Software Defined Networking
Zukunft des Netzes?

Zdravko Bozakov
z@bozakov.de

GUUG-Frühjahrsfachgespräch
27. März, 2015
Stuttgart

The Big Picture...

The Big Picture...
Software Defined Networking

The Big Picture

Warum ein neuer Ansatz für Netzwerke?

- Netzwerke vereinen eine Vielzahl von Technologien und Protokollen (mit Abhängigkeiten)
- Komplexität muss von Netzwerkadminstratoren bewältigt werden
- Konfiguration über proprietäre Schnittstellen (z.B. CLI)
- Änderungen in der Kontrollebene benötigen Updates der Netzwerkgeräte
- dezentrale Algorithmen sind schwer

Die Komplexität in heutigen Netzwerken erschwert Innovationen

. . . sowie den wissenschaftlichen Erkenntnisgewinn

The Big Picture

Warum ein neuer Ansatz für Netzwerke?

- Netzwerke vereinen eine Vielzahl von Technologien und Protokollen (mit Abhängigkeiten)
- Komplexität muss von Netzwerkadminstratoren bewältigt werden
- Konfiguration über proprietäre Schnittstellen (z.B. CLI)
- Änderungen in der Kontrollebene benötigen Updates der Netzwerkgeräte
- dezentrale Algorithmen sind schwer

Die Komplexität in heutigen Netzwerken erschwert Innovationen

. . . sowie den wissenschaftlichen Erkenntnisgewinn

The Big Picture

Warum ein neuer Ansatz für Netzwerke?

- Netzwerke vereinen eine Vielzahl von Technologien und Protokollen (mit Abhängigkeiten)
- Komplexität muss von Netzwerkadminstratoren bewältigt werden
- Konfiguration über proprietäre Schnittstellen (z.B. CLI)
- Änderungen in der Kontrollebene benötigen Updates der Netzwerkgeräte
- dezentrale Algorithmen sind schwer

Die Komplexität in heutigen Netzwerken erschwert Innovationen

. . . sowie den wissenschaftlichen Erkenntnisgewinn

Das Software-Defined-Networking Paradigma hat das Ziel die Frage zu beantworten:

Welche Abstraktionsebenen können den Betrieb von Computernetzwerken verbessern und gleichzeitig den Gewinn neuer Erkenntnisse fördern?

. . . um neue Dienste effizient zu entwickeln und schnell in Betrieb zu nehmen

Das Software-Defined-Networking Paradigma hat das Ziel die Frage zu beantworten:

> Welche Abstraktionsebenen können den Betrieb von Computernetzwerken verbessern und gleichzeitig den Gewinn neuer Erkenntnisse fördern?

