

Software Defined Networking

Ph. Aeschlimann

ZHAW

Gastvorlesung MAS-I5 - SDN - OpenFlow

- 1 Nur kurz, versprochen...
- 2 Software Defined Networking
- 3 OpenFlow - Eine Implementation
- 4 SDN im ICCLab

- Philipp Aeschlimann [aapp@zhaw.ch, <https://wiki.teukros.ch>]
- Studium zum Maschinenbauingenieur -> Informatik? -> Studium zum Informatikingenieur
- Aktuell: WiMa am InIT im ICCLab www.cloudcomp.ch
- Themen: SDN - OpenFlow - OpenStack - Middleware - Datacenters - interne IT Infrastruktur

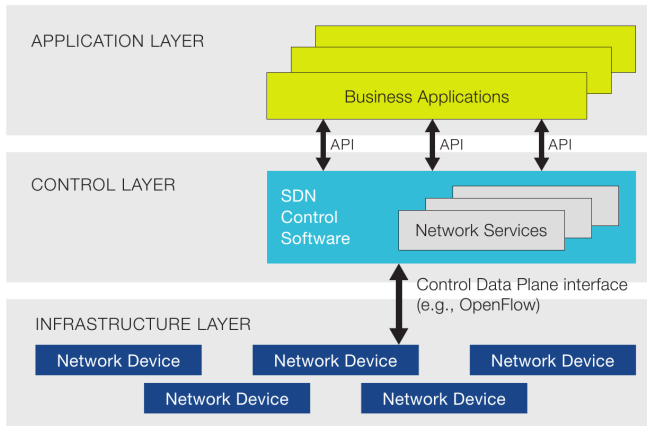


- Aktuelle Netzwerkarchitekturen sind i.d.R. hierarchisch organisiert
 - klassische Client-Server Struktur
 - eher ungeeignet für mobile Endgeräte, Virtualisierung von Server und dynamische Strukturen
- Netzwerkschemas stehen vor neuen Anforderungen
 - Umgang mit einem hybriden CloudStack
 - Netzwerk für Benutzer auf einer "on-demand" basis
 - Dienste werden angeboten in einer Cloud
 - grosse Datenmengen und Zuverlässigkeit

- Komplexität
 - Netzwerkgeräte in einer Topologie neu platzieren
 - was muss getan werden?
- Konsistente Netzwerkstrategie
 - ändern einer Policy oder anpassen von ACL
- Ein Netzwerk skalieren
 - wie umgehen mit mehreren Klienten (Multi-tenancy)
 - wie schnell kann ein Netzwerk wachsen?
- Abhängigkeit von Herstellern
 - oftmals nicht gewährleistet

- Anwendungsfall Hochschule
 - teilen des Netzwerks in ein experimentelles und produktives Netzwerk
 - Netzwerk verfügbar mache für mehrere Hochschulen (abhängig von der Applikation)
- DataCenter
 - hochskalierbare, virtuelle Netzwerke
 - automatisierte Migration von virtuellen Maschinen und dem zugehörigen Netzwerk
- Cloud
 - ein wichtiger Bestandteil ist die Elastizität
 - Netzwerk als Dienst anbieten
- Gameserver
 - betrachtung folgt später...

Jetzt aber, SDN - ein Paradigma



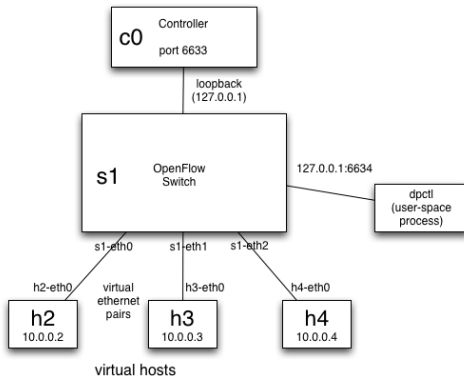
- Gameserver Video

- OpenFlow ist "die Implementation" des Paradigmas SDN
 - OpenFlow selbst ist "nur" das Protokoll
 - entwickelt an der Stanford University
<https://openflow.stanford.edu/dashboard.action>
- für eine konkrete Implementation braucht es mehr als ein Protokoll
 - "OpenFlow ready hardware"
 - einen zentralen "controller"
 - Netzwerklogik als Software im controller
- the freedom of choice, we all love it!
 - welche Geräte sollen gekauft werden?
 - welche Implementation soll verwendet werden?
 - Protokoll und zentraler Controller
 - wer entwickelt die Netzwerklogik?

