnt sind. Sie propagieren untereinander ihre Netze und können auch als Gateway zu anderen Netzen dienen. Für dieses einfache Beispiel wurde eine Area (0.0.0.0) erstellt, welche als Backbone-Area fungiert. Der Open Networking-Switch agiert hier als Router und propagiert sein statisch angebundenes Netz (192.168.10.0/24) an alle OSPF-Nachbarn (redistribute connected). Der Cisco-Router propagiert

ebenfalls sein lokales Netz. Die Authentifizierung und Absicherung des OSPF-Routing-Prozesses erfolgt mittels md5-gehashtem Passwort (SUPERSECRETkey).

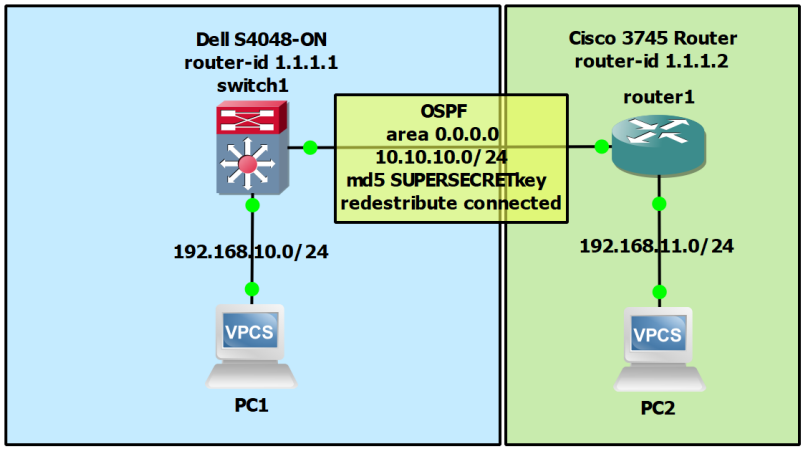

Abbildung 22 - Kompatibilität zu Cisco - OSPF-Routing

Da Cisco bei Routing-Protokollen oft eigene Befehle und Funktionen einbaut, können hier Kompatibilitätsprobleme auftreten. Normalerweise sollte OSPF aber ohne Umstände herstellerübergreifend funktionieren und eine Layer3-Anbindung ermöglichen.

Am virtualisierten Cisco-Router wurde folgende Konfiguration vorgenommen:

```
router1#
interface fastEthernet0/0
 ip address 10.10.10.2 255.255.255.0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 SUPERSECRETkey
 ip ospf network point-to-point
router ospf 1
 router-id 1.1.1.2
 redistribute connected
 passive-interface default
 no passive-interface f0/0
 network 192.168.11.0 0.0.0.255 area 0.0.0.0
```

Layer3 OSPF-Routing IPv6

Da auch im Datacenter IPv6 bereits vereinzelt zum Einsatz kommt, ist es notwendig, auch eine Kompatibilitätsprüfung von OSPF für IPv6 durchzuführen. Der Ablauf und die Logik ist hierbei dieselbe, der einzige Unterschied ist die Verwendung der neueren Version von OSPF und der Konfiguration von IPv6. Dadurch wird auch die Kompatibilität von Open Networking-Betriebssystemen zu IPv6 geprüft. Eine gute Umsetzung und Unterstützung von IPv6 wäre wünschenswert, da damit zukünftige Herausforderungen einfacher gemeistert werden könnten.

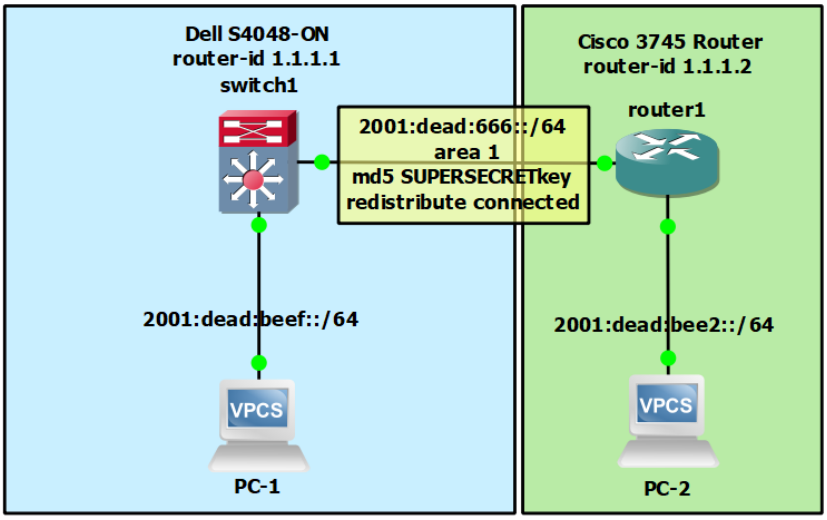

Abbildung 23 - Kompatibilität zu Cisco - OSPF-Routing IPv6

Am virtualisierten Cisco-Router wurde folgende Konfiguration vorgenommen:

```
router1#
ipv6 unicast-routing
interface fastEthernet0/0
 ipv6 address 2001:dead:bee2::1/64
 ipv6 ospf authentication ipsec spi 257 md5 7 SUPERSECRETkey
 ipv6 ipv6 ospf authentication ipsec spi 257 md5 7 SUPERSECRETkey ospf network point-to-point
ipv6 router ospf 1
 router-id 1.1.1.2
 redistribute connected
 passive-interface default
 no passive-interface f0/0
```

7.2. VLAN und Security

7.2.1. Cumulus Network Linux

Auf der Herstellerseite wird mit CumulusVX ein virtueller Switch für GNS3 angeboten, welcher ohne zusätzliche Kosten getestet werden kann. Aus diesem Grund ist hier eine Konfiguration mit zwei Switches möglich. Der zweite Switch wurde mithilfe von GNS3 virtualisiert und an physische bzw. virtuelle Netzwerkadapter angeschlossen.

Unter Cumulus Linux gibt es zwei verschiedene Möglichkeiten, Netzwerkbrücken anzulegen, den „VLAN-aware Bridge Mode“ und den „Traditional Bridge Mode“. Beide Modi werden, wie unter Linux üblich, in der Datei /etc/network/interfaces verwaltet, beide unterstützen verschiedene Spanning-Tree-Protokolle und die Netzwerkadapter werden mit ifupdown-Kommandos gesteuert. Cumulus selbst legt die Nutzung des „VLAN-aware Bridge Mode“ nahe, da mehr VLANs gleichzeitig betrieben werden können und die Konfigurationssyntax einfacher ist. Auf der Internetseite des Herstellers werden als Gründe für die Verwendung des traditionellen Modus nur PVSTP+, welches nichg im neueren Modus nicht unterstützt wird, und die Linux Syntax genannt. Im „VLAN-aware Bridge Mode“ kommt ausschließlich das schnellere RSTP-Protokoll zum Einsatz. Um eine Verbindung mithilfe eines Kupfer-SFPs zustande zu bekommen, muss auto-negotiation aktiviert werden, da ansonsten für jeden Port alle Einstellungen wie speed, duplex, pause, etc. selbst vorgenommen werden müssten. [56]

```
net add interface swp1-52 link autoneg on
net commit
```

Die gesamte in allen nachfolgenden Beispielen gezeigte Konfiguration wurde mithilfe des „Cumulus Linux User Guide“ erstellt. Die Dokumentation ist mit einigen Beispielen ausgestattet und bietet viele Hintergrundinformationen. [57]

VLAN Bridge

Zur Konfiguration von „VLAN-aware Bridges“ kann das NCLU verwendet werden, wodurch die Konfiguration abermals um einiges vereinfacht wird. Die Konfiguration der VLANs und der Netzwerkbrücke ist einfach und selbsterklärend, wodurch es leichtfällt, diese vorzunehmen. Die nachfolgende Konfiguration wurde sowohl auf der echten Hardware als auch am virtuellen Switch (CumulusVX) vorgenommen.

```
#Cumulus NCLU aufrufen
nclu

#VLANs erstellen und L3 IP-Adressen anlegen
net add vlan 10-11
net add vlan 10 ip address 10.0.10.1/24
net add vlan 11 ip address 10.0.11.1/24

#Die Ports zur VLAN-aware Bridge hinzufügen
net add bridge bridge ports swp1,swp10-11

#Switchport 1 als Trunk definieren und erlaubte VLANs definieren
net add int swp1 bridge trunk vlans 10-11

#VLANs auf die Bridge binden
net add bridge bridge vids 10,11

#Access Ports konfigurieren
net add interface swp10 bridge access 10
net add interface swp11 bridge access 11

#Überprüfung und Bestätigung
net pending
net commit
```

Absicherung der Switchports nach CCNA Security Chapter 6 Lab A

Die im CCNA Security Lab angegebenen Sicherheitsparameter konnten fast zur Gänze umgesetzt werden. Lediglich das Deaktivieren von DTP und das optionale MAC-Whitelisting ist nicht möglich, wobei dafür auch auf Linux-Tools zurückgegriffen werden könnte. Das Feature Port-Fast wurde von Cumulus erweitert und wird hier PortAutoEdge genannt. Diese Funktion ermöglicht, dass der Switch selbst entscheidet, ob es sich hierbei um einen Access-Port handelt oder nicht. Das Whitelisting von einzelnen VLANs am Trunk-Port ist hier hinfällig, da dies für jeden Trunk und jede Bridge, wie im vorigen Beispiel konfiguriert, vorgenommen werden muss. Cumulus setzt alle Interfaces automatisch in den „up“-Status, weshalb alle nicht gebrauchten Interfaces händisch ausgeschaltet werden müssen.
Storm-Control ist nicht mit der NCLU zu konfigurieren, hier nimmt man die Einstellungen in der switchd-Konfigurationsdatei (/etc/switchd.conf) vor. Dieser Parameter lässt sich aber nicht wie auf Cisco-Switches auf eine Prozent-Angabe einstellen, sondern muss mit Paketen-pro-Sekunde (pps) limitiert werden. Ebenso muss Storm-Control für jeden physikalischen Port einzeln angepasst und danach der Dienst switchd neugestartet werden.

Die Tatsache, dass für ein Feature Set, wie die Absicherung von Switchports, verschiedene Konfigurationsmöglichkeiten bestehen und verwendet werden müssen, verkompliziert die Abläufe. Möglicherweise werden Funktionen wie Storm-Control noch in die NCLU implementiert, jedoch ist das Setup mit den jetzigen Möglichkeiten mühsam. Es bleibt zu hoffen, dass Cumulus alle konfigurierbaren Parameter in das NCLU aufnimmt und daraus eine zentrale Stelle für die Einstellungen macht.

Konfiguration der Sicherheitsfeatures an einem Trunk-Port:

```
#Erstellen des Native-Vlan:
net add vlan 99

#PVID ist der Primary VLAN Identifier, also das Native VLAN – standardmäßig 1
net add bridge bridge pvid 99

#Storm-Control für Broadcasts konfigurieren
vi /etc/cumulus/switchd.conf
> interface.swp1.storm_control.broadcast = 400

#Bestätigung
net commit

net show bridge vlan
Interface    VLAN    Flags
swp1         10-11
             99      PVID, Egress Untagged
```

Konfiguration der Sicherheitsfeatures an einem Access-Port:

```
#Unbenutzte Interfaces in den “down”-Status versetzen
net add interface swpX(-Y) link down

#BPDU-Guard und PortAutoEdge (PortFast) am Access-Port aktivieren
net add interface swpX stp bpduguard
net add interface swpX stp portautoedge

#Test des BPDU-Guard mit einem Cisco Switch
sudo tail -f /var/log/syslog
2018-03-31T21:32:41.877915+00:00 switch1 mstpd: error, MSTP_IN_rx_bpdu: bridge:swp10 Recvd BPDU on BPDU Guard Port -
Port Down
2018-03-31T21:32:41.878928+00:00 switch1 mstpd: set_if_up: Port swp10 : down
2018-03-31T21:32:41.879430+00:00 switch1 mstpd: MSTP_OUT_set_state: bridge:swp10:0 entering blocking    state(Disabled)

#Error-Disabled Interface wieder einschalten
ifup swp10
```

802.1x-Atuhentifizierung mit lokalem Radius Server

Das Dot1X-Feature kann unter Cumulus nur auf Switches mit einem Broadcom-ASIC vorgenommen werden. Außerdem kann 8021.x nur im „VLAN-aware Bridge Modus“ verwendet werden und die Authentifikation erfolgt ausschließlich mit RADIUS. In der unten abgebildeten Grafik aus dem Cumulus-User-Guide wird zwar auch DIAMETER als Protokoll angegeben, jedoch gibt es keine Konfigurationsbeispiele oder Anmerkungen diesbezüglich. Außerdem empfiehlt Cumulus, den RADIUS-Server nicht auf dem Switch selbst zu installieren, obwohl es sich hier um ein funktionsfähiges Linux-Betriebssystem handelt.

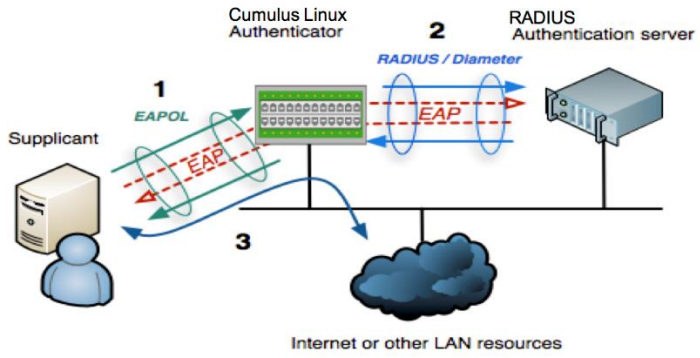

Abbildung 24 - 802.1x Cumulus Linux [57, p. 271]

Um 802.1x unter Cumulus zu aktivieren, muss zuerst der Dienst hostapd (Host access point Deamon) installiert werden, welcher hier verwendet wird, um den Zugriff zu regeln. Erst dann ist die Verwendung von dot1x möglich. Die eigentliche Konfiguration von 802.1x unter Cumulus ist jener von traditionellen NOS wieder ähnlich, da hier auch das NCLU verwendet wird. Alle mit dot1x zu authentifizierenden Ports müssen derselben Bridge zugeordnet sein. Wenn eine dynamische VLAN-Zuweisung gewünscht ist, muss diese ebenfalls mit einem einfachen Befehl aktiviert werden. Um eine Quarantäne-VLAN für unautorisierte Ports einzurichten, muss zuerst ein sogenanntes „parking-VLAN“ angelegt werden. Die Konfiguration ist sehr schlicht gehalten und kommt gänzlich ohne Linux-Konfigurationsdateien aus, wobei man die Konfiguration auch in den Konfigurationsdateien vornehmen könnte.

```
#Hostapd installieren
sudo -E apt-get update
sudo -E apt-get install hostapd
sudo -E apt-get upgrade

#Alle zu authentifizierenden Ports einer Bridge hinzufügen
net add bridge bridge ports swp10-11

#Radius Server und Shared-Secret konfigurieren
net add dot1x radius server-ip <RADIUS-IP>
net add dot1x radius shared-secret SUPERSECRETkey

#Parking VLAN einstellen
net add vlan 666
net add dot1x parking-vlan-id 666

#Dynamische VLAN Zuweisung
net add dot1x dynamic-vlan required

#Dot1X aktivieren
net add interface swp10-11 dot1x

#Bestätigung
net commit
```