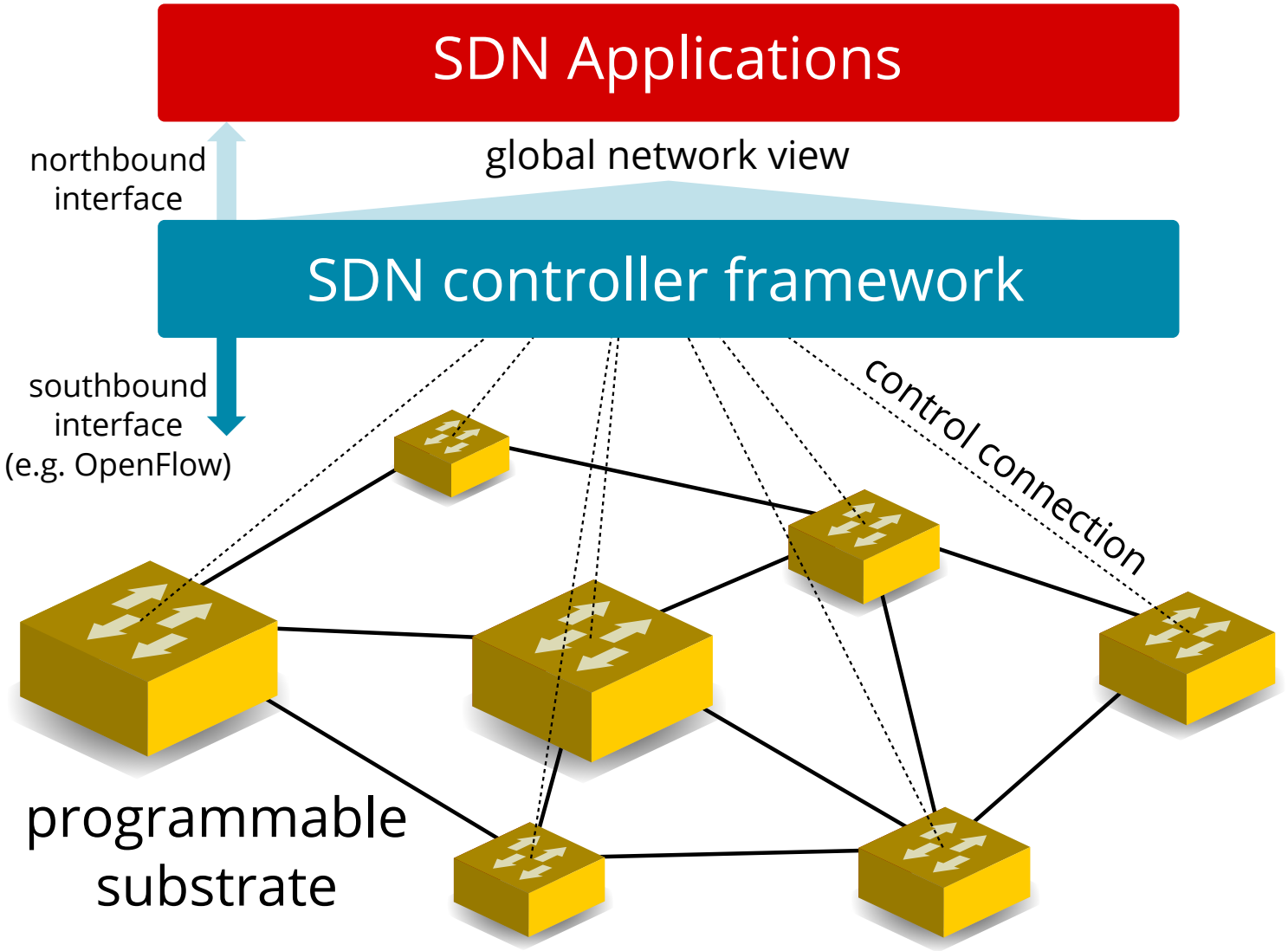
. . . um neue Dienste effizient zu entwickeln und schnell in Betrieb zu nehmen

Abstraktionen werde extrem erfolgreich in den Betriebssystem- und Software-Engineering Bereichen angewendet:

- Komplexität wird durch Abstraktionen verborgen (z.B. virtueller Speicher, Dateisysteme, virtuelle Maschinen)
- Automatisierung/Optimierung (Compiler, Caching, Multitasking)
- passende Tools für unterschiedliche Probleme (Assembler $\rightarrow$ C $\rightarrow$ Fortran $\rightarrow$ Java $\rightarrow$ Python)
- Entwickler können sich auf die Lösung von Problemen konzentrieren
- Wesentlich schnellerer Entwicklungszyklus im Softwarebereich

Das SDN Modell führt drei Abstraktionen ein

Spezifikation	Das Verhalten von Netzwerkdiensten wird durch ein zentralisiertes Kontrollprogram definiert
Verteilter Netzzustand	Allen Diensten steht eine globale Sicht des Netzwerks zur Verfügung
Konfiguration	Implementierungsdetails im Datenpfad sind vom Kontrollprogram verborgen


SDN Applications
northbound interface
global network view
SDN controller framework
southbound interface (e.g. OpenFlow)
control connection
programmable substrate

SDN Plattformen

Interoperabilität und offene Schnittstellen sind wichtiger Bestandteil der SDN Idee

Es existieren mehrere Open Source Controller-Frameworks mit unterschiedlichen Schwerpunkten

- NOX/POX (C/Python)
- Floodlight, OpenDaylight (Java)
- Trema (Ruby)
- Ryu (Python)
- u.v.m. . . .