- Controller haben individuelle Stärken
- Floodlight
 - in Java geschrieben, bietet ein Web-GUI und verfügt über eine Vielzahl von "Plugins"
- NOX (POX)
 - der OpenFlow Referenz-Controller geschrieben in C, zeigt bisher die beste Performance
 - POX erlaubt die Implementierung von Python Code in NOX (mehr Plugins verfügbar)
- Trema
 - ein Controller geschrieben in Ruby mit nativen Rubyerweiterungen in C
- Beacon
 - ein "fork" von Floodlight mit verbesserter/anderer Multithreading Implementation
- NodeFlow
 - vielversprechende Implementation mit node.js und JavaScript (Google v8 burns like hell)

- Verschiedene Entwicklungswerkzeuge vorhanden
- DPCTL
 - ein Werkzeug um Datenpfade zu administrieren
 - kann auch Einträge in der "flow-table" editieren
 - nicht als Ersatz für einen Controller gedacht
- mininet
 - eine Software, die virtuelle Netzwerke erstellen kann, speziell für SDN
 - mit mininet kann man problemlos grosse virtuelle Infrastrukturen schaffen
- iperf
 - ein Tool um den "throughput" von Netzwerkgeräten zu messen
- cbench
 - Benchmarkingtool zum generieren von Verkehr

Durchgängiger Anwendungsfall



- 1 h2 möchte prüfen, ob h3 erreichbar ist: h2 ping -c1 h3
- 2 das Packet erreicht s1 aber s1 weiss nicht, was mit diesem Packet gemacht werden soll
 - 1 s1 sendet das Packet an c0 und c0 kann eine intelligente Entscheidung treffen
- 3 c0 kennt jetzt den Port von h2 aber nicht den Port von h3
 - 1 s1 wird einen "flood" durchführen
- 4 h3 erhält das Packet und wird natürlich eine Antwort senden
- 5 s1 erhält das Packet und wird nochmals c0 fragen, was mit dem Packet zu tun ist
 - 1 da c0 jetzt weiss, an welchem Port h2 ist, muss kein flood gemacht werden
 - 2 bei dieser Gelegenheit, sieht c0 auch an welchem Port h3 angeschlossen ist

Die FlowTable

- beim vorher beschriebenen Ablauf, muss der Controller jedes mal gefragt werden
 - nicht sinnvoll, da wir Netzwerkgeräte mit multicore CPU's haben
- besser wäre, der Switch könnte selbstständig entscheiden
 - in OpenFlow führt jeder Switch eine FlowTable
- wer befüllt die FlowTable?
 - der Controller anhand der programmierten Logik

OpenFlow-enabled Network Device

Flow Table comparable to an instruction set

MAC src	MAC dst	IP Src	IP Dst	TCP dport	...	Action	Count
*	10:20:..	*	*	*	*	port 1	250
*	*	*	5.6.7.8	*	*	port 2	300
*	*	*	*	25	*	drop	892
*	*	*	192.*	*	*	local	120
*	*	*	*	*	*	controller	11

- starten des Controllers: `def start_switch (event):`
- Konstruktor des Controllers: `def __init__ (self, connection):`
 - Erstellen des MAC Adressen Speichers
- der Event Packet in: `def _handle_PacketIn (self, event):`
 - entscheiden, welche Logik des Controllers ausgeführt werden soll
- Switch Programmlogik: `def act_like_lswitch (self, packet, packet_in):`
 - MAC Adresse Speichern
 - opt. FlowTable Eintrag installieren
 - Packet ausliefern, direkt an Empfänger oder via "flood"

Die "matching" Klassen

- ein zentraler Bestandteil des Controllers sind die "matching" Funktionen
 - Siehe auch Deep Packet Inspection
- ein Netzwerk Controller abstrahiert mit speziellen "matching" Funktionen die Betrachtung der Packet's

```
#fm.match = of.ofp_match.from_packet(packet)
fm.match.in_port = packet_in.in_port fm.match.dl_dst = dstaddr
fm.actions.append(of.ofp_action_output(port =
self.macStore[dstaddr.toStr()]))
#fm.idle_timeout = 10
#fm.hard_timeout = 30 log.debug("Installing FlowTable entry")
self.connection.send(fm)
```


OpenFlow - Das Ende der Fahnenstange?

- Google selbst hat das eigene WAN auf OpenFlow umgestellt
 - Controller unbekannt resp. modifiziert
- OpenFlow und die Open Network Foundation (ONF) sind breit abgestützt
- OpenFlow befindet sich unter stetiger Entwicklung obschon die Spezifikationen klar sind
- offene Fragen
 - forwarding in gemischten Netzwerke
 - API gegen die Applikationen
 - QoS bisher eher stiefmütterlich behandelt
- es ist fraglich, ob grosse Player im Netzwerkbusiness OpenFlow unterstützen werden

- derzeit zwei konkrete Projekte im SDN Bereich mit OpenFlow
 - Future Middleware KIARA (EU-Projekt, FI-PPP, FI-WARE)
 - SDN in OpenStack - Netzwerk als Dienst dem Klienten anbieten
- enge Zusammenarbeit mit SWITCH
 - SWITCH selbst ist Partner in einem EU Projekt und baut ein SDN WAN
 - zentrale Rolle der Schweiz als Drehscheibe
- weitere Informationen über das ICCLab unter www.cloudcomp.ch



- <https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf>
- <http://www.openflow.org/documents/openflow-wp-latest.pdf>
- http://www.openflow.org/wk/index.php/Main_Page
- <https://www.opennetworking.org/>
- <https://openflow.stanford.edu/display/ONL/POX+Wiki>