Leider hat die automatische VLAN-Zuweisung mit 802.1x-Authenifizierung nicht funktioniert. Das Problem lag bei der Kommunikation zwischen FreeRADIUS und Cumulus. Laut Fehlermeldung sendet der RADIUS-Server das VLAN-Tag nicht mit der Erfolgs-Meldung mit, wodurch der Port keinem VLAN zugeordnet werden kann. Der PC (Windows 10 Client) wird zwar zur Authentifizierung aufgefordert, jedoch scheitert die Authentifizierung danach. Der Ablauf der fehlgeschlagenen Authentifizierung ist im Logfile von hostapd protokolliert.

```
sudo journalctl -f -u hostapd

Mar 31 22:36:08 cumulus hostapd[20453]: Handle_data:recv: packet on swp1
Mar 31 22:36:08 cumulus hostapd[20453]: swp1: STA fc:aa:14:2d:e8:46 IEEE 802.1X: authentication server did not include
required VLAN ID in Access-Accep
Mar 31 22:36:08 cumulus hostapd[20453]: swp1: STA fc:aa:14:2d:e8:46 IEEE 802.1X: authentication failed - EAP type: 25
(PEAP)
Mar 31 22:36:13 cumulus hostapd[20453]: swp1: STA fc:aa:14:2d:e8:46 IEEE 802.11: deauthenticated due to local deauth
request
```

Nach dem Deaktivieren der dynamischen VLAN-Zuweisung (net del dot1x dynamic-vlan) und der statischen Zuweisung der VLANs an den Access-Ports funktionierte die Authentifizierung.

```
sudo journalctl -f -u hostapd

Mar 31 22:56:32 cumulus hostapd[21667]: Handle_data:recv: packet on swp4
Mar 31 22:56:32 cumulus hostapd[21667]: swp4: CTRL-EVENT-EAP-SUCCESS2 fc:aa:14:2d:e8:46
Mar 31 22:56:32 cumulus hostapd[21667]: swp4: AP-STA-CONNECTED fc:aa:14:2d:e8:46
Mar 31 22:56:33 cumulus hostapd[21667]: Installing acl policy
Mar 31 22:56:33 cumulus hostapd[21667]: swp4: STA fc:aa:14:2d:e8:46 RADIUS: starting accounting session 6D2AB8DECB4F9
Mar 31 22:56:33 cumulus hostapd[21667]: swp4: STA fc:aa:14:2d:e8:46 IEEE 802.1X: authenticated - EAP type: 25 (PEAP)
```

7.2.2. Pica8 PicOS

PicOS kann entweder im L2/L3 Mode oder im OVS Mode betrieben werden. Im L2/L3 Mode können herkömmliche Funktionen, wie Routing- oder Switching-Protokolle, eines Switches, direkt am Switch konfiguriert werden. Im OVS Mode (Open vSwitch Mode), wird PicOS für OpenFlow Applikationen optimiert und betrieben. In diesem Modus gibt es keine Möglichkeit, von Konfigurationen am Switch selbst. Diese werden über OpenFlow von OpenvSwitch übertragen und dort konfiguriert bzw. betrieben. Für die nachfolgenden Szenarien wurde der L2/L3-Modus verwendet. Die Konfiguration wurde mithilfe der PicOS-Dokumentation erstellt und basiert teilweise darauf, außerdem findet sich hier eine Kommandoreferenz und weitere Informationen zu PicOS. [51]

Auch PicOS benutzt für die Konfiguration mehrere verschiedene Command-Line-Interfaces. Zum einen kann die Linux Shell und zum anderen die bereitgestellte PicOS CLI verwendet werden, jedoch unterstützt die CLI alle notwendigen Kommandos, um eine vernünftige Konfiguration vorzunehmen. Mit dem Bash-Kommando „cli“ wechselt man in die CLI, welche die Konfiguration erleichtert. Sie ist an eine klassische CLI angelehnt, wurde aber um ein paar Funktionen erweitert. So müssen wie bei Cumulus Linux alle Änderungen mit dem Befehl „commit“ bestätigt werden. Zusätzlich gibt es hier die Möglichkeit, die Konfiguration nur auf Zeit (10 Minuten) zu übernehmen. Alle Parameter werden mit „set“ gesetzt und mit „del“ wieder gelöscht. Die Konfiguration wird im JSON-Format ausgegeben, was etwas gewöhnungsbedürftig scheint. Es gibt aber auch die Möglichkeit, die eingegebenen Kommandos auszugeben. Außerdem werden Statusinformationen mit einem vorangestellten „run“ ausgegeben, während ein einfaches „show“-Kommando die aktuelle Konfiguration im JSON-Format liefert. Ein „show interfaces“ gibt also die Konfiguration aller Interfaces aus, während ein „run show interfaces“ aktuelle Statusinformationen zu den Interfaces anzeigt.

```
show
    interface {
        gigabit-ethernet "te-1/1/1" {
            ether-options {
            }
            family {
                ethernet-switching {
                    native-vlan-id: 1010
        ……

run show  version
 Copyright (C) 2009-2018 Pica8, Inc.
 Base ethernet MAC Address     : 14:18:77:01:9f:01
 Hardware Model                : DELL S4048-ON
 L2/L3 Released Date           : 01/19/2018
  ……
```

Da Pica8 keine virtuelle Appliance seiner Switchsoftware zur Verfügung stellt, konnten die nachfolgenden Tests nur mit einem Switch durchgeführt werden. In der Testversion, die von Pica8 zur Verfügung gestellt wurde, sind nur die ersten 4 Switchports frei konfigurierbar, alle anderen sind gesperrt und können mit der entsprechenden Lizenz freigeschaltet werden.

VLAN Bridge

Beim Anlegen von VLANs unterscheidet sich PicOS nicht maßgeblich von anderen NOS, lediglich die Syntax ist naturgemäß anders. Die Konfiguration selbst fällt etwas schwerfälliger aus, da immer der gesamte Konfigurationspfad angegeben werden muss. Auch das Löschen von Einträgen ist relativ aufwendig, da jeder abgesetzte Befehl einzeln rückgängig gemacht werden muss. Um einem Access-Port ein VLAN zuzuweisen, wird hier der Befehl „native-vlan-id“ verwendet, welcher universell für Access-Port und Trunk einsetzbar ist.

```
#PicOS-CLI aufrufen
cli

#In den Konfigurationsmodus wechseln
configure

#Vlans und L3 Interfaces bzw. IP-Adressen anlegen
set vlans vlan-id 10
set vlans vlan-id 11
set vlans vlan-id 10 l3-interface vlan-10
set vlans vlan-id 11 l3-interface vlan-11
set vlan-interface interface vlan-10 vif vlan-10 address 10.0.10.1 prefix-length 24
set vlan-interface interface vlan-11 vif vlan-11 address 10.0.11.1 prefix-length 24

#Switchport 1 als Trunk und zu übertragende VLANs definieren
set interface gigabit-ethernet te-1/1/1 family ethernet-switching port-mode trunk
set interface gigabit-ethernet te-1/1/1 family ethernet-switching vlan members 10
set interface gigabit-ethernet te-1/1/1 family ethernet-switching vlan members 11
set interface gigabit-ethernet te-1/1/1 disable false

#Accss-Ports und Access-VLANs definieren
set interface gigabit-ethernet te-1/1/3 family ethernet-switching port-mode access
set interface gigabit-ethernet te-1/1/3 family ethernet-switching native-vlan-id 10
set interface gigabit-ethernet te-1/1/3 disable false
set interface gigabit-ethernet te-1/1/4 family ethernet-switching port-mode access
set interface gigabit-ethernet te-1/1/4 family ethernet-switching native-vlan-id 11
set interface gigabit-ethernet te-1/1/4 disable flase

#Änderungen abspeichern
commit
```

Absicherung der Switchports nach CCNA Security Chapter 6 Lab A

Fast alle Security-Einstellungen konnten unter PicOS vorgenommen werden. Bis auf das Deaktivieren von DTP, unter Cisco IOS nonegotiate genannt, gab es keine großen Schwierigkeiten bei der Umsetzung des Best-Practice. Pica8 bietet sogar einen Command-Line-Befehl für das Aktivieren des Features Port-Security, welches für das MAC-Adressen-Whitelisting benötigt wird. Weiterhin ist das automatische Deaktivieren von Spanning-Tree auf den Access-Ports nur mit dem Spanning-Tree-Protokoll MSTP möglich. Das Deaktivieren aller nicht benötigten Switchports ist nicht notwendig, da alle Ports sowieso zuerst eingeschaltet werden müssen.

```
#Native VLAN anlegen
set vlans vlan-id 99

#Trunk native VLAN ändern
set interface gigabit-ethernet te-1/1/1 family ethernet-switching native-vlan-id 99
#Statt nonegotiate wird hier flow-control verwendet
set interface gigabit-ethernet te-1/1/1 ether-options flow-control false
#Storm-Control Broadcast-Ratio konfigurieren
set interface gigabit-ethernet te-1/1/1 storm-control broadcast ratio 50

#BPDU-Guard, PortFast und MAC-Restriction aktivieren
set protocols spanning-tree mvst interface te-1/1/3 bpdu-guard true
set protocols spanning-tree mvst interface te-1/1/3 edge true
set interface gigabit-ethernet te-1/1/3 port-security mac-limit 1
set interface gigabit-ethernet te-1/1/3 port-security sticky true

#BPDU-Guard, PortFast und MAC-Restriction aktivieren
set protocols spanning-tree mvst interface te-1/1/4 bpdu-guard true
set protocols spanning-tree mvst interface te-1/1/4 edge true
set interface gigabit-ethernet te-1/1/4 port-security mac-limit 1
set interface gigabit-ethernet te-1/1/4 port-security sticky true

#Bestätigen
Commit

run show vlans
VlanID   Tag        Interfaces
10       tagged     te-1/1/1
11       tagged     te-1/1/1
99       untagged   te-1/1/1
```

Beim Testen des BPDU-Guards fiel auf, dass unter PicOS eine besondere Vorgangsweise beim Reaktivieren des Switchports notwendig ist. Zuerst muss der BPDU-Guard am Port deaktiviert und das Interface ausgeschaltet werden. Nach einer Bestätigung des Vorgangs muss der BPDU-Guard wieder aktiviert und das Interface anschließend wieder eingeschaltet werden.

```
#Test des BPDU-Guard mit einem Cisco Switch
run show interface gigabit-ethernet te-1/1/3
Physical interface: te-1/1/3, Enabled, error-discard True, Physical link is Down
Source filtering: Disabled, Flow control: Disabled, Auto-negotiation: Enabled
Interface flags: Hardware-Down SNMP-Traps Internal: 0x0
force up mode:false

#Error-Disabled Interface wieder einschalten

#Zuerst Deaktivieren des BPDU-Guards und Deaktivieren des Interfaces inklusive commit
delete protocols spanning-tree pvst interface te-1/1/3 bpdu-guard
set interface gigabit-ethernet te-1/1/3 disable true
commit

#Danach Reaktivieren des BPDU-Guards und Aktivieren des Interfaces inklusive commit
set protocols spanning-tree pvst interface te-1/1/3 bpdu-guard
set interface gigabit-ethernet te-1/1/3 disable false
commit

run show interface gigabit-ethernet te-1/1/3
Physical interface: te-1/1/3, Enabled, error-discard False, Physical link is Up
Source filtering: Disabled, Flow control: Disabled, Auto-negotiation: Enabled
Interface flags: Hardware-Up SNMP-Traps Internal: 0x0
force up mode:true
```

802.1X-Authentifizierung mit lokalem Radius Server

Das letzte Beispiel aus dem Bereich VLAN und Security, 802.1x, ist unter PicOS sehr einfach zu konfigurieren. Im Grunde wird lediglich ein einzelnes Kommando auf der CLI abgesetzt, um die 802.1x-Authentifizierung zu aktivieren. Dot1x muss danach nur noch auf den Switchports selbst aktiviert werden, was wiederum mit einem Deaktivieren des Ausschalters funktioniert. Ob es eine Möglichkeit gibt, ein anderes Protokoll zur Authentifizierung zu verwenden, ist nicht dokumentiert und in der CLI selbst nicht möglich. Die Umsetzung von 802.1x ist jedoch noch nicht perfekt, da es leider keine Möglichkeit gibt, eine automatische VLAN-Zuweisung durchzuführen. In der Dokumentation von Pica8 gibt es leider keine Anhaltspunkte für eine mögliche Konfiguration dieses Features. Ebenso ist es auch nicht möglich, ein „parking“ oder Quarantäne-VLAN zu definieren.

```
#802.1x-Authentifizierung allgemein aktivieren
set protocols dot1x aaa radius authentication server-ip 10.0.0.113 shared-key SUPERSECRETkey

#802.1x-Authentifizierung für jeden Switchport aktivieren
set protocols dot1x interface te-1/1/3 disable false
set protocols dot1x interface te-1/1/4 disable false
#Bestätigen
commit
```

Nach der Konfiguration wird der Switchport authentifiziert und es ist eine Anmeldung erforderlich. Dies geschieht jedoch ausschließlich mit EAP-MD5, wodurch es nicht möglich ist, einen Windows-PC ohne Zertifikat zu authentifizieren. Mit einem Debian-Client und dem Tool „wpa-supplicant“ war eine Authentifikation mit Username und Passwort erfolgreich.

```
run show dot1x interface
te-1/1/3       no supplicant      UNAUTHORIZED
te-1/1/4       fc:aa:14:2d:e8:46  AUTHORIZED

run show dot1x interface gigabit-ethernet te-1/1/4
Dot1x Info for te-1/1/4
------------------------------
  PortEnabled          = true
  PortControl          = AUTO
  QuietPeriod          = 60
  ServerTimeout        = 30
  ReAuthentication     = true
  ReAuthPeriod         = 3600
```

```
Dot1x Authenticator Client
--------------------------
  Supplicant           = fc:aa:14:2d:e8:46
  Port Status          = AUTHORIZED
  Auth SM State        = AUTHENTICATED
  Auth BEND SM State   = IDLE
```

7.2.3. IP Infusion OcNOS

IP Infusions OcNOS bietet zur Konfiguration eine CLI an, welche sehr stark an eine klassische IOS-CLI erinnert. In der CLI gibt es verschiedene Modi, welche mit denen der CLI von IOS gleich sind. Nach dem Login befindet man sich im „executive mode“. Um das Gerät zu konfigurieren, muss in den „privileged executive mode“ und anschließend in den „configure mode“ gewechselt werden. Diese Vorgangsweise ist von allen herkömmlichen NOS bekannt und sollte zu keinen Problemen beim Umstieg führen. Theoretisch gibt es auch die Möglichkeit, eine Anmeldung mit dem „root“-User durchzuführen und somit Parameter direkt am System zu ändern. Da aber OcNOS offensichtlich darauf ausgelegt ist, wie ein traditionelles Betriebssystem ohne viele Neuerungen auszukommen, ist dies nicht notwendig. Die OcNOS-CLI verwendet oft sogar dieselben Kommandos für die Konfiguration wie Cisco. Es fehlt hier aber das von Cumulus und PicOS bekannte „commit“, um die Änderungen zu übernehmen. Ipinfusion setzt die Kommandos direkt nach der Eingabe um, ein „write“ speichert die aktive Konfiguration für den nächsten Neustart.
Die Dokumentation ist nur mit Registrierung und Kauf eines Produkts erhältlich, dafür aber ausführlich beschrieben und vollständig. Sie wurde für alle nachfolgenden Beispielkonfigurationen und Inhalte des praktischen Teils als Referenz verwendet. [47]