. . . sowie kommerzielle *"SDN"* Produkte

- VMware NSX, Cisco ACI, Open Contrail, Big Switch

SDN Plattformen

Interoperabilität und offene Schnittstellen sind wichtiger Bestandteil der SDN Idee

Es existieren mehrere Open Source Controller-Frameworks mit unterschiedlichen Schwerpunkten

- NOX/POX (C/Python)
- Floodlight, OpenDaylight (Java)
- Trema (Ruby)
- Ryu (Python)
- u.v.m. . . .

. . . sowie kommerzielle *"SDN"* Produkte

- VMware NSX, Cisco ACI, Open Contrail, Big Switch

Interessante SDN Projekte

- höhere Programmiersprachen zur Beschreibung von Netzwerklogik und Konnektivität (Compiler/Debugger z.B. Frenetic)
- automatisierte Verifikation von Netzwerkregeln (z.B. ACL)
- transparente Migration von Netzwerkzuständen

OpenFlow

Was ist OpenFlow?

Offene Schnittstelle zur externen Programmierung der Forwardingtabellen in kommerziellen Switches

Southbound-Interface für SDN

Was ist OpenFlow?

Offene Schnittstelle zur externen Programmierung der Forwardingtabellen in kommerziellen Switches

Southbound-Interface für SDN

OpenFlow: Motivation

Begann als Forschungsprojekt (Stanford + Berkley) mit dem Ziel Forschung in Produktivnetzen zu ermöglichen. OpenFlow wurde *vor* SDN entwickelt!

OpenFlow sollte akute Probleme beheben:

- Vermeidung von Vendor Lock-In
- Reduktion von Kosten
- Benutzer-definierte Kontrolllogik
- Wiederverwendbarkeit von Softwarekomponenten

. . . und langfristige Ziele erreichen:

- Stagnation im Netzwerkbereich verhindern (Ossification)
- Verkürzung der Zeit zwischen Forschungsideen und deren Umsetzung in Produktivnetzen
- Neue Herangehensweisen in der Netzwerkforschung

OpenFlow: Motivation

Begann als Forschungsprojekt (Stanford + Berkley) mit dem Ziel Forschung in Produktivnetzen zu ermöglichen. OpenFlow wurde *vor* SDN entwickelt!

OpenFlow sollte akute Probleme beheben:

- Vermeidung von Vendor Lock-In
- Reduktion von Kosten
- Benutzer-definierte Kontrolllogik
- Wiederverwendbarkeit von Softwarekomponenten

. . . und langfristige Ziele erreichen:

- Stagnation im Netzwerkbereich verhindern (Ossification)
- Verkürzung der Zeit zwischen Forschungsideen und deren Umsetzung in Produktivnetzen
- Neue Herangehensweisen in der Netzwerkforschung

OpenFlow: Motivation

Begann als Forschungsprojekt (Stanford + Berkley) mit dem Ziel Forschung in Produktivnetzen zu ermöglichen. OpenFlow wurde *vor* SDN entwickelt!

OpenFlow sollte akute Probleme beheben:

- Vermeidung von Vendor Lock-In
- Reduktion von Kosten
- Benutzer-definierte Kontrolllogik
- Wiederverwendbarkeit von Softwarekomponenten

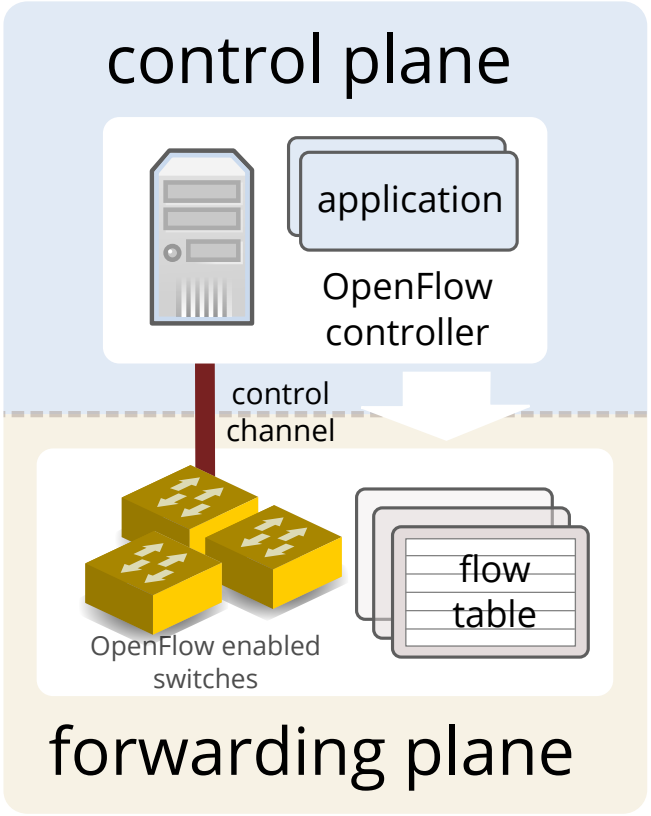
. . . und langfristige Ziele erreichen:

- Stagnation im Netzwerkbereich verhindern (Ossification)
- Verkürzung der Zeit zwischen Forschungsideen und deren Umsetzung in Produktivnetzen
- Neue Herangehensweisen in der Netzwerkforschung

OpenFlow: Architektur

OpenFlow lagert die Kontrolllogik von kommerziellen Switches auf einen externen Controller aus

- OpenFlow Switches mit programmierbaren Forwarding-Tabellen
- logisch zentralisierter Controller
- Kommunikation über Kontrollkanal
  (SSL, in-band/out-of-band)

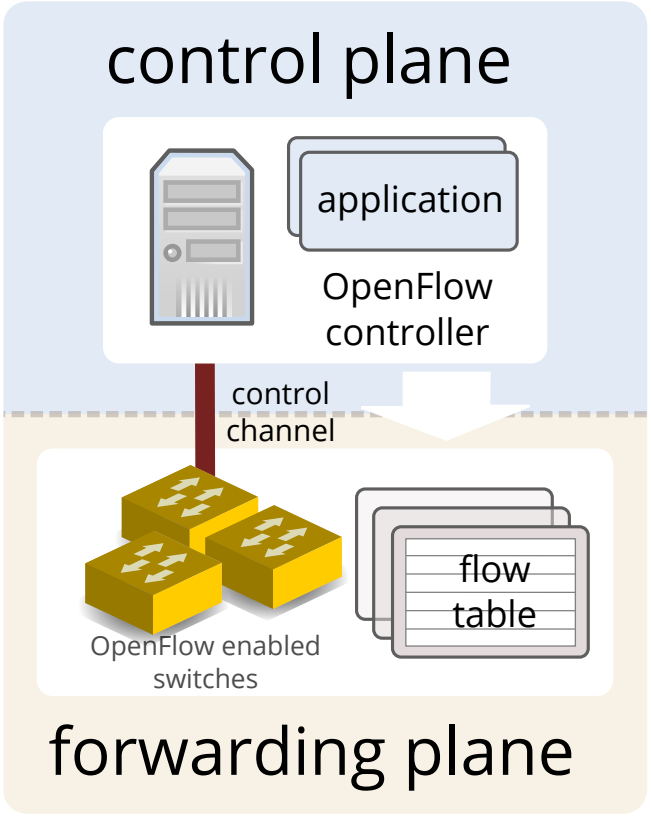


OpenFlow spezifiziert die API sowie das Kommunikationprotokoll

OpenFlow: Architektur

OpenFlow lagert die Kontrolllogik von kommerziellen Switches auf einen externen Controller aus

- OpenFlow Switches mit programmierbaren Forwarding-Tabellen
- logisch zentralisierter Controller
- Kommunikation über Kontrollkanal
  (SSL, in-band/out-of-band)

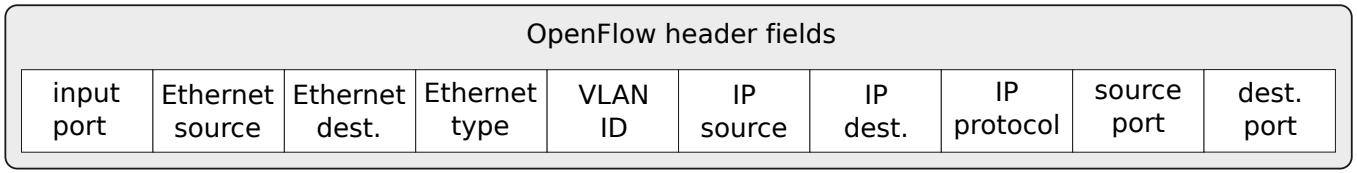


OpenFlow spezifiziert die API sowie das Kommunikationprotokoll

OpenFlow: Flow Tabelle

OpenFlow Switches werden als Flow-Tabellen abstrahiert.
Jede Tabelle enthält “Flow-Einträge”:

Match Header-Bitfolge mit der ankommende Pakete verglichen werden



Actions Forwarding- und Verarbeitungs-Operationen, die auf jedes passende Paket ausgeführt werden sollen
z.B.: `drop`, `out_port`, `send_to_controller`, `modify_IP_dst`, `set_VLAN_ID`, `...`

Counters zählen das Datenvolumen des “Flows”

OpenFlow: Beispiel

"Für allen IP Paketen mit TCP Port 8182 und dem Ziel 10.1.1.0/24, ändere die L2-Adressen und leite sie an Ports 1 und 2 weiter! "

Datapath ID: 1234567890
ofp_flow_mod: OFPC_ADD

OpenFlow header fields

input port	Ethernet source	Ethernet dest.	Ethernet type	VLAN ID	IP source	IP dest.	IP protocol	source port	dest. port
*	*	*	0x0800	*	*	10.1.1.0/24	0x06	*	8182

OpenFlow actions

modify eth src: 0a:76:06:ff:ff:00
modify eth dst: 00:4e:46:32:43:00
out port: 1
out port: 3

Controller generiert eine OF-Protokoll-Nachricht `ofp_flow_add` und sendet diese über den Kontrollkanal an den Switch

OpenFlow: Zusammenfassung

OpenFlow Schlüsselpunkte

- trennt die Kontroll- und den Datenpfade über eine klar definierte Schnittstelle
- abstrahiert Switches als programmierbare Flow-Tabellen (Konfigurations-Abstraktion)
- Implementierungsdetails des Herstellers müssen nicht offengelegt werden (geistiges Eigentum)
- viele Hersteller bieten OpenFlow Switches an
- OpenFlow wird bereits in Produktivnetzen eingesetzt (z.B. Google)
- extrem erfolgreiches Produkt aus der Forschung

OpenFlow specification is managed by the Open Networking Foundation `www.opennetworking.org`

OpenFlow: Zusammenfassung

OpenFlow Schlüsselpunkte

- trennt die Kontroll- und den Datenpfade über eine klar definierte Schnittstelle
- abstrahiert Switches als programmierbare Flow-Tabellen (Konfigurations-Abstraktion)
- Implementierungsdetails des Herstellers müssen nicht offengelegt werden (geistiges Eigentum)
- viele Hersteller bieten OpenFlow Switches an
- OpenFlow wird bereits in Produktivnetzen eingesetzt (z.B. Google)
- extrem erfolgreiches Produkt aus der Forschung

OpenFlow specification is managed by the Open Networking Foundation `www.opennetworking.org`

Experimentieren mit OpenFlow

Wie kann man OpenFlow ausprobieren?

1. OpenVSwitch installieren
2. Mininet installieren (Netwzerk Emulator)
3. Controller installieren (z.B. POX)

Wie kann man OpenFlow ausprobieren?

1. OpenVSwitch installieren
2. Mininet installieren (Netwzerk Emulator)
3. Controller installieren (z.B. POX)

Wie kann man OpenFlow ausprobieren?

1. OpenVSwitch installieren
2. Mininet installieren (Netwzerk Emulator)
3. Controller installieren (z.B. POX)

Wie kann man OpenFlow ausprobieren?