VLAN Bridge

Durch die Ähnlichkeit zur IOS-CLI, fällt auch die Konfiguration der VLANs nicht sonderlich schwer aus. Der größte Unterschied liegt darin, dass auch hier zuerst eine bridge erstellt werden muss. Cisco behandelt einen Switch immer als eine Bridge, während OcNOS wie die anderen getesteten NOS die Konfiguration mehrerer Bridges ermöglicht. Mit dem Parameter „ieee vlan-bridge“ wird dem Betriebssystem mitgeteilt, dass es sich hierbei um eine Netzwerkbrücke handelt, die VLANs berücksichtigen muss. Um einen Switchport überhaupt erst verwenden zu können, muss dieser mit dem Kommando „switchport“ als solcher definiert werden. Erst dann sind weitere Konfigurationen, wie die Zugehörigkeit zu einer Netzwerkbrücke und der Switchport-Modus, möglich. Um einem VLAN eine IP-Adresse zuzuordnen, muss ein Subinterface am L3-Interface des VLANs angelegt werden.

```
configure terminal
enable

#Netzwerkbrücke erstellen, die VLANs berücksichtigt
bridge 1 protocol ieee vlan-bridge

#VLANs in der VLAN-Database anlegen
vlan database
 vlan 10 bridge 1 state enable
 vlan 11 bridge 1 state enable

#VLAN L3 Interfaces erstellen
interface vlan1.10
 ip address 10.0.10.1 255.255.255.0
interface vlan1.11
 ip address 10.0.11.1 255.255.255.0

#Switchport Mode setzen Port, dieser muss zuerst als L2-Port (switchport) markiert werden
interface xe1
 switchport
 bridge-group 1
 switchport mode trunk
 switchport trunk allowed vlan all
 no shutdown
```

```
interface xe10
 switchport
 bridge-group 1
 switchport mode access
 switchport access vlan 10
 no shutdown


interface xe11
 switchport
 bridge-group 1
 switchport mode access
 switchport access vlan 11
 no shutdown

show vlan all bridge 1
Bridge  VLAN ID     Name         State   H/W Status      Member ports (u)-Untagged, (t)-Tagged
1       1       default          ACTIVE  Up          xe1(u)
1       10      VLAN0010         ACTIVE  Up          xe1(t) xe10(u)
1       11      VLAN0011         ACTIVE  Up          xe1(t) xe11(u)
```

Absicherung der Switchports nach CCNA Security Chapter 6 Lab A

Obwohl eine starke Ähnlichkeit zu Ciscos IOS gegeben ist, konnten nicht alle im Best-Practice genannten Sicherheitsregeln definiert werden. Obwohl die Anleitung teilweise auf OcNOS anwendbar ist, fehlen einige Befehle. So ist es zum Beispiel nicht möglich, das DTP zu deaktivieren. Außerdem besteht in der CLI selbst keine Möglichkeit, Port-Security und MAC-Whitelisting zu aktivieren. Die Switchports werden in diesem Betriebssystem initial ausgeschaltet und müssen für den Betrieb zuerst eingeschaltet werden. Dies erfolgt wie bekannt mit der Cisco-Logik und dem Befehl “no shutdown”. Im Unterschied zu Cumulus und PicOS werden bei diesem Betriebssystem automatisch alle VLANs auf einem Trunk erlaubt, weshalb ein extra Kommando abgesetzt werden muss, um denselben Effekt zu erzielen.

```
#VLAN 99 anlegen
vlan database
 vlan 99 bridge 1

#Trunk-Port
int xe1
 #Trunk native VLAN-ändern
 switchport trunk native vlan 99
 #Am Trunk erlaubte VLANs definieren
 switchport trunk allowed vlan 10,11,99
 #Broadcast Level auf 50% des Traffics setzen.
 storm-control broadcast level 50

#Access-Ports
int xe10
 #Portfast aktiveren
 spanning-tree portfast
 #BPDU-Guard aktivieren
 spanning-tree bpduguard enable
int xe11
 #Portfast aktiveren
 spanning-tree portfast
 #BPDU-Guard aktivieren
 spanning-tree bpduguard enable
```

Der Test des BPDU-Guards verlief erfolgreich, die vorgenommenen Einstellungen können mit dem Befehl “show spanning-tree” angezeigt werden. Das Reaktivieren eines “error-disabled”-Ports erfolgt durch das Deaktivieren und Reaktivieren des Switchports.

```
show spanning-tree
%   xe10: port Number 914 - Ifindex 5010 - Port Id 0x8392 - Role Disabled - State Discarding
%   xe10: portfast configured - Current  portfast on
%   xe10: bpdu-guard configured  - Current bpdu-guard on
%   xe10: bpdu-filter default  - Current bpdu-filter off
%   xe10: auto-edge configured - Current port Auto Edge on

#Reaktivieren eines gesperrten Switchports
interface xe10
shutdown
no shutdown
```

802.1x-Authentifizierung mit lokalem Radius Server

Die Konfiguration von 802.1x unter OcNOS ist einfach und unkompliziert, leider fehlen aber einige wichtige Funktionen, wie die automatische VLAN-Zuweisung oder ein Quarantäne-VLAN für unautorisierte PCs. Zuerst muss dot1x Systemweit aktiviert und die Konfiguration des Radius-Servers vorgenommen werden, was mit drei einfachen Befehlen erledigt ist. Danach muss für jeden Port einzeln die Authentifizierung aktiviert werden, weil es in diesem Betriebssystem keine Möglichkeit gibt, mehrere Interfaces selbst zu konfigurieren. In der Dokumentation gibt es keine Anhaltspunkte für „interface ranges“, was bei einem Access-Switch mit 48 Ports zu einem relativ hohen Arbeitsaufwand führt.

```
#Dot1x systemweit aktivieren und konfigurieren
dot1x system-auth-ctrl
radius-server host 10.0.0.108
radius-server key SUPERSECRETkey


#802.1x muss auf jedem Interface einzeln aktiviert werden
interface xe10
 dot1x port-control auto
interface xe11
 dot1x port-control auto
…
```

Im Test hat die Authentifizierung des Test-PCs einwandfrei funktioniert. Es bleibt anzumerken, dass sich der Radius-Server nicht im “Out-of-Band”-Netz befinden darf und somit nicht über den Management-Port des Switches erreichbar sein darf.

```
show dot1x all
802.1X Port-Based Authentication Enabled
  RADIUS server address: 10.0.11.17:1812
  Next radius message id: 0
  RADIUS client address: not configured

802.1X info for interface xe10
  portEnabled: true - portControl: Auto
  portStatus: Authorized - currentId: 10
  protocol version: 2
  reAuthenticate: disabled
  reAuthPeriod: 3600
  abort:F fail:F start:F timeout:F success:T
  PAE: state: Authorized - portMode: Auto
```

7.2.4. OpenSwitch OPX

Die Community-Version von OpenSwitch kann nur über die CPS-API oder mithilfe von Linux native APIs (Linux Kommandos) konfiguriert werden. Anzumerken ist, dass es abgesehen von FRR-Routing keine CLI zur Konfiguration im eigentlichen Sinne gibt. Der klare Fokus liegt aber auf der CPS-API, welche aber noch mitten in der Entwicklung steckt und deshalb noch nicht ausgereift ist. Da hier aber ein komplett anderer Ansatz zur Konfiguration gewählt wurde, ist vor allem für Netzwerktechniker, die sich auf der Command-Line wohl fühlen und diese gewohnt sind, der Umstieg schwer. OPX ist auf Programmierer ausgerichtet, die die CPS-API entweder über C++ oder Python ansprechen und mithilfe von YANG-Models Konfigurationen vornehmen. Um die Möglichkeiten der API zu testen, wurden zum Teil selbst programmierte Scripts und bereits vorhandene Beispielprogramme verwendet. Leider ist die Dokumentation der Netzwerkintegration und Konfiguration von OpenSwitch sehr spärlich und nicht am aktuellen Stand der Möglichkeiten. Da es sich hierbei um ein Open-Source-Projekt handelt, ist eine Instandhaltung der Dokumentation nicht immer sofort möglich. Die Konfiguration mit Linux-Kommandos ist für Linux-Admins geeignet und verwendet hauptsächlich die Linux-Befehle „ip link“ und „brctl“. Um einen Überblick über die Konfigurationsmöglichkeiten und das Handling von OpenSwitch OPX zu erhalten, wurden, sofern dies möglich war, für die Umsetzung der Testszenarien beide Konfigurationswege verwendet.

VLAN Bridge

Da es sich bei OPX um ein Linux-Betriebssystem für Switches, ohne viele Extras und Erweiterungen, handelt, müssen alle Konfigurationen wie auf einem Linux Server vorgenommen werden. Deshalb werden das Bridge-Utility (Brctl) und IP-Link für die Konfiguration verwendet. Auch hier muss für jedes VLAN eine eigene Bridge angelegt werden, wobei aber zwei VLANs nicht miteinander gebridged werden dürfen. Für die Konfiguration mithilfe der CPS-API wurde ein Beispielscript in der Programmiersprache Python erstellt, welches drei Funktionen bietet: create_vlan, add_port_to_vlan, add_ip_to_int. Es muss aber angemerkt werden, dass in der API-Referenz und in der Dokumentation keine Möglichkeit gefunden wurde, einen Switchport getaggt zu einer Bridge hinzuzufügen. Es ist lediglich möglich, ungetaggte Ports mithilfe des CPS-Objekts als Mitglied einer Bridge zu konfigurieren, weshalb für die Konfiguration eines Trunk-Ports trotzdem die Linux-Kommandos verwendet werden müssen.

```
Switch1# Konfiguration nur mithilfe von Linux-Kommandos
#VLAN-Bridges erstellen
brctl addbr br10
brctl addbr br11

#IP-Adressen zur VLAN-Bridge hinzufügen
ip addr add 10.0.10.1/24 dev br10
ip addr add 10.0.11.1/24 dev br11

#Trunk-Port konfigurieren (Getaggte Subinterfaces erstellen)
ip link add link e101-001-0 name e101-001-0.10 type vlan id 10
ip link add link e101-001-0 name e101-001-0.11 type vlan id 11

#Interfaces zur Bridge hinzufügen
brctl addif br10 e101-001-0.10
brctl addif br11 e101-001-0.11
brctl addif br10 e101-002-0
brctl addif br11 e101-003-0
```

```
Switch2# Konfiguration mithilfe der CPS-API und Linux-Kommandos
vi vlan_and_ip_script.py
```

```
import cps_object
import cps

def create_vlan(vlan_id):
    cps_obj = cps_object.CPSObject('dell-base-if-cmn/if/interfaces/interface')
    cps_obj.add_attr("base-if-vlan/if/interfaces/interface/id",vlan_id)
    cps_obj.add_attr('if/interfaces/interface/type','ianaift:l2vlan')

    cps_update = {'change':cps_obj.get(),'operation': 'create'}
    transaction = cps.transaction([cps_update])

    if not transaction:
        raise RuntimeError ("Error creating Vlan")
    print "Successfully created VLAN", vlan_id
    obj

def add_port_to_vlan(vlan_id, port_list):
    cps_obj = cps_object.CPSObject('dell-base-if-cmn/if/interfaces/interface')
    cps_obj.add_attr('if/interfaces/interface/name',vlan_id)
    cps_obj.add_attr('dell-if/if/interfaces/interface/untagged-ports',port_list)

    cps_update = {'change':cps_obj.get(),'operation': 'set'}
    transaction = cps.transaction([cps_update])

    if not   transaction:
        raise   RuntimeError   ("Error   in   adding   port   to   Vlan")
    print "Successfully added Port(s)", port_list, "to VLAN", vlan_id

def add_ip_to_int(ifindex, ip_addr, prefix_len):
    ip_attributes   =  {"base-ip/ipv4/ifindex":   ifindex,"ip":ip_addr,"prefix-length":prefix_len}
    cps_obj = cps_object.CPSObject('base-ip/ipv4/address',data=ip_attributes)
    cps_object.add_attr('base-ip/ipv4/address/ip',"ipv4")

    cps_update   =  ('change':cps_obj.get(), 'operation':'create')
    transaction   =  cps_object.transaction([cps_update])

    if not   transaction:
        raise   RuntimeError   ("Error   in   adding   IP   to   Interface")
    print "Successfully added IP", ip_addr, "/", prefix_len, "to Interface (index)", ifindex
```

```
if __name__ == "__main__":
    #VLANs 10 und 11 erstellen
    create_vlan(10)
    create_vlan(11)
    #IPs zur VLAN-Bridge hinzufügen (Interface-Index muss vorher bekannt sein!)
    add_ip_to_int(48, '10.0.10.2', 24)
    add_ip_to_int(49, '10.0.11.2', 24)
    #Ungetaggte Ports zur VLAN-Bridge hinzufügen
    add_port_to_vlan(10, ['e101-002-0'])
    add_port_to_vlan(10, ['e101-003-0'])


#Erstellen der Subinterfaces für getaggte VLANs am Trunk kann nur mit Linux-Kommandos erfolgen
ip link add link e101-001-0 name e101-001-0.10 type vlan id 10
ip link add link e101-001-0 name e101-001-0.11 type vlan id 11

#Subinterfaces zur Bridge hinzufügen
brctl addif br10 e101-001-0.10
brctl addif br11 e101-001-0.1
```

Absicherung der Switchports nach CCNA Security Chapter 6 Lab A

Die Absicherung der Switchports konnte zu großen Teilen nicht umgesetzt werden, da OPX einerseits nur für den Einsatz im Datacenter-Bereich konzipiert wurde, andererseits die Entwicklung noch nicht soweit fortgeschritten ist, dass alle Funktionen implementiert wurden. Ein Aktivieren des BDPU-Guard ist mit Linux-Standard-Tools, genauso wie Portfast oder Storm-Control, nicht möglich und wird deshalb von OPX nicht unterstützt. Ebenso bietet die CPS-API als auch Metaswitch mit seinen Protocol Stacks keine Möglichkeit zur Konfiguration dieser Technologien, weshalb es nicht möglich ist, den Vorschlägen des Best-Practice-Guides zu folgen und eine Absicherung der Switchports durchzuführen. Einzig das Ändern des native VLANs für den Trunk-Port funktioniert sowohl mit Linux-Kommandos als auch mit der CPS-API.

```
Switch1# Konfiguration mit Linux-Tools
brctl addbr br99
ip addr add 10.0.99.1/24 dev br99
brctl addif br99 e101-001-0


Switch2# Konfiguration mit der CPS-API (Beispielscript)
……
if __name__ == "__main__":
    create_vlan(99)
    add_port_to_vlan(99, ['e101-001-0'])
    add_ip_to_int(50, '10.0.99.2', 24)
……
```

802.1x-Authentifizierung mit lokalem Radius Server

Die Dokumentation von OPX bietet keine Informationen zur Konfiguration von 802.1x-Authentifizierung, weshalb dieses Testszenario nicht umgesetzt werden konnte. Es finden sich auch in der CPS-API und in den zur Verfügung gestellten Beispielen und Referenzen keine Ansätze dafür. Vermutlich liegt dies daran, dass OpenSwitch für den Einsatz im Datacenter gedacht ist, und hier normalerweise keine 802.1x-Authentifizierung notwendig ist. Möglicherweise könnte mithilfe von Linux-Tools wie hostapd und wpasupplicant eine Authentifizierung der Switchports ermöglicht werden, jedoch gibt es auch hierfür keine Referenzen oder Beispiele.

7.3. LAG/MCLAG

7.3.1. Cumulus Network Linux

LAG (Link Aggregation Group)

Cumulus nennt die Aggregation von mehreren physischen Switchports zu einer Verbindung „Bond“. Die Bezeichnung stammt von Linux, wo ebenfalls immer von „bonding“ gesprochen wird. Die Konfiguration eines Bonds, also einer LAG, funktioniert mit einem einfachen Kommando in der NCLU oder in der Datei „/etc/network/interfaces“. Zusätzlich kann noch der Betriebsmodus des Bonds ausgewählt werden. Dabei handelt es sich um LACP oder Balance-XOR.

```
#Bond anlegen und Modus wählen
net add bond bond0 bond slaves swp20-23
net add bond bond0 bond mode balance xor

#Bestätigen
net commit

net show interface bond0
UP  bond0   00:02:00:00:00:12  40G      1500   Bond

Bond Details
Bond Mode:       Balance-XOR

UP  swp20(P)  10G         0     0      0               0
UP  swp21(P)  10G         0     0      0               0
UP  swp22(P)  10G         0     0      0               0
UP  swp23(P)  10G         0     0      0               0
```

MCLAG (Multi Chassis LAG)

Laut Cumulus-Dokumentation handelt es sich bei diesem Betriebssystem um eine „echte“ Mulit-Chassis Link Aggregation (MLAG). Da Cumulus eine virtuelle Appliance anbietet, hätte hier eine komplette Konfiguration vorgenommen werden können. Die Konfiguration des MLAG war auf dem Dell-Switch zwar möglich, jedoch konnte der Peerlink in der virtuellen Umgebung nicht gestartet werden, da CumulusVX diese Funktion nicht unterstützt. Deshalb erfolgt die Beschreibung der Konfiguration ohne abschließende Überprüfung.

Die Konfiguration erfolgt komplett mit dem NCLU und ist mit ein paar Kommandos erledigt. Zuerst muss das Peering erstellt werden, wobei hier auf beiden Interfaces dieselben MAC-Adressen, aber unterschiedliche IP-Adressen verwendet werden. Die MAC-Adresse muss im Bereich zwischen 44:38:39:ff:00:00 und 44:38:39:ff:ff:ff liegen.
Aufgrund zu weniger RJ45-SFPs konnte für den Peer-Link nur ein einzelner Port verwendet werden. Zusätzlich muss auf beiden Switches am Access-Port dieselbe CLAG-ID verwendet werden. In der NCLU wird stets die Bezeichnung CLAG verwendet, während Cumulus immer von einem MLAG spricht. Dies ist darauf zurückzuführen, dass vor einigen Versionen die Bezeichnung geändert wurde. Zum Schluss wird noch das VLAN untagged am Switchport (bond-to-host) gesetzt.

```
#CLAG-MAC und CLAG-Peer konfigurieren
net add clag peer sys-mac 44:38:39:FF:F0:F0 interface swp1 primary backup-ip 10.0.0.2

#VLAN erstellen und Link zu Host in neue LAG hinzufügen
net add vlan 11
net add bond bond-to-host-11 bridge access 11

#CLAG für die Host-LAG setzen
net add clag port bond bond-to-host-11 interface swp4 clag-id 1
#Konfiuration speichern
net pending
net commit

switch2 (nicht vorhanden)#
```

```
#Selbe Konfiguration wie Switch 1, bis auf:
net add clag peer sys-mac 44:38:39:FF:F0:F0 interface swp1 secondary backup-ip 10.0.0.1

net pending
net commit
```

7.3.2. Pica8 PicOS

LAG (Link Aggregation Group)

Eine Link Aggregation kann in PicOS in der CLI konfiguriert werden. Um die Konfiguration durchzuführen, muss lediglich ein „aggregate-ethernet“-Interface erstellt werden. Diesem werden dann die einzelnen Interfaces zugeordnet. Es ist möglich, die LAG entweder im „Hash-Mapping“ oder im LACP-Modus zu betreiben. Dieser Modus wird beim Erstellen des Aggregate-Interfaces gewählt und kann nachträglich nicht mehr geändert werden.

```
#Aggregate-Interface erstellen und Modus wählen
set interface aggregate-ethernet ae0 aggregated-ether-options lacp enable true

#Interfaces zum Aggregate-Interface hinzufügen
set interface gigabit-ethernet ge-1/1/1 ether-options 802.3ad ae10
set interface gigabit-ethernet ge-1/1/2 ether-options 802.3ad ae10
set interface gigabit-ethernet ge-1/1/3 ether-options 802.3ad ae10
set interface gigabit-ethernet ge-1/1/4 ether-options 802.3ad ae0

run show interface aggregate-ethernet ae1
Physical interface: ae1, Enabled, Physical link is Down
Aggregated link protocol: LACP

ge-1/1/1 down(inactive) Auto
ge-1/1/2 down(inactive) Auto
ge-1/1/3 down(inactive) Auto
ge-1/1/4 down(inactive) Auto
```

MCLAG (Multi Chassis LAG)

Da keine Möglichkeit zur Virtualisierung von PicOS besteht, konnte die Konfiguration des MLAG ebenfalls nur mit einem Switch durchgeführt werden. Der gesamte Vorgang ist jedoch relativ aufwendig, wenn man bedenkt, wie viele Schritte man für diese einfache Konfiguration benötigt. Zuerst müssen alle beteiligten Links in LAGs zusammengefasst werden. Dabei ist es egal, ob es sich hierbei nur um eine einzelne Verbindung über ein Kabel handelt oder nicht. Eine LAG muss in jedem Fall angelegt werden, da für das MCLAG nur bereits aggregierte Verbindungen verwendet werden können. Zusätzlich muss dann noch ein VLAN für den Peer-Link und eine IP-Adresse festgelegt werden. Abschließend bleibt zu bemerken, dass die Konfiguration einer MCLAG relativ aufwendig ist, sofern sie manuell auf der Kommandozeile konfiguriert wird.

```
#Link Aggregation konfigurieren – ae1 = Link zu Host, ae2 = Peerlink
set interface aggregate-ethernet ae1 aggregated-ether-options lacp enable true
set interface aggregate-ethernet ae2 aggregated-ether-options lacp enable true

#MLAG auf einem aggregierten Link aktivieren
set interface aggregate-ethernet ae1 aggregated-ether-options mlag disable false

#MLAG Domain ID konfigurieren
set interface aggregate-ethernet ae1 aggregated-ether-options mlag domain-id 1

#Interfaces zu den aggregierten Links hinzufügen
set interface gigabit-ethernet te-1/1/1 ether-options 802.3ad ae1
set interface gigabit-ethernet te-1/1/3 ether-options 802.3ad ae2
set interface gigabit-ethernet te-1/1/4 ether-options 802.3ad ae2

#VLAN anlegen
set vlans vlan-id 10
set vlans vlan-id 4094 l3-interface vlan4094
set vlan-interface interface vlan4094 vif 4094 address 10.0.0.1 prefix-length 24
```

```
#VLANs zum LAG hinzufügen:
set interface aggregate-ethernet ae1 family ethernet-switching port-mode access
set interface aggregate-ethernet ae1 family ethernet-switching native-vlan-id 10
set interface aggregate-ethernet ae2 family ethernet-switching port-mode trunk
set interface aggregate-ethernet ae2 family ethernet-switching native-vlan-id 4094
set interface aggregate-ethernet ae2 family ethernet-switching vlan members 10

#MLAG-Peer Adresse konfigurieren
set interface aggregate-ethernet ae1 aggregated-ether-options mlag peer 10.0.0.1 peer-link ae2

commit

run show mlag
Domain-id  Local-LAG  System-id          Prio  Source         Peer           Peer-link  Hello-interval
1          ae1        14:18:77:01:9f:01  0     N/A            10.0.0.2       ae2        4

switch2 (nicht vorhanden)#
#Selbe Konfiguration wie Switch 1, bis auf:

#VLANs anlegen
set vlan-interface interface vlan4094 vif 4094 address 10.0.0.2 prefix-length 24

#MLAG-Peer Adresse konfigurieren
set interface aggregate-ethernet ae1 aggregated-ether-options mlag peer 10.0.0.2 peer-link ae2

commit
```

7.3.3. IP Infusion OcNOS

LAG (Link Aggregation Group)

Das Protokoll LACP ist die einzige Möglichkeit für die Link-Aggregation unter OcNOS. Die Konfiguration wird wie gewohnt in der CLI im „config“-Modus vorgenommen und ist sehr einfach. Alle Interfaces, welche Teil einer LAG werden sollen, müssen einzeln mit einem Befehl hinzugefügt werden. Es wird automatisch ein Interface mit der Kennung „po“ für Port-Channel und der ID angelegt. Anzumerken ist, dass OcNOS unterschiedliche Benennungen für ein und dieselbe LAG verwendet, was für Unklarheiten sorgen kann.

```
#Switchports einer neuen Channel-Group (LAG) hinzufügen
interface xe20
 channel-group 1 mode active
interface xe21
 channel-group 1 mode active
interface xe22
 channel-group 1 mode active
interface xe23
 channel-group 1 mode active

show etherchannel detail
% Aggregator po1 7
% Receive link count: 4 - Transmit link count: 4
% Link: xe20 (3) sync: 0
% Link: xe21 (4) sync: 0
% Link: xe22 (5) sync: 0
% Link: xe23 (6) sync: 0
```

MCLAG (Multi Chassis LAG)

Leider bietet ipinfusion ebenso keine Möglichkeit zur Virtualisierung von OcNOS an, weshalb die Konfiguration einer MCLAG nur theoretisch möglich war. Der Aufwand für die Konfiguration hält sich in Grenzen und ist bei weitem nicht so groß wie bei anderen Betriebssystemen. Wichtig ist, dass für die Konfiguration zwingend eine Channel-Group (LAG) zum Client erforderlich ist, auch wenn die LAG nur ein einzelnes Interface enthält. Auf diese Aggregation, deren Interface Bezeichnung dann aber Po (Port-Channel) lautet, wird die MCLAG dann gebunden. Die MCLAG wird bereits im Vorhinein konfiguriert und muss auf beiden Switches mit denselben Parametern konfiguriert werden. Der Peer-Link, hier „intra-domain link“ genannt, muss nicht Teil einer LAG sein. Anzumerken ist, dass eine Konfiguration einer MCLAG nur im RSTP-Modus möglich ist. Unter den anderen Spanning-Tree-Modi wird dies von OcNOS

nicht unterstützt. Diese Tatsache und die, dass die Bezeichnungen ständig wechseln, fällt negativ auf, während sich die Syntax wieder an Ciscos IOS orientiert.

```
switch1#
#Spanning-Tree-Protokoll auf RSTP ändern, damit ein MCLAG möglich ist
bridge 1 protocol rstp vlan-bridge

#MCLAG konfigurieren, diese Konfiguration ist auf beiden Switches ident
mcec domain configuration
 domain-address 1111.2222.3333
 domain-system-number 1
 intra-domain link xe49

#Interface zum Server einer Channel-Group zuordnen
interface xe9
 switchport
 bridge-group 1
 switchport mode access
 switchport access vlan 10
 channel-group 1 mode active

#MCLAG auf Channel-Group aktivieren
interface po1
 mlag 1

switch2 (nicht vorhanden)#
#Selbe Konfiguration wie Switch 1
```

7.3.4. OpenSwitch OPX

LAG (Link Aggregation Group)

Eine Link Aggregation, unter Linux und OPX „bond“ genannt, kann in OpenSwitch sowohl mithilfe von Linux-Kommandos (iplink), als auch mit der CPS-API konfiguriert werden. Als Protokoll für den Bond wird LACP verwendet, wobei zwischen verschiedenen Modus und Hashing-Verfahren gewählt werden kann. Für die Konfiguration mithilfe der CPS-API wurde das bereits vorhandene Beispiel-Script „cps_confg_lag.py“ verwendet, welches von den Entwicklern zur Verfügung gestellt wurde, um die CPS-API mit einfachen Funktionen zu erklären.

```
Switch1# (Konfiguration des Bond mit Linux-Tools)
#Bond erstellen und Modus wählen
ip link add bond1 type bond mode balance-rr miimon 50

#Interfaces zum Bond hinzufügen
ip link set e101-005-0 master bond1
ip link set e101-006-0 master bond1

#Bond starten
ip link set dev bond1 up

Switch2# (Konfiguration des Bond mit CPS-API - Beispielscript)
#Bond erstellen
cps_config_lag.py --create lname bond1

#Interfaces zum Bond hinzufügen (Getrennt durch Beistrich)
cps_config_lag.py --add lname bond1 --port e101-005-0,e101-006-0

#Bond starten
cps_config_lag.py --set lname bond1 –admn up
```

MCLAG (Multi Chassis LAG)

Leider konnte weder in der Dokumentation von OPX und der CPS-API, noch in der API-Referenz ein Hinweis auf eine Konfiguration eines MCLAGs gefunden werden, weshalb dieses Testszenario ebenfalls nicht umgesetzt werden konnte. Möglicherweise wird diese Funktion noch in einer zukünftigen Version von OPX nachgereicht und die Konfiguration ermöglicht. Eine ähnliche Lösung wie Cumulus‘ clagd könnte auch hier eingesetzt werden. Bis jedoch an dieser Stelle weitergearbeitet wird, muss OPX ohne die Möglichkeit eines MCLAGs auskommen.

7.4. Kompatibilität zu traditionellen Systemen (Cisco)

7.4.1. Cumulus Network Linux

VLAN Bridge, Spanning-Tree und shared VLANs

Ohne eine zusätzliche Konfiguration funktioniert die Kommunikation über den Trunk der beiden Switches nicht, da beide ein anderes Spanning-Tree-Protokoll sprechen. Der Cisco-Switch verwendet das veraltete PVSTP (Per VLAN Spanning-Tree-Protocol), während eine „VLAN-aware Bridge“ in Cumulus Linux nur den RSTP (Rapid-Spanning-Tree-Protocol) Modus unterstützt. Um nun eine Kommunikation zwischen den beiden Switches herzustellen, muss am Cisco-Switch das Protokoll auf MSTP (Multiple Spanning-Tree-Protocol) geändert werden. Das von Cumulus verwendete RSTP kann mit MSTP kommunizieren, wodurch die Verwendung des Trunks möglich wird.

```
Cisco-Switch#
#Spanning-Tree-Protokoll ändern
spanning-tree mode rstp
```

Layer3 OSPF-Routing

Unter Cumulus Linux ist das Aktivieren von OSPF relativ aufwendig, da zuerst FRRouting, ein IP-Routing Framework für Linux, konfiguriert werden muss. Dazu müssen in der Datei /etc/frr/deamons die Routing-Dienste „zebra“ und „ospfd“ aktiviert werden. Danach erfolgt ein Neustart von FRRouting und die Konfiguration von OSPF ist möglich. Hier gibt es nun wiederum mehrere Möglichkeiten zur Konfiguration, was für Verwirrung sorgen kann. Einerseits gibt es die von FRRouting zur Verfügung gestellte vtysh, welche sehr an die Cisco-CLI angelehnt ist und vor allem dazu dient, den Routing-Prozess anzulegen. Andererseits gibt es das NCLU, welches ebenfalls die Konfiguration ermöglicht. Die Punkt-zu-Punkt-Verbindung funktionierte ohne Probleme und die beiden Clients konnten sich untereinander erreichen.

```
#OSPFv2 Dienst aktivieren
vi /etc/frr/daemons
> ospfd=yes

#FRRouting Konfiguration neu einlesen
sudo systemctl start frr.service

#OSPF-Prozess erstellen und parametisieren
net add ospf router-id 1.1.1.2
net add ospf network 10.10.10.0/24 area 0.0.0.0
net add ospf redistribute connected
net add ospf passive-interface swp10

#Interfaceoptionen für OSPF festlegen (Verbindungstyp, Area, Security)
net add interface swp1 ospf area 0.0.0.0
net add interface swp1 ospf network point-to-point
net add interface swp1 ospf message-digest-key 1 md5 SUPERSECRETkey

net show ospf neighbor
Neighbor ID     Pri State           Dead Time Address         Interface            RXmtL RqstL DBsmL
1.1.1.1           1 Full/DROther      31.695s 10.10.10.2      swp1:10.10.10.1          0     0     0
route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.10.10.0      *               255.255.255.0   U     0      0        0 swp1
192.168.10.0    *               255.255.255.0   U     0      0        0 swp2
192.168.200.0   10.10.10.2      255.255.255.0   UG    20     0        0 swp1
```

Layer3 OSPF-Routing IPv6

Cumulus Linux ist mit IPv6 kompatibel und kann einfach konfiguriert werden. Im Grunde genügt es bei den meisten Kommandos, „ip“ durch „ipv6“ zu ersetzen. Um ein IPv6-fähiges Routing-Protokoll zu verwenden, muss dieses ebenfalls in der Konfigurationsdatei von FRRouting aktiviert werden. Dies hat zur Folge, dass die Konfiguration von FRRouting-Dienst neu eingelesen werden muss. Danach muss in der NCLU die Konfiguration des OSPFv3-Prozesses vorgenommen und auf das Interface gebunden werden. Laut Definition würde OSPFv3 sowohl IPv4 als auch IPv6 unterstützen, jedoch bietet die Implementierung von FRRouting nur eine Kompatibilität zu IPv6. Die beiden Geräte konnten miteinander kommunizieren, jedoch ohne Absicherung der Punkt-zu-Punkt-Verbindung, da es nicht möglich ist, einen Authentifizierungsschlüssel für die OSPF-Verbindung zu setzen.

```
#OSPFv3-Dienst aktivieren
vi /etc/frr/daemons
> ospfd=yes

#FRRouting-Konfiguration neu einlesen
sudo systemctl reload frr.service

#OSPFv3 Prozess erstellen und parametisieren
net add ospf6 router-id 1.1.1.1
net add ospf6 interface swp1 area 0.0.0.0
net add ospf6 redistribute connected

#Interface Verbindungstyp wählen
net add interface swp1 ospf6 network point-to-point

net show ospf6 neighbor
Neighbor ID     Pri           DeadTime        State/IfState         Duration       I/F[State]
1.1.1.1         1             00:00:35        Full/DR               00:03:02       swp1[BDR]

net show ospf6 route
O        2001:dead:666::/64    [110/100] is directly connected, swp1, 00:03:07
O>*      2001:dead:beef::/64   [110/100] via 2001:dead:666::2/64, swp1, 00:03:07
```