1. OpenVSwitch installieren
2. Mininet installieren (Netwzerk Emulator)
3. Controller installieren (z.B. POX)

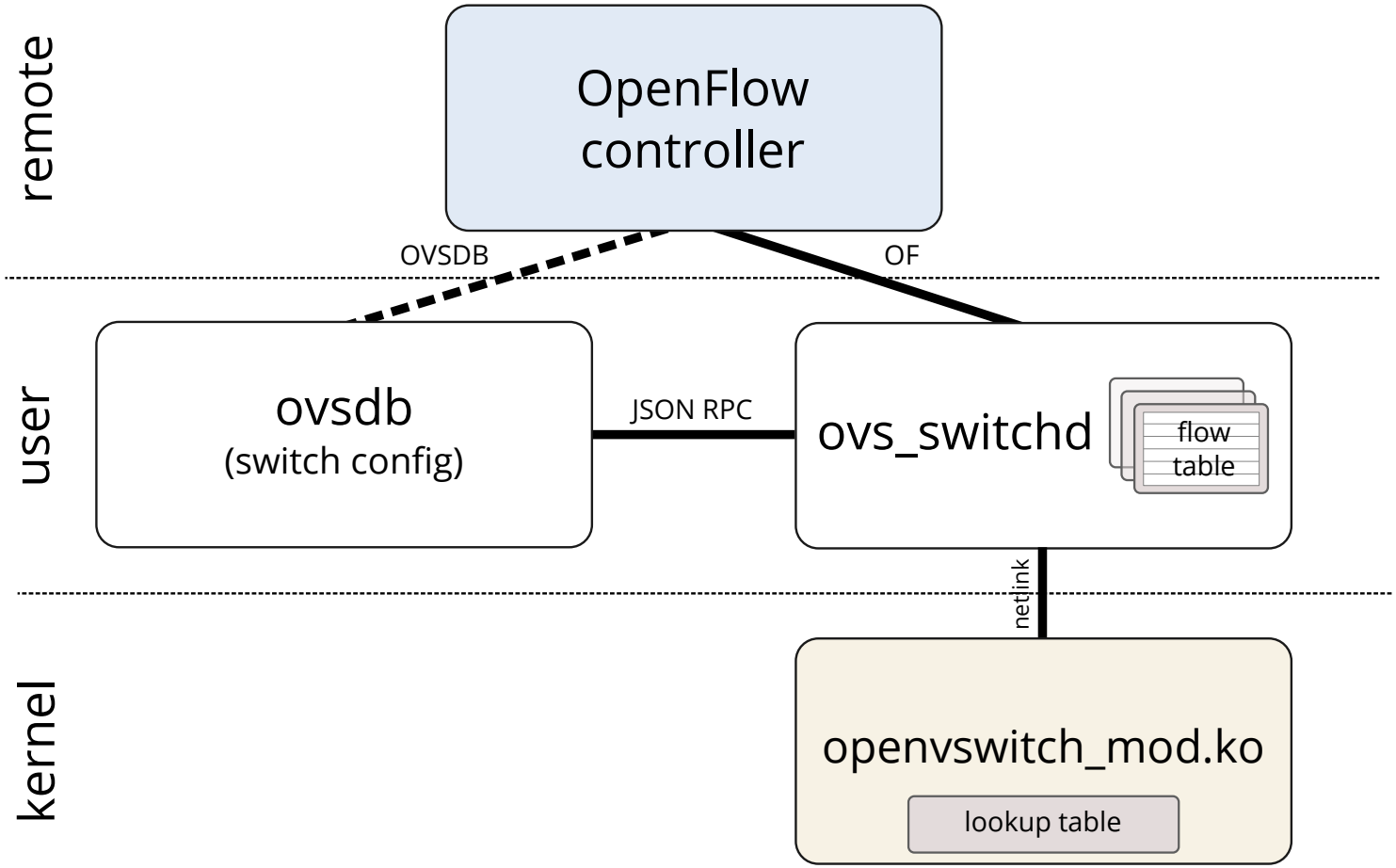
OpenVSwitch

OpenVSwitch ist ein Open Source virtueller Switch

- “drop-in” Alternative zur Linux Bridge
- Unterstützt eine Reihe von Management- und Netzwerk Technologien (NetFlow, sFlow, **OpenFlow**, SPAN, GRE, VxLAN, LACP, . . .)