7.4.2. Pica8 PicOS

VLAN Bridge, Spanning-Tree und shared VLANs

Die Auswahl an Spanning-Tree-Protokollen ist unter PicOS größer als bei anderen NOS, weshalb es hier theoretisch keine Kompatibilitätsprobleme geben sollte. Verwendet man die Standardkonfiguration von PicOS, können der Cisco Switch und der Open Networking-Switch über den Trunk kommunizieren, da beide das PVSTP (Per Vlan Spanning Tree Protocol) verwenden. Da dieses Protokoll aber unter PortFast, von PicOS Edge genannt, nicht funktioniert, muss MSTP konfiguriert und betrieben werden. Für eine Kommunikation zwischen den beiden Switches ist es dann notwendig, dass der Cisco-Switch ebenfalls dieses Protokoll spricht. Hierfür muss am Cisco-Switch wieder das Protokoll RSTP konfiguriert werden, welches mit MSTP kompatibel ist. Leider gibt die Dokumentation von Pica8 keine Auskunft darüber, ob eine Konfiguration aller Sicherheitsfeatures auch mit RSTP möglich wäre.

```
Cisco-Switch#
#Spanning-Tree-Protokoll ändern
spanning-tree mode rstp

PicOS#
#Spanning-Tree-Protokoll ändern
set protocls spanning-tree force-version 3   -   0=STP, 2=RSTP, 3=MSTP, 4=Rapid PVST+
```

Layer3 OSPF-Routing

Auch dieses Beispiel konnte fehlerfrei umgesetzt werden, jedoch ist die Dokumentation nicht ausreichend beschrieben. Es benötigte mehrere Versuche, bis die Bedeutung und der Zweck der VLAN-Interfaces, die für die Konfiguration von OSPF benötigt werden, klar wurden. PicOS benötigt nämlich für

die Verbindung zwischen OSPF-Prozess und physischem Interface zwingend ein VLAN-Interface mit IP-Adresse. Warum diese Notwendigkeit besteht, konnte leider nicht in Erfahrung gebracht werden, vermutlich dient es internen Abläufen im System. Außerdem ist es zwingend erforderlich, zuerst die Konfiguration des VLANs durchzuführen und diese anschließend zu bestätigen, da die Konfiguration des OSPF-Prozesses davon abhängt. Da das VLAN am Switchport dann als native-vlan definiert werden muss, ist es auf der Gegenseite nicht notwendig, eine ähnliche Konfiguration vorzunehmen und bestätigt den Verdacht, dass es sich hier um systeminterne Abhängigkeiten handelt. Eine weitere Eigenheit ist das Weiterleiten von verbundenen Netzwerkrouten, unter Cisco bekannt als „redistribute connected“. Um denselben Effekt zu erzielen, muss zuerst eine Routing-Policy erstellt und anschließend dem OSPF-Prozess übergeben werden.

```
#VLAN und VLAN-Interface erstellen
set vlans vlan-id 1010
set interface gigabit-ethernet te-1/1/1 family ethernet-switching native-vlan-id 1010
set vlans vlan-id 1010 l3-interface vlan-1010
set vlan-interface interface vlan-1010 vif vlan-1010 address 10.10.10.1 prefix-length 24

#Diese Konfiguration muss vor dem OSPF-Prozess bestätigt werden
commit

#OSPF-Prozess konfigurieren
set protocols ospf4 router-id 1.1.1.1
set protocols ospf4 area 0.0.0.0 interface vlan-1010 vif vlan-1010 address 10.10.10.1
set protocols ospf4 area 0.0.0.0 interface vlan-1010 vif vlan-1010 address 10.10.10.1 authentication md5 1 password
SUPERSECRETkey

#”Connected” Subnetze in OSPF exportieren
set policy policy-statement ospf-connected term ospf-connected  from protocol connected
set protocols opsf4 export ospf-connected

#Bestätigen
commit
```

Nach einigen Versuchen hat auch hier die Punkt-zu-Punkt-Verbindung zwischen den beiden Switches funktioniert und die Clients konnten sich untereinander erreichen.

```
run show ospf4 neighbor
10.10.10.2        vlan-1010/vlan-1010    Full      1.1.1.2          1      32

run show ospf4 database
   OSPF link state database, Area 0.0.0.0
Router   *1.1.1.1          1.1.1.1          0x80000003  10    0x2  0x2feb  36
Network  *10.10.10.1       1.1.1.1          0x80000001  780   0x2  0xb581  32
Network  *192.168.200.0    1.1.1.1          0x80000001  790   0x2  0xba21  32
Router    1.1.1.2          1.1.1.2          0x80000003  791   0x22 0x9042  48
ASExt-2  *192.168.10.0     1.1.1.2          0x80000001  10    0x2  0xd3e6  36
ASExt-2  *10.10.10.0       1.1.1.2          0x80000001  10    0x2  0xc0e2  36
```

Layer3 OSPF-Routing IPv6

Die Konfiguration von OSPFv3 ist nahezu ident mit jener von OSPFv2, weil lediglich das Prefix „ospf6“ verwendet werden muss. Es gibt trotzdem einige Einschränkungen, wie zum Beispiel die Tatsache, dass keine Absicherung der Verbindung zwischen den beiden Routern möglich ist. Ansonsten bietet PicOS eine uneingeschränkte Unterstützung von IPv6 und ist somit für den Dual-Stack-Betrieb geeignet.

```
run show ospf6 database
#VLAN und VLAN-Interface erstellen
set vlans vlan-id 1010
set interface gigabit-ethernet te-1/1/1 family ethernet-switching native-vlan-id 1010
set vlans vlan-id 1010 l3-interface vlan-1010
set vlan-interface interface vlan-1010 vif vlan-1010 address 2001:dead:666::1 prefix-length 64

#Diese Konfiguration muss vor dem OSPF-Prozess bestätigt werden
commit

#OSPF-Prozess konfigurieren
set protocols ospf6 router-id 1.1.1.1
set protocols ospf6 area 0.0.0.0 interface vlan-1010 vif vlan-1010 address 2001:dead:666:1 disable false
```

```
#”Connected” Subnetze in OSPF exportieren
set policy policy-statement ospf6-connected term ospf6-connected  from protocol connected
set protocols opsf6 export ospf6-connected
commit
run show ospf6 neighbor
2001:dead:beef::2        vlan-1010/vlan-1010    Full      1.1.1.2          1      37

run show ospf6 database
   OSPF link state database, Area 0.0.0.0
Router   *1.1.1.1              1.1.1.1
Network  *2001:dead:666::      1.1.1.1
Network  *2001:dead:beef::     1.1.1.1
Router    1.1.1.2              1.1.1.2
ASExt-2  *2001:dead:bee2::     1.1.1.2
ASExt-2  *2001:dead:666::      1.1.1.2
```

7.4.3. IP Infusion OcNOS

VLAN Bridge, Spanning-Tree und shared VLANs

Um eine Kommunikation zwischen den beiden Switches zu ermöglichen, ist es auch bei OcNOS notwendig, das Spanning-Tree-Protokoll zu ändern. Es gibt zwar eine Unterstützung von STP und RSTP, jedoch verwendet der Cisco 2960 Switch das PVSTP-Protokoll, welches von OcNOS nicht unterstützt wird. Es ist zwar möglich, am Cisco-Switch RSTP zu konfigurieren, jedoch war es nicht möglich, eine Verbindung zwischen den beiden Switches zustande zu bringen. Deshalb musste hier ebenfalls ein Kompromiss und somit das MSTP gewählt werden, damit beide Switches miteinander kommunizieren können. Dies bringt unter OcNOS jedoch Nachteile mit sich, da, wie bereits oben festgestellt, dadurch keine MCLAGs mehr in dieser Bridge möglich sind. Um Abhilfe zu schaffen, könnte man mit mehreren Bridges arbeiten und diese mit einer L3-Route verbinden. Nach der Konfiguration von MSTP am Dell-Switch und RSTP am Cisco-Switch war es möglich, Pakete über den Trunk zu senden.

```
Cisco-Switch#
#Spanning-Tree-Protokoll ändern
spanning-tree mode rstp

OcNOS#
#Spanning-Tree-Protokoll ändern
Bridge 1 protocol mstp
```

Layer3 OSPF-Routing

Die Konfiguration von OSPF unter OcNOS ist problemlos möglich und funktioniert genauso wie unter Ciscos IOS. Bei diesem Beispiel war sogar jedes Kommando ident, was Vorteile mit sich bringen kann. Zuerst muss der OSPF-Prozess erstellt und parametrisiert werden, anschließend wird am Interface die zusätzliche Konfiguration vorgenommen. Die Verbindung kam sofort zustande, Uneindeutigkeiten während der Konfiguration konnten keine gefunden werden.

```
#OSPF-Prozess erstellen
router ospf 1
 network 10.10.1.0/24 area 0.0.0.0
 redistribute connected
 router-id 1.1.1.1

#OSPF-Prozess an Switchport binden und Konfiguation vornehmen
int xe2
 ip address 10.10.1.1/24
 ip ospf message-digest-key 1 md5 SUPERSECRETkey
 ip ospf authentication message-digest
 ip ospf network point-to-point

show ip route
IP Route Table for VRF "default"
C       192.168.10.0/24  is directly connected, vlan1.10
O       192.168.200.0/24 via 1.1.1.2, xe2, Area 0.0.0.0
C       10.10.10.0/24 is directly connected, xe2
C       127.0.0.0/8  is directly connected, lo
```

```
show ip ospf neighbor
1.1.1.2           1   Full/ -          00:00:34    10.10.1.2       xe2                    0
```

Layer3 OSPF-Routing IPv6

OcNOS bietet eine komplette Integration des IPv6-Stacks in das Betriebssystem. Die Konfiguration von IPv6-Befehlen funktioniert mit dem „ipv6“-Prefix. Die Parameter für das Routing-Protokoll OSPFv3 werden wie OSPFv2 in der CLI gesetzt. Nach der Konfiguration wurden Ping-Tests durchgeführt, welche erfolgreich abgeschlossen werden konnten.

```
#OSPF-Prozess erstellen
router ipv6 ospf 1
 redistribute connected
 router-id 1.1.1.1

#OSPF-Prozess an Switchport binden und Konfiguation vornehmen
int xe2
 ipv6 address 2001:dead:666::1/65
 ipv6 router ospf area 0.0.0.0 instance-id 1
 ipv6 ospf network point-to-point
 no shutdown

show ipv6 route
IP Route Table for VRF "default"
C       2001:dead:beef::/64  is directly connected, vlan1.10
O       2001:dead:bee2::/64 via 1.1.1.2, xe2, Area 0.0.0.0
C       2001:dead:666::/64 is directly connected, xe2
C       ::1/128  is directly connected, lo

show ipv6 ospf neighbor
1.1.1.2           1   Full/DR -          00:02:11    xe2   0
```

7.4.4. OpenSwitch OPX

VLAN Bridge, Spanning-Tree und shared VLANs

Um die Kommunikation zwischen dem Cisco-Switch und OPX zu ermöglichen, muss grundsätzlich das STP aktiviert werden. OPX unterstützt aufgrund der verwendeten Linux-Tools keine anderen Spanning-Tree Protokolle als PerVLAN STP. Dieses Protokoll muss auf jeder VLAN-Bridge einzeln aktiviert werden, was mithilfe des Bridge-Utilities passieren muss, weil die CPS-API das Aktivieren von STP noch nicht unterstützt. Da der Cisco-Switch jedoch ebenfalls das PvSTP unterstützt, ist eine Kommunikation über den Trunk-Port der Switches, ohne weitere Konfiguration, möglich.

```
Switch1# (OPX)
#Spanning-Tree-Protokoll auf allen VLAN-Bridges aktivieren
brctl stp br10 on
brctl stp br11 on
brctl stp br99 on
```

Layer3 OSPF-Routing

Da es sich bei OPX um ein reines Linux-System handelt und es nativ keine Möglichkeit für die Konfiguration eines dynamischen Routingprotokolls gibt, muss für die Konfiguration von OSPF auf ein third-party Tool zurückgegriffen werden. Standardmäßig verwendet OPX für den L3-Stack FRRouting als Teil seines Ecosystems. Durch FRRouting ist es möglich, verschiedene Routing-Protokolle am Switch zu aktivieren und somit eine Layer3-Konnektivität zwischen dem Router und dem L3-Switch zu ermöglichen. Dafür muss zuerst der zebra und ospfd-Daemon in der Konfigurationsdatei von FRRouting aktiviert werden und der Server anschließend neu gestartet werden. Danach ist die Konfiguration des OSPF-Prozesses in der vtysh, der CLI von FRRouting, möglich. Diese CLI ist sehr an jene von Cisco-Routern angelehnt, kann jedoch nicht mit der CPS-API konfiguriert werden. Dadurch ergibt sich eine dritte Schnittstelle zur Konfiguration von OPX und erschwert es, den Überblick über die verschiedenen

Möglichkeiten zu behalten. Der Test der OSPF-Route konnte erfolgreich abgeschlossen werden und die Test-PCs hatten untereinander eine Konnektivität.

```
Switch1#
vi /etc/frr/dameons
zebra=yes
ospfd=yes

service frr restart

vtysh
OPX#

#OSPF-Prozess erstellen
router ospf 1
 network 10.10.1.0/24 area 0.0.0.0
 redistribute connected
 router-id 1.1.1.1

#OSPF-Prozess an Switchport binden und Konfiguation vornehmen
int e101-001-0
 ip address 10.10.1.1/24
 ip ospf message-digest-key 1 md5 SUPERSECRETkey
 ip ospf authentication message-digest
 ip ospf network point-to-point
 no shutdown

 show ip route
 O       192.168.200.0/24 via 1.1.1.2, e101-001-0, Area 0.0.0.0
 C>*     10.10.10.0/24 is directly connected, e101-001-0
 C>*     10.0.0.0/24 is directly connected, eth0

 show ip ospf neighbor
 1.1.1.2           1   Full/ -          00:03:22    10.10.1.2       e101-001-0
```

Layer3 OSPF-Routing IPv6

Die Implementierung des IPv6-Stacks in OPX ist durch den Linux-Kernel bereits komplett erledigt und gegeben. FRRouting bietet ebenfalls einen eigenen Dienst für die IPv6-Version von OSPF, OSPFv3 (ospf6d). Die Konfiguration funktioniert fast identisch wie die der IPv4-Version, jedoch muss darauf geachtet werden, dass einige Parameter nicht mehr gesetzt werden können. Auch hier erfolgt die Konfiguration mit der vtysh von FRRouting, während keine Möglichkeit zur Parametrisierung mithilfe der CPS-API existiert. So ist es nicht möglich, eine Authentifizierung für die Punkt-zu-Punkt-Verbindung vorzunehmen. Die Konnektivität zwischen den Test-PCs war gegeben und die Verbindung hat einwandfrei funktioniert.

```
vi /etc/frr/dameons
zebra=yes
ospf6d=yes

service frr restart

vtysh

#OSPF-Prozess erstellen
router ipv6 ospf 1
 redistribute connected
 router-id 1.1.1.1

#OSPF-Prozess an Switchport binden und Konfiguation vornehmen
int e101-001-0
 ipv6 address 2001:dead:666::1/65
 ipv6 router ospf area 0.0.0.0 instance-id 1
 ipv6 ospf network point-to-point
 no shutdown

show ipv6 route
IP Route Table for VRF "default"
O>*       2001:dead:bee2::/64 via 1.1.1.2, e101-001-0, Area 0.0.0.0
C>*       2001:dead:666::/64 is directly connected, e101-001-0
C>*       fe80::/64 is directly connected, eth0

show ipv6 ospf neighbor
1.1.1.2           1   Full/DR -          00:05:12    e101-001-0   0
```

8. Beantwortung der Forschungsfragen

Dieses Kapitel dient zur abschließenden Beantwortung der Forschungsfragen anhand der zuvor vorgenommenen praktischen Tests, der Vorstellung der Betriebssysteme und Projekte sowie der allgemeinen Grundlagen zum Thema. Dadurch soll klar werden, ob Open Networking für den Betrieb in Produktionssystemen geeignet ist und welche Punkte für oder gegen den Einsatz dieser Technologie sprechen. Die Antworten dieser Fragen dienen als Grundlage für den Ausblick und das Fazit dieser Forschungsarbeit.

8.1. Welche Vor- und Nachteile bietet Open Networking und wo gibt es noch Aufholbedarf?

Mit der Entkoppelung von Hardware und Software wird ein Konzept, welches sich schon vor Jahren im Server-Bereich etabliert hat, nun auch in die Netzwerkinfrastruktur und Netzwerk-Hardware portiert. Dadurch ergeben sich in erster Linie dieselben Vorteile wie im Server-Bereich, also die freie Wahl der Hardware und Software und das Vermeiden einer Abhängigkeit von einem einzelnen Hersteller. Der Kostenfaktor spielt bei dieser Architektur ebenfalls eine große Rolle, da Bare-Metal und Whitebox Switches im Vergleich zu Blackbox Switches wesentlich günstiger in der Anschaffung als auch in der Wartung sind. Durch die Trennung der Hard- und Software ist es dem Kunden möglich, für den passenden Einsatzzweck den passenden Switch und eine beliebige Software zu wählen und somit die Flexibilität und die Kosteneffizienz zu steigern. Durch die Einführung von Open Networking hat sich eine große Community um viele Open-Source-Projekte gebildet, welche die Entwicklung dieser Technologie stark vorantreibt. Die starke Unterstützung der Linux Foundation und des Open Compute Projects beziehungsweise der Open Networking Foundation ist ebenfalls positiv anzumerken. Während ein Netzwerk bestehend aus Blackbox-Komponenten verschiedener Hersteller in vielen Fällen zu Kompatibilitätsproblemen und einem stark erhöhten Administrations- und Management-Aufwand führt, ist es das Ziel von einigen Open Networking Projekten, einheitliche Schnittstellen und Software-Agents, selbst für Blackbox Switches, zu erstellen. Die besten Beispiele dafür sind SONiC beziehungsweise FBOSS, welche als Software-Agents auf beliebiger Hardware und unterstützten Betriebssystemen installiert werden können, um das Management zu vereinfachen. Aber auch die vorgestellten Betriebssysteme verfolgen denselben Ansatz, empfehlen aber, dass auf allen Switches dasselbe Betriebssystem installiert wird, während das bei FBOSS und SONiC nicht unbedingt der Fall ist, da es sich um Software-Stacks und Agents handelt. Durch die Verwendung von Linux als Basisbetriebssystem entstehen große Vorteile für das Netzwerkmanagement und die Verwendung von DevOps-Tools. Die beliebige Erweiterbarkeit des Ecosystems eines Betriebssystems mit Drittanbieter-Anwendungen ist ebenfalls ein großer Vorteil gegenüber traditioneller Switch-Software mit geschlossenen Ecosystemen. Diese Betriebssysteme können entweder gar nicht oder nur mit zusätzlichen Lizenzgebühren um Funktionen erweitert werden.

Obwohl die Architektur nicht neu ist und bereits seit einigen Jahren Lösungen, wie jene von Pica8, existieren, sind Betriebssysteme für Open Networking Switches noch nicht ausgereift. Es gibt bei jedem der getesteten Betriebssysteme noch Aufholbedarf, vor allem bei der Vereinheitlichung der Konfigurationsmöglichkeiten. Die vielen verschiedenen Schnittstellen unterstützen oft nicht alle Parameter, wodurch es notwendig wird, die Konfiguration komplexerer Szenarien und Architekturen mit mehreren Schnittstellen vorzunehmen. Ein gutes Beispiel dafür ist Cumulus Linux, wo für die Absicherung der Switchports zwar fast die komplette Konfiguration in der NCLU durchgeführt werden kann, jedoch Storm-Control in der Konfigurationsdatei von switchd konfiguriert werden muss. Dass alle Parameter auch in der Linux-Shell und in Konfigurationsdateien vorgenommen werden können, ist zwar ein oft beworbenes Feature und einer der Hauptbeweggründe für die Verwendung von Open Networking Software, jedoch ist der Umstieg für Netzwerktechniker, die traditionelle Systeme wie jene von Cisco und Juniper gewohnt sind, schwer. Für diese Nutzergruppe gibt es aber Betriebssysteme wie jene von Dell und IP Infusion, welche sich durch die perfekte Integration der CLI auszeichnen. Laut Cumulus Networks ist einer der größten Stolpersteine bei der Integration von Open

Networking Systemen die Akzeptanz der Administratoren für die Integration einer neuen Technologie. [58] In einem sich so schnell entwickelndem Umfeld wie der IT ist es jedoch immer wichtig, am aktuellen Stand der Dinge zu bleiben und offen mit neuen Technologien umzugehen, um dann kurz- oder mittelfristig davon zu profitieren. Durch die vielen verschiedenen Open-Source-Projekte und Projektgruppen gibt es verschiedene Lösungen und Ansätze für Schnittstellen zur Hardware, was für unterschiedliche Kompatibilitätslisten der Betriebssysteme sorgt. Es gibt zwar Open-Source-Betriebssysteme, welche aber nur als Basis für eigene Weiterentwicklungen dienen sollen. Während mit dem ONIE-Bootloader eine einheitliche Möglichkeit zur Installation von NOS geschafften wurde, gibt es mit ONL und OPX zwei Open-Source-Ansätze für die Basis eines NOS. Diese setzen auf unterschiedliche Schnittstellen und Architekturen, was nicht unbedingt als Nachteil zu betrachten ist, da dadurch mehr Möglichkeiten für Betriebssystemhersteller entstehen und auch Betreiber von großen Rechenzentren eigens programmierte Software, basierend auf einer der beiden Grundgerüste, verwenden können.

Im Großen und Ganzen überwiegen aber die Vorteile von Open Networking, welche aber erst dann ausgenutzt werden können, wenn auch die Administratoren den Willen zur Umsetzung eines Open Networking Projekts haben. Viele Betriebssysteme sind zwar noch nicht ausgereift und auch viele, von den verschiedenen Institutionen geleiteten Projekte, sind erst in Entwicklung, aber die Hardware für Open Networking Switches ist bereits seit Jahren durch die Verwendung von „off-the-shelf“ ASICs und offenen Plattformen ausgereift. Auch bei der Hardware sind aber noch Verbesserungen, wie die weitere Forcierung und Erweiterung beziehungsweise Adaptierung der Open Switches, mit offener Architektur und Plattform, möglich.

8.2. Welche Auswirkungen hat Open Networking auf Netzwerkmanagement?

Durch die Verwendung eines, in den meisten Fällen, offenen Linux-Betriebssystems als Grundlage des NOS wird es möglich, DevOps-Tools wie Ansible, Chef oder Puppet ohne Zusatzmodule und Erweiterungen für das Netzwerkmanagement zu verwenden. Da die Konfiguration sowohl über die Linux-Shell als auch über eine etwaig vorhandene CLI vorgenommen werden kann, sind auch Linux-Administratoren ohne Umschulungen in der Lage, die Netzwerkkomponenten zu betreiben und zu warten. Das Netzwerkmanagement und die Orchestrierung vieler verschiedener Komponenten fällt bei proprietären Systemen oft schwer. Grund dafür sind die unterschiedlichen Management-Lösungen für verschiedene Hardware-Generationen und Produktfamilien traditioneller Netzwerklösungen. Oft sind die Produkte einer neuen Revision nicht mit den bestehenden Netzwerkmanagement-Tools kompatibel, wodurch es notwendig ist, mehrere Programme oder Eigenentwicklungen zur Inventarisierung und zum Konfigurationsmangement zu betreiben. Aber auch beim Einsatz von Netzwerkkomponenten verschiedener Hersteller kommt es oft zu Problemen beim Netzwerkmanagement, da die proprietären Orchestrierungstools nicht mit Komponenten anderer Firmen kompatibel sind. Hier ist ebenfalls der Einsatz mehrerer Programme für denselben Zweck, oder die Verwendung einer teuren Drittanbieterlösung mit Multi-Vendor-Support notwendig, welche dann oft nicht mit den aktuellsten Hardware-Generationen kompatibel ist.

Auch die Konfiguration mithilfe von REST-APIs, YANG-Models und NETCONF wird von fast allen Open Networking Betriebssystemen unterstützt. Durch dieses einheitliche System ist die Wartung der Komponenten und das Management der Infrastruktur, zusammen mit DevOps-Tools, einheitlich und mithilfe von Scripts möglich. Um diese Technologien zu verwenden, ist jedoch auch die Initiative der Administratoren gefragt, da am Anfang ein erhöhter Aufwand für das Netzwerkmanagement besteht. Durch diese Scripting-Möglichkeiten sind jedoch eher Linux-Administratoren als herkömmliche Netzwerktechniker für die Automatisierung und den Betrieb des Netzwerkmanagements gefragt. [37] Dell zeigt mit OpenSwitch OPX, mit der Verwendung der CPS-API als zentrale Schnittstelle zwischen allen Komponenten der Architektur, welche Möglichkeiten APIs bieten und in welche Richtung sich die Konfiguration von Netzwerkkomponenten entwickeln wird. Die Entwicklung von YANG-Models ist aufgrund der Ähnlichkeit der Syntax zu C eher für Programmierer geeignet. Durch diese Entwicklungen des Netzwerkmanagements wird es in Zukunft

notwendig sein, im Netzwerkbereich ebenfalls Full-Stack-Developer, also Administratoren, die Fähigkeiten aller Bereiche vereinen, einzustellen. Herkömmliche Netzwerktechniker werden aber weiterhin für die Planung und Wartung der Infrastruktur notwendig sein, während das Netzwerkmanagement in die Verantwortlichkeit von Full-Stack-Developern und Linux-Administratoren wandern wird.

Diese Veränderungen sind jedoch auch in traditionellen Netzwerken spürbar, da die Netzwerkautomatisierung auch hier immer schon Probleme bereitet hat. Viele Hersteller von Blackbox Switches haben diese Veränderungen und Auswirkungen bereits erkannt und ihre Systeme für solche Zwecke teilweise geöffnet. So hat Cisco mit dem DevNet eine große Community entwickelt, welche sich mit der Netzwerkautomatisierung von NX-OS, IOS-XR sowie der SDN-Lösung ACI beschäftigt. Hier wird die Entwicklung von Scripts, die Verwendung von NETCONF und die Modellierung von YANG-Models für die teilweise geöffneten Systeme gelehrt. Auch die Automatisierung mit DevOps-Tools ist mit Komponenten traditioneller Systeme möglich, jedoch sind dafür eigene Module und Erweiterungen verantwortlich, welche mit der CLI dieser Komponenten kommunizieren. Dell hat als Hersteller von Netzwerkkomponenten ebenfalls diese Auswirkungen erkannt und sich bereits früh für die Öffnung der Betriebssysteme entschieden. Durch die Veröffentlichung des Source-Codes von OPX ermöglicht Dell die Weiterentwicklung der CPS-API von der Community und profitiert selbst davon.

Durch die Einführung von Open Networking und offenen Betriebssystemen wird es in Zukunft möglich sein, Betriebssysteme verschiedener Hersteller, auch Blackbox-Komponenten, mit einheitlichen Methoden und DevOps-Tools zu orchestrieren. Dies ist auch auf die teilweise Öffnung der Betriebssysteme von Herstellern proprietärer Systeme zurückzuführen, welche den Mehrwert für Administratoren erkannt haben. Lösungen wie SONiC oder FBOSS sind der nächste Schritt in diese Richtung, da es dann möglich wird, am Betriebssystem Agents zu installieren. Diese kommunizieren direkt mit der Switching-ASIC und ermöglichen eine einheitliche Steuerung verschiedenster Komponenten durch vorher definierte Funktionen und Parameter. Auch Cisco hat Microsoft die Unterstützung von SONiC auf seinem Betriebssystem NXOS zugesprochen und ermöglicht damit eine Implementierung des Switch Abstraction interfaces (SAI) auf der aktuellen Nexus-Switch-Generation, wodurch möglicherweise auch von Cisco weitere Schritte in Richtung Open Networking getätigt werden. [53] Facebook und Microsoft zeigen mit ihren Open-Source-Projekten, wie das Netzwerkmanagement in Zukunft aussehen könnte. Dadurch kann erkannt werden, welche Vorteile diese Lösungen mit sich bringen und welche Auswirkungen Open Networking auf das Netzwerkmanagement hat.

8.3. Wie gut ist die Kompatibilität und Konnektivität zwischen traditionellen NOS und Open Networking NOS?

Wenn ein Unternehmen die Entscheidung trifft, die bestehende Netzwerkinfrastruktur mit Open Networking zu ersetzen, muss davon ausgegangen werden, dass zuerst nur einzelne Bereiche oder Teile des Netzwerks umgebaut werden. Ein Produktionssystem kann erfahrungsgemäß nicht an einem Tag umgestellt werden, da man außerdem vor und während der Umstellung Testsysteme benötigt. Um diese Testsysteme bestehend aus Open Networking-Komponenten einzubinden, muss eine Konnektivität und Kompatibilität zwischen proprietären Systemen und den offenen Systemen hergestellt werden. Die praktischen Tests im vorherigen Artikel haben gezeigt, dass es oft noch Schwierigkeiten bei der Layer2-Anbindung geben kann, da manche Betriebssysteme nicht alle Modi des Spanning-Tree unterstützen. Dadurch kann der Spanning-Tree nicht aufgebaut werden und der Datenaustausch funktioniert nicht. Die Layer3-Anbindung zwischen solchen Systemen mit gerouteten Verbindungen ist empfehlenswerter, da hier das offene Protokoll OSPF verwendet werden kann, welches von fast allen erhältlichen Netzwerbetriebsystemen, egal ob traditionelle oder offene NOS, unterstützt wird. Die Testfälle mit OSPF haben bei allen Betriebssystemen funktioniert und erlaubten eine problemlose Anbindung der offenen Infrastruktur an ein proprietäres System. Die Anbindung mit der IPv6-Version von OSPF, OSPFv3, wird ebenfalls unterstützt und hat einwandfrei funktioniert, auch wenn IPv6 in vielen Rechenzentren noch nicht zum Einsatz kommt.

Die Layerr2-Konnektivität konnte nur mit einem Campus-Switch von Cisco, ausgestattet mit IOS, getestet werden. Wie die Konnektivität zwischen Datacenter-Switches, wie zum Beispiel NXOS von Cisco, aussehen würde, lässt sich aber daraus ableiten. Solange offene Routingprotokolle verwendet werden, die auch vom Layer3-Stack des Ecosystems des verwendeten NOS unterstützt werden, ist eine Layer3-Konnektivität gegeben. Eine Anbindung mithilfe von Layer2 ist im Datacenter nicht mehr state-of-the-art und sollte deshalb vermieden werden. Falls man nicht auf ein dynamisches Routing-Protokoll zurückgreifen möchte, ist es auch möglich, statische Routen zu setzen und so eine Layer3-Verbindung zwischen den beiden Infrastrukturen zu erreichen. Die Anbindung mithilfe von Bonds, also der Aggregation von mehreren Switchports, konnte nicht getestet werden. Für eine ausfallsichere Anbindung muss ein ausfallsicheres System auf beiden Seiten vorliegen, wodurch dann zusätzliche Möglichkeiten für die Anbindung, wie LACP oder HSRP, bestehen. Diese Technologien konnten bei den Kompatibilitätstests leider nicht getestet werden. Da es sich aber um standardisierte Protokolle handelt, müsste eine Konnektivität herstellbar sein, vorausgesetzt, die verwendeten NOS bieten eine Unterstützung für diese Protokolle.