`www.openvswitch.org`

OpenVSwitch: Architektur



Kernelmodul seit V3.3 im Mainline Linux Kernel

`mininet.org`

- Mininet erstellt virtuelle Topologien auf einer Maschine (Namespaces, cgroups)
- virtuelle Hosts werden mit OpenVSwitches verbunden
- Switches können mit beliebigen externen OpenFlow Controllern gesteuert werden
- Auf den virtuellen Hosts kann beliebiger Code ausgeführt werden (auch interaktiv)
- Scripting mit Python
- Anbindung an reales Netzwerk möglich

SDN: Herausforderungen

Es existieren noch einige Probleme und offene Forschungsfragen

- Wie kann die Konsistenz der globalen Netzwerkes Sicht in SDN garantiert werden
- Skalierbarkeit von SDN (verteilten) SDN Controllern
- kein einheitliches SDN Northbound-Interface

- Interoperabilität zwischen OpenFlow Herstellerimplementierungen
- Konvergenz der Networking und Software-Engineering Communities

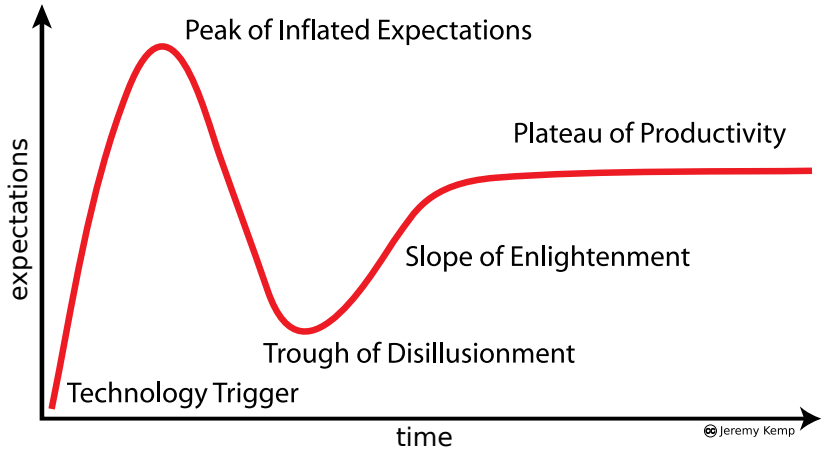
SDN: Herausforderungen

Es existieren noch einige Probleme und offene Forschungsfragen

- Wie kann die Konsistenz der globalen Netzwerkes Sicht in SDN garantiert werden
- Skalierbarkeit von SDN (verteilten) SDN Controllern
- kein einheitliches SDN Northbound-Interface

- Interoperabilität zwischen OpenFlow Herstellerimplementierungen
- Konvergenz der Networking und Software-Engineering Communities

Wir haben den Höhepunkt des SDN Hypes bereits überschritten

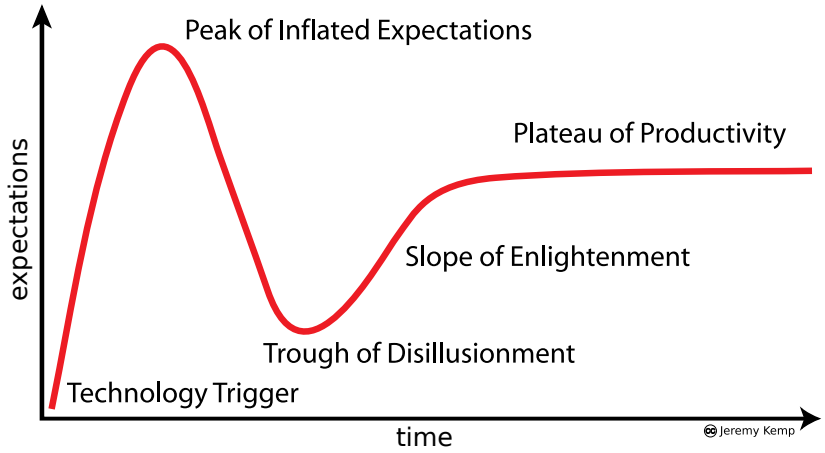


Best case: Neue Herangehensweisen, um Netzwerke effizient zu Betreiben, Probleme Strukturiert zu Lösen und neue Dienste schnell in Betrieb zu nehmen

Worst case: SDN Insellösungen, OpenFlow für Programmierbarkeit im Netzwerk

Institut für Kommunikations-Technik

Wir haben den Höhepunkt des SDN Hypes bereits überschritten



Best case: Neue Herangehensweisen, um Netzwerke effizient zu Betreiben, Probleme Strukturiert zu Lösen und neue Dienste schnell in Betrieb zu nehmen

Worst case: SDN Insellösungen, OpenFlow für Programmierbarkeit im Netzwerk

Vielen Dank für Ihre Aufmerksamkeit!

Fragen?