Falls man anstrebt, ein gemischtes System von Switches im Campus-Bereich zu betreiben, muss dazu geraten werden, keine proprietären Protokolle wie das PerVLAN-Spanning-Tree-Protokoll von Cisco einzusetzen, sondern aktuelle Spanning-Tree-Protokolle wie RSTP oder RSTP+ zu verwenden. Wenn die Betriebssysteme mithilfe von RSTP kommunizieren können, können die Switches auch im Campus eingesetzt werden. Es gilt aber zu beachten, dass der Fokus von Open Networking eindeutig im Datacenter liegt und viele Features wie Dot1X-Authentifizierung von einigen NOS nicht unterstützt werden. Im Datacenter ist ein gemischter Betrieb von proprietären Systemen und Open Networking nicht empfehlenswert, da dadurch ein zusätzlicher Netzwerkmanagement-Aufwand entstehen würde. Solange standardisierte Schnittstellen und Protokolle eingesetzt werden, ist die Kompatibilität und Konnektivität auch hier gegeben. Das Ziel sollte aber sein, ganze Infrastrukturen oder Projekte auf der Basis von Open Networking zu betreiben und damit alle Vorteile dieser Technologie zu nutzen. Damit kann etwaigen Komplikationen und Problemen, auch bei der Verwendung von proprietären Protokollen, in einem Produktionssystem entgegengewirkt und ein sicherer sowie hochverfügbarer Betrieb gewährleistet werden.

8.4. Kann Open Networking geschlossene Systeme ablösen und ist ein Einsatz in Produktionssystemen möglich?

Der Einsatz von Open Networking in Produktivsystemen ist möglich und wird auch von den Herstellern der Betriebssysteme und der Hardware beworben. Am stärksten vertreten sind Netzwerkinfrastrukturen auf Basis von Open Networking bei großen Unternehmen und Service Providern. Microsoft, Google und Facebook setzen allesamt selbst programmierte Netzwerkbetriebssysteme oder Agents auf ihren Netzwerkkomponenten ein. Facebook geht hier sogar noch ein paar Schritte weiter und lässt eigene Switches auf Basis einer offenen Architektur für die eigenen Rechenzentren herstellen, welche auf die eigene Open Networking Software FBOSS angepasst wurden. [12] Wenn ein Unternehmen eine gewisse Größe erreicht hat, zahlt es sich oft aus, Eigenkreationen zu erstellen und die Optimierung der Netzwerkinfrastruktur selbst in die Hand zu nehmen. Die getesteten Betriebssysteme richten sich aber nicht an Industriegiganten und riesige Service-Provider, welche in der Lage sind, selbst die Optimierung der Netzwerke durchzuführen. Cumulus Networks listet als eigene Kunden unter anderem Adobe, die NASA und NTT auf. Von Adobe wurde die Entwicklung eines Next-Gen-Datacenters auf Basis von Bare-Metal Switches und Cumulus Network Linux in Auftrag gegeben [59], während NTT Cumulus Linux für den Betrieb von Kubernetes und Docker auf der Switch-Software für die Kostenoptimierung und Steigerung der Flexibilität gewählt hat [60]. IP Infusions OcNOS wird als Betriebssystem für Bare-Metal-Switches von Edgcore am neuen London-Internet-Exchange-Knoten (LINX – LON2) verwendet. Während diese Arbeit verfasst wurde, ist bekannt geworden, dass der Internetknoten Londin-Internet-Exchange (LINX/LON2) Bare-Metal-Switches von Edgecore und OcNOS von IP Infusion verwenden wird, um einen optimalen

Betrieb und eine vollständige Automatisierung der Switches mit 100-Gigabit-Ethernet-Ports zu gewährleisten. [61]

Während diese Technologie in Amerika und Asien bereits in den Produktivsystemen einiger namhafter Unternehmen im Einsatz ist, hat sich diese Technologie in Österreich noch nicht durchgesetzt. Die Netzwerkautomatisierung ist auch hierzulande ein großes Thema und die Vorteile sind offensichtlich, jedoch gibt es nur vereinzelt Unternehmen, die Open Networking einsetzen. Einer der größten Betreiber von Open Networking in Österreich ist Hutchinson Drei, wo Dell-Switches der Type Z9100 mit der BigSwitch Monitoring Fabric und BigSwitch Light OS zum Einsatz kommen. In Österreich setzen die Unternehmen, laut Informationen von DellEMC, hauptsächlich auf traditionelle Systeme, wobei Dell versucht, seinen Kunden Open Networking Hardware in Form von Whitebox Switches und OS10 anzubieten. Cumulus Linux wird in der Schweiz von der Universität Lausanne, Nagravision und einem kleinen Service Provider mit Sitz in Genf verwendet. Die Vorteile von Open Networking, wie zum Beispiel der enorme Preisvorteil und die Möglichkeit zur totalen Automatisierung, werden wahrscheinlich aber auch in Österreich erkannt werden. [37]

Um Open Networking in einem Produktivsystem einzusetzen, ist es notwendig, auch die Bereitschaft und den Willen der Mitarbeiter für die Weiterbildung und die Innovation zu haben. Der größte Vorteil, der durch Open Networking entsteht, nämlich die freie Wahl des Betriebssystems und die Basis Linux, stellt Netzwerktechniker mit dem Erlernen von DevOps-Tools und Python vor neue Herausforderungen. Es wird notwendig sein, auch erfahrene Full-Stack-Developer und Linux-Administratoren in das Netzwerkteam aufzunehmen, um die Vorteile der Netzwerkautomatisierung und das Netzwerkmanagement nutzen zu können. Eine Integration von Open Networking in bestehende Rechenzentren sollte, wie von SDN-Lösungen bekannt, in neue Netzbereiche oder getrennte Infrastrukturen erfolgen. Dadurch und durch die Verwendung von Layer3-Verbindungen zwischen den Infrastrukturen ergeben sich, wie in den vorherigen Tests festgestellt wurde, keine Probleme bei der Kompatibilität und eine optimale Einbindung dieser Technologie in bestehende Netzwerke. Eine Ablösung ganzer Netzwerkinfrastrukturen und ein Umstieg von proprietären Systemen auf Open Networking ist ebenfalls möglich, sollte aber wie jeder Generationenwechsel schrittweise erfolgen. Diese Technologie hat sich in den letzten Jahren etabliert und die Betriebssysteme haben eine Stabilität erreicht, welche es erlaubt, Open Networking in Produktionssystemen einzusetzen. Diese Entwicklung ist nicht nur durch die großen Industriegiganten wie Microsoft oder Google sichtbar, sondern auch durch die große Community hinter Open Networking sowie die teilweise Öffnung von proprietären Betriebssystemen. Durch den Einsatz werden die Vorteile dieser Technologie zur Gänze ausgeschöpft und beim Betrieb der Netzwerkinfrastruktur die Flexibilität, die Kosteneffizienz, der Administrationsaufwand und die Automatisierungsrate gesteigert. Unter Berücksichtigungen aller Anforderungen an die Komponenten sowie an die Mitarbeiter ist es möglich, eine Netzwerkinfrastruktur auf Basis von Open Networking aufzubauen und als Produktivsysteme zu betreiben.

9. Ausblick und Zukunft

Die Entkoppelung von Hardware und Software wird seit Jahrzenten im Server-Bereich eingesetzt und ist mittlerweile state-of-the-art in der Branche, während die Möglichkeit des Einsatzes dieser Technologie im Netzwerk-Bereich für viele noch unbekannt ist. Trotzdem setzen viele branchenführende Serviceprovider und IT-Unternehmen Open Networking in den eigenen Rechenzentren ein und nutzen die großen Vorteile dieser Technologie. Bisher war die Trennung von Hardware und Software im Netzwerkbereich nur für Unternehmen geeignet, die es sich leisten konnten, genügend Geld in die Entwicklung solcher Systeme zu investieren. Durch die Investition in die Optimierung der Netzwerkinfrastruktur können Unternehmen Kosten einsparen und die Produktivität ihrer Systeme steigern, während der Administrationsaufwand für die Techniker sinkt. Durch den Kauf von Bare-Metal oder Whitebox Switches können nun auch Unternehmen, deren Budget keine Eigenentwicklungen ermöglicht, Open Networking in ihren Rechenzentren einsetzen und davon profitieren. Der Einsatz in dieser Technologie schreitet voran, wie man im „Gartner Hype Cycle for Enterprise Networking and Communications, 2017“ [62] nachlesen kann. Das Open Compute Networking Project wird an den Höhepunkt des Hype Cycles positioniert, während Britebox-Switching diesen erst vor kurzem erreicht und damit die Schwelle überwunden hat. [62] Im Jahr 2015 wurde Whitebox Switching noch am Höhepunkt des Zyklus gelistet, während es in neueren Versionen gar nicht mehr aufscheint. [63] Durch den produktiven Einsatz dieser Technologie bei vielen führenden Unternehmen der IT-Branche ist davon auszugehen, dass Open Networking den Gartner Hype Cycle bereits teileweise verlassen hat und das „Plateau der Produktivität“ erreicht hat.

Hersteller von Open Networking Switches erweitern ihr Portfolio laufend und veröffentlichen auch teilweise die Switch-Designs, wodurch die Kompatibilität zu Open-Source-Projekten gefördert werden soll. Aber auch Hersteller von proprietären Lösungen öffnen ihre Systeme zumindest teilweise für die Orchestrierung und Automatisierung. Manche Hersteller setzen hingegen komplett auf Open Networking als Verkaufsargument und verwenden Open-Source-Schnittstellen und -Elemente im Betriebssystem. Während Dell den Verkauf von Open Networking Switches forciert und das hauseigene Betriebssystem OS10 auf einer Open-Source-Basis aufgebaut hat, setzt Cisco zwar weiterhin auf ein geschlossenes System aus Hardware und Software, öffnet aber zumindest teilweise Betriebssysteme für den Einsatz verschiedener Orchestrierungs- und Managementtools sowie Open-Source-Schnittstellen wie SAI. Durch die weitere Forcierung vieler Switch-Hersteller wird die Disaggregation von Hardware und Software auch Hersteller von proprietären Lösungen nicht erspart bleiben. Cisco bietet bereits jetzt mit IOS on Unix eine Möglichkeit, das Switch-Betriebssystem IOS auf einem Unix-Betriebssystem als Dienst zu starten. Diese Entwicklung dient zwar zurzeit genauso wie NX-OSv nur zu Testzwecken, kann jedoch als erster Schritt in Richtung Open Networking gesehen werden. Dass Cisco bereits unter dem Druck, welcher von Open Networking ausgeht, steht, sieht man auch an der Einführung von IOS-XE, einem Unix-basierten, jedoch noch geschlossenen, Betriebssystem für Campus Switches. Das Datacenter-Pendant dazu, NX-OS, ist für seine vielen offenen Schnittstellen bekannt, funktioniert aber ebenfalls nur auf Hardware von Cisco. Während Cisco mittelfristig eine Öffnung der eigenen Hardware durchführen muss und wohl auch wird, hat Dell diese bereits vorgenommen und ist stehts bemüht, die Kompatibilität vieler führender Open Networking NOS zu seinen Switches zu ermöglichen. Jedoch ist Dell von der Zuarbeit der Entwickler dieser Betriebssysteme abhängig. Durch die Offenlegung der Hardware-Schnittstelle und die Forcierung von Open-Source-Projekten wie OPX kann die Kompatibilität auch in Zukunft gesteigert und erweitert werden. Viele andere Hersteller von Open Networking Switches werden ebenfalls die Unterstützung der eigenen Hardware vorantreiben wollen und ebenso die Hardware-Schnittstellen oder die Designs beim Open Computer Project offenlegen, um optimale Bedingungen für Portierungen der Betriebssysteme zu schaffen.

Die Entwickler der Betriebssysteme schlagen ebenfalls neue Wege ein, da viele Unternehmen zwar auf die Verwendung verschiedenster, für den Anwendungsfall passender, Betriebssysteme setzen wollen, jedoch an einheitlichen Schnittstellen auf allen Netzwerkkomponenten interessiert sind. SONiC und FBOSS sind

die ersten Entwicklungen in diese Richtung und sind bei weitem noch nicht für den Einsatz in Produktivsystemen geeignet, bieten aber vor allem in Cloud-Infrastrukturen viele zusätzliche Vorteile. Die Verwendung von Agents auf einem beliebigen NOS zur Steuerung des Switches ermöglicht auch für Hersteller traditioneller Systeme neue Möglichkeiten zur Einführung von Open Networking, wie man anhand von Cisco und dem SAI bzw. SONiC sieht. Außerdem setzt man in Zukunft vermehrt auf eigenständige Applikationen im Ecosystem, um den Kunden die optimalen Features zu bieten. Unter dem Sammelbegriff „Open Networking 2.0“ sind all jene Entwicklungen gemeint, welche eine Entkoppelung der Bestandteile der Software und des Ecosystems oder die Steuerung mithilfe zentraler Agents durchführen. In Zukunft soll es dadurch möglich werden, ein beliebiges Betriebssystem auf einem beliebigen Switch zu betreiben. Dieses soll nur die benötigten Applikationen und Features ohne Overhead bieten und mit zentralen Agents gesteuert werden. Mithilfe von „Open Networking 2.0“ kann dann ein hochskalierbares, flexibles, zentral steuerbares und anpassbares Netzwerkbetriebssystem auf jedem beliebigen Switch betrieben werden.

10. Literaturverzeichnis

[1] Dell Inc., „White Paper on Open-Networking," 07 2014. [Online]. Available: http://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/White-Paper-on-Dell-Open-Networking.pdf. [Zugriff am 02 04 2018].

[2] AIRHEADS Community Aruba Networks, „What is Open Networking?," 21 03 2016. [Online]. Available: http://community.arubanetworks.com/t5/Technology-Blog/What-is-Open-Networking/ba-p/260377. [Zugriff am 15 04 2018].

[3] D. Root, „Open Networking - The Foundation for Future-Ready IT," Dell EMC, 27 06 2016. [Online]. Available: https://blog.dellemc.com/en-us/open-networking-the-foundation-for-future-ready-it/. [Zugriff am 14 04 2018].

[4] Open Networking Foundation, „ONF - Our Mission," [Online]. Available: https://www.opennetworking.org/mission/. [Zugriff am 12 04 2018].

[5] J. Doyle, „Clearing the fog around open switching terminology," Network World, 01 07 2015. [Online]. Available: https://www.networkworld.com/article/2919599/cisco-subnet/clearing-the-fog-around-open-switching-terminology.html. [Zugriff am 16 04 2018].

[6] Pica8, „Pica8 Enterprise Solution Brief," [Online]. Available: https://www.pica8.com/wp-content/uploads/Pica8-Enterprise-Solution-Brief.pdf. [Zugriff am 04 17 2018].

[7] Dell, „Dell-EMC-Networking-S4048-ON-Spec_Sheet," 03 2017. [Online]. Available: http://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Dell-EMC-Networking-S4048-ON-Spec-Sheet.pdf. [Zugriff am 28 03 2018].

[8] Dell EMC, „Dell EMC Open-Networking," 2017. [Online]. Available: http://en.community.dell.com/techcenter/networking/w/wiki/7463.dell-emc-open-networking. [Zugriff am 12 04 2018].

[9] Open Compute Project, „Networking - SepcsAndDesigns Accepted Hardware," 2018. [Online]. Available: http://www.opencompute.org/wiki/Networking/SpecsAndDesigns#Accepted_Hardware. [Zugriff am 04 12 2018].

[10] Open Compute Project, „Facebook - Wedge 100 32x100G Top of Rack Switch," 14 09 2016. [Online]. Available: http://files.opencompute.org/oc/public.php?service=files&t=80dde82c251e1c3f5ad8ff7fd2675061. [Zugriff am 17 04 2018].

[11] Facebook, „Wedge 100: More open and versatile than ever," 18 10 2016. [Online]. Available: https://code.facebook.com/posts/1802489260027439/wedge-100-more-open-and-versatile-than-ever/. [Zugriff am 12 03 2018].

[12] Facebook, „Introducing "Wedge" and "FBOSS", the next steps toward a disaggregated network," 18 07 2014. [Online]. Available: https://code.facebook.com/posts/681382905244727/introducing-wedge-and-fboss-the-next-steps-toward-a-disaggregated-network/. [Zugriff am 12 03 2018].

[13] Facebook, „Opening designs for 6-pack and Wedge 100," 9 03 2016. [Online]. Available: https://code.facebook.com/posts/203733993317833/opening-designs-for-6-pack-and-wedge-100/. [Zugriff am 12 03 2018].

[14] Cumulus Network Linux, „Bare Metal Switches - Is there a cost benefit?," 2015. [Online]. Available: https://cumulusnetworks.com/media/cumulus/pdf/misc/Business-Brief-Capex.pdf. [Zugriff am 02 04 2018].

[15] Pica8, „Bare Metal Networking-Leveraging "White Box" Thinking," 2014. [Online]. Available: https://www.pica8.com/wp-content/uploads/pica-whitepaper-white-box.pdf. [Zugriff am 17 04 2018].

[16] IT-Wissen, „ASIC (application specific integrated circuit)," 12 02 2013. [Online]. Available: https://www.itwissen.info/ASIC-application-specific-integrated-circuit.html. [Zugriff am 19 04 2018].

[17] T. Hollingsworth, „Inside White-Box Switches," 13 08 2014. [Online]. Available: https://www.networkcomputing.com/networking/inside-white-box-switches/666746411. [Zugriff am 04 19 2018].

[18] e. a. U. S. D. Nathan Farrington, „Data Center Switch Architecture in the Age of Merchant Silicon," 25 08 2009. [Online]. Available: http://cseweb.ucsd.edu/~vahdat/papers/hoti09.pdf. [Zugriff am 19 04 2018].

[19] O. C. Project, „Networking/ONIE," 2018. [Online]. Available: http://www.opencompute.org/wiki/Networking/ONIE. [Zugriff am 20 04 2018].

[20] OpenComputeProject, „ONIE Project Overview," 2018. [Online]. Available: https://opencomputeproject.github.io/onie/overview/index.html#project-overview. [Zugriff am 04 25 2018].

[21] E. Banks, „What is ONIE?," 14 07 2014. [Online]. Available: http://ethancbanks.com/2014/07/17/what-is-onie-open-network-install-environment/. [Zugriff am 24 04 2018].

[22] A. Liu, „Open Network Install Environment (ONIE) Instruction," 27 10 2017. [Online]. Available: http://www.fiberopticshare.com/open-network-install-environment-onie-introduction.html. [Zugriff am 25 04 2018].

[23] Broadcom, „Github OF-DPA," 2018. [Online]. Available: https://github.com/Broadcom-Switch/of-dpa/. [Zugriff am 26 04 2018].

[24] Broadcom, „Github OpenNSL," 2018. [Online]. Available: https://github.com/Broadcom-Switch/OpenNSL/. [Zugriff am 26 04 2018].

[25] OpenNetLinux, „ONL FAQ," 2018. [Online]. Available: https://opennetlinux.org/faq. [Zugriff am 2 04 2018].

[26] GNS3, „The growing Open Networking Ecosystem," 16 07 2015. [Online]. Available: https://www.gns3.com/news/article/the-growing-open-networking-ecos. [Zugriff am 04 28 2018].

[27] S. Cavanaugh, „Automation for Network Engineers," 08 25 2017. [Online]. Available: https://support.cumulusnetworks.com/hc/en-us/articles/206336458-Automation-for-Network-Engineers. [Zugriff am 22 04 2018].

[28] IETF, „Network Configuration Protocol (NETCONF)," 06 2011. [Online]. Available: https://tools.ietf.org/html/rfc6241. [Zugriff am 05 05 2018].

[29] IETF, „YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," 10 2010. [Online]. Available: https://tools.ietf.org/html/rfc6241. [Zugriff am 05 05 2018].

[30] Cumulus Networks, „Network Automation Best Practices for DevOps," 2018. [Online]. Available: https://cumulusnetworks.com/blog/network-automation-best-practices-devops/. [Zugriff am 05 04 2018].

[31] M. Rouse, „zero touch provisioning (ZTP)," 2018. [Online]. Available: https://searchitoperations.techtarget.com/definition/zero-touch-provisioning-ZTP. [Zugriff am 02 04 2018].

[32] L. MacVittie, „SDX Central: Why Network Automation Needs DevOps," 21 10 2016. [Online]. Available: https://www.sdxcentral.com/articles/contributed/why-network-automation-needs-devops/2016/10/. [Zugriff am 01 04 2018].

[33] Cisco, „Embrace netdevops part 1," 2018. [Online]. Available: https://blogs.cisco.com/developer/embrace-netdevops-part-1. [Zugriff am 01 02 2018].

[34] Cisco, „Cisco NX-OS," 2018. [Online]. Available: https://www.cisco.com/c/en/us/products/ios-nx-os-software/nx-os/index.html. [Zugriff am 02 04 2018].

[35] Cisco, „Cisco IOS-XE," 2018. [Online]. Available: https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-xe/index.html. [Zugriff am 02 05 2018].

[36] J. Networks, „JunOS," 2018. [Online]. Available: https://www.juniper.net/us/en/products-services/nos/junos/. [Zugriff am 12 02 2018].

[37] H. Kuehn, Interviewee, *Fragen zur Open Networking Politik von Dell.* [Interview]. 24 04 2018.

[38] OpenSwitch, „OpenSwitch OPX Datasheet," 2018. [Online]. Available: https://www.openswitch.net/wp-content/uploads/sites/31/2018/04/OpenSwitch_OPX_datasheet.pdf. [Zugriff am 22 04 2018].

[39] C. Networks, „Cumulus Linux," 2018. [Online]. Available: https://cumulusnetworks.com/products/cumulus-linux/. [Zugriff am 02 05 2018].

[40] C. Networks, „Understanding web-scale networking," 2018. [Online]. Available: https://cumulusnetworks.com/learn/web-scale-networking-education/. [Zugriff am 19 04 2018].

[41] Cumulus Network Linux, „Cumulus Linux Architecture," 2018. [Online]. Available: https://cumulusnetworks.com/learn/web-scale-networking-resources/product-collateral/cumulus-linux-architecture/. [Zugriff am 07 05 2018].

[42] C. N. Linux, „Cumulus Linux Architecture," 2018. [Online]. Available: https://cumulusnetworks.com/learn/web-scale-networking-resources/product-collateral/cumulus-linux-architecture/. [Zugriff am 07 05 2018].

[43] S. Cavamaigj, 08 01 2018. [Online]. Available: https://support.cumulusnetworks.com/hc/en-us/articles/115010587028. [Zugriff am 07 05 2018].

[44] C. Networks, „Securing Cumulus Linux," 2018. [Online]. Available: https://cumulusnetworks.com/learn/web-scale-networking-resources/white-papers/securing-cumulus-linux/. [Zugriff am 02 05 2018].

[45] B. Carrol, „Who is IP Infusion and Why Have You Never Heard of Them?," 11 04 2017. [Online]. Available: http://datanetworkingtalk.com/who-is-ip-infusion-and-why-have-you-never-heard-of-them/. [Zugriff am 2 05 2018].

[46] IP Infusion, „OcNOS," 2018. [Online]. Available: https://www.ipinfusion.com/products/ocnos/. [Zugriff am 02 05 2018].

[47] ipinfusion, „OcNOS Technical Documentation," 2018.

[48] XORP, „XORP," 2011. [Online]. Available: http://www.xorp.org/. [Zugriff am 07 05 2018].

[49] Wikipedia, „Wikipedia Pica8," 2018. [Online]. Available: https://en.wikipedia.org/wiki/Pica8. [Zugriff am 07 05 2018].

[50] Pica8, „PicOS Overview," 2018. [Online]. Available: https://www.pica8.com/wp-content/uploads/pica8-whitepaper-picos-overview.pdf. [Zugriff am 07 05 2018].

[51] Pica8, „PicOS 2.11 Configuration Guide," Pica8, 28 03 2018. [Online]. Available: https://docs.pica8.com/display/PICOS2111cg. [Zugriff am 02 04 2018].

[52] lgouhan, „Github Azure/SONiC Architecture," 13 09 2017. [Online]. Available: https://github.com/Azure/SONiC/wiki/Architecture. [Zugriff am 07 05 2018].

[53] L. Hadesty, „SDX-Central: Cisco Gets on Board Microsoft's SONiC Train," 28 07 2017. [Online]. Available: https://www.sdxcentral.com/articles/news/cisco-gets-board-microsofts-sonic-train/2017/06/. [Zugriff am 09 05 2018].

[54] Facebook, „Facebook Open Switching System FBOSS and Wedge in the open," 10 03 2015. [Online]. Available: https://code.facebook.com/posts/843620439027582/facebook-open-switching-system-fboss-and-wedge-in-the-open/. [Zugriff am 07 05 2018].

[55] Github, „Github FBOSS," 2018. [Online]. Available: https://github.com/facebook/fboss. [Zugriff am 08 05 2018].

[56] Cumulus Networks, „Comparing Traditional Bridge Mode to VLAN-aware Bridge Mode," Cumulus Networks, 06 12 2016. [Online]. Available: https://support.cumulusnetworks.com/hc/en-us/articles/204909397. [Zugriff am 26 03 2018].

[57] Cumulus Networks, „Cumulus Linux User Guide 3.5.2," Cumulus Networks, 02 2018. [Online]. Available:

https://docs.cumulusnetworks.com/download/attachments/7112747/Cumulus%20Linux%203.5.3%20User%20Guide.pdf?version=1&modificationDate=1520028092000&api=v2. [Zugriff am 26 03 2018].

[58] S. Eens, Interviewee, *Questions about Open Networking.* [Interview]. 12 03 2018.

[59] *OCPUS18 – Project Greenfield – Building the Nextgen Datacenter at Adobe.* [Film]. Cumulus Networks, Adobe, 2018.

[60] Cumulus Linux, „NTT Builds a Flexible and Simple network with Cumulus Linux," 2018. [Online]. Available: https://cumulusnetworks.com/customers/ntt/. [Zugriff am 09 05 2018].

[61] IP Infusion, „The London Internet Exchange to Deploy IP Infusion's OcNOS™ Network Operating System in New London Interconnect Platform," 08 05 2018. [Online]. Available: https://www.ipinfusion.com/news-events/the-london-internet-exchange-to-deploy-ip-infusions-ocnos-network-operating-system-in-new-london-interconnect-platform/. [Zugriff am 09 05 2018].

[62] Gartner, „Hype Cycle for Enterprise Networking and Communications, 2017," 20 07 2017. [Online]. Available: https://www.gartner.com/doc/3764377/hype-cycle-enterprise-networking-communications. [Zugriff am 05 16 201].

[63] Gartner, „Hype Cycle for Networking and Communications, 2015," 07 27 2015. [Online]. Available: https://www.gartner.com/doc/3100229/hype-cycle-networking-communications-. [Zugriff am 10 05 2018].

[64] Moor Insights & Strategy, „Open Networking Continues Momentum," 29 03 2016. [Online]. Available: http://www.moorinsightsstrategy.com/wp-content/uploads/2016/03/Open-Networking-Continues-Momentum-by-Moor-Insights-and-Strategy.pdf. [Zugriff am 10 03 2018].

[65] Pluribus Networks, „Open Networking FAQs," 2018. [Online]. Available: https://www.pluribusnetworks.com/resources/open-networking-faq/. [Zugriff am 10 04 2018].

[66] E. Banks, „SDN-Grundlagen: Zentrale Kontrolle und Programmierbarkeit," Packet Pushers Interactive, 02 2014. [Online]. Available: https://www.searchnetworking.de/lernprogramm/SDN-Grundlagen-Zentrale-Kontrolle-und-Programmierbarkeit. [Zugriff am 17 03 2018].

[67] Pluribus Networks, „Open-Networking FAQ," 2018. [Online]. Available: https://www.pluribusnetworks.com/resources/open-networking-faq/. [Zugriff am 10 04 2018].

[68] Colfax Direct, „Colfax Direct Quanta BMS T7032-IX1 32-Portt 100Gbe Switch with Cumulus Linux," 2018. [Online]. Available: http://www.colfaxdirect.com/store/pc/viewPrd.asp?idproduct=3205&idcategory=7. [Zugriff am 12 04 2018].

[69] Metaswitch, „Protocl Stacks for OEMs," 2018. [Online]. Available: https://www.metaswitch.com/products/networking-software/protocol-stacks-for-oem. [Zugriff am 08 03 2018].

[70] BigSwitch Networks, „Big Monitoring Fabric," 2018. [Online]. Available: https://www.bigswitch.com/sdn-products/sdn-products/big-monitoring-fabric/overview. [Zugriff am 05 03 2018].

[71] BigSwitch Networks, „Big Cloud Fabric," 2018. [Online]. Available: https://www.bigswitch.com/products/big-cloud-fabrictm/switching-fabric-overview. [Zugriff am 08 03 2018].

[72] R. Chirgwin, „Cisco seperates switching and routing software from hardware," 28 03 2018. [Online]. Available: https://www.theregister.co.uk/2018/03/28/cisco_disaggregation_strategy/. [Zugriff am 02 05 2018].

[73] Edge-Core, „Pica8 Adopts ONIE (Open Network Install Environment) Simplifies Deployment of Bare Metal Switches with Pica8 Operating System," 07 10 2014. [Online]. Available: https://www.edge-core.com/news-inquiry.php?cls=1&id=60. [Zugriff am 28 03 2018].

[74] R. Chirgwin, „Everybody loves Microsoft's open switch software, SONiC,“ 22 03 2018. [Online]. Available: https://www.theregister.co.uk/2018/03/22/networking_news_roundup/. [Zugriff am 02 04 2018].

[75] Microsoft, „Github Troubleshooting Guide SONiC,“ 2018. [Online]. Available: https://github.com/Azure/SONiC/wiki/Troubleshooting-Guide. [Zugriff am 02 05 2018].

[76] B. Carroll, „Who is IP Infusion and Why Have You Never Heard of Them?,“ 11 04 2017. [Online]. Available: http://datanetworkingtalk.com/who-is-ip-infusion-and-why-have-you-never-heard-of-them/. [Zugriff am 10 04 2018].

[77] DellEMC, „Let The Transformation Beign - Open networking,“ 2018. [Online]. Available: http://dellemcevents.com/uploads/03_Morocco-Forum_All-Flash-presentation_Mai-2017.pptx.pdf. [Zugriff am 02 04 2018].

[78] P. L. Ventre und et.al, „SDN-Based IP and Layer 2 Services with an Open Networking Operating System in the GÉANT Service Provider Network,“ 17 04 2017. [Online]. Available: https://ezproxy.fhstp.ac.at:2329/document/7901480/. [Zugriff am 02 04 2018].

[79] D. Fritzsche und e. al, „Basebox — Integrating Whitebox Switches into Linux: A Controller Implementation for OF-DPA Hardware,“ 26 06 2017. [Online]. Available: https://ezproxy.fhstp.ac.at:2329/document/7956052/. [Zugriff am 30 04 2018].

[80] S. Central, „Open Networking reveals its fractal nature,“ 09 09 2016. [Online]. Available: https://www.sdxcentral.com/articles/contributed/open-networking-reveals-its-fractal-nature/2016/09/. [Zugriff am 18 05 2018].