Programmierbare Netzwerk Dataplanes

afics

19. Februar 2020

Einleitung

- Routing & Switching Primer
- Software Defined Networking (SDN)
- P4

- Source & Destination MAC Addresse als Zustellinformation
- Broadcast-, Unknown Unicast, Multicast (BUM) Traffic
- Unicast Traffic
- (Ether)Type (IPv$\{4,6\}$, ARP/NDP, STP, ...)
- Identifiziert Netzwerkkarten (NICs)

- Source & Destination IP Addresse als Zustellinformation
- Identifiziert Hosts im Netzwerk
- Time-to-Live
- Checksum
- Protocol (ICMP, UDP/TCP/SCTP, IPIP/GRE, ESP/AH, OSPF, VRRP/CARP)
- **Routing** Longest Prefix Match (LPM)

Klassisches Netzwerk

Hardware

Vereint in einem Gerät:

- **Controlplane** Entscheidet, was mit Paketen passieren soll
- **Dataplane** Führt die Verarbeitung der Pakete nach den Vorgaben der Controlplane aus

Klassisches Netzwerk

Controlplane

- Verteiltes System, die einzelnen Controlplanes arbeiten über Protokolle zusammen, z.B.:
  - Spanning Tree (STP)
  - Open Shortest Path First (OSPF)
  - Border Gateway Protocol (BGP)
  - Multi-Protocol Label Switching mit BGP und Label Distribution Protocol (LDP)
  - Virtual Extensible LAN (VXLAN)
  - uvm.
- Die Geräte werden einzeln konfiguriert

Software Defined Networking

Einführung

- Trennen der Controlplane von den Netzwerkgeräten und zentralisieren auf einem (redundanten) SDN Controller
- Der SDN Controller provisioniert die Dataplanes der einzelnen Netzwerkgeräte
- Protokoll zur Kommunikation zwischen Controller und Dataplane meist **OpenFlow**

Software Defined Networking

Allgemeines

- Jeder Hersteller versteht SDN anders
- Gibt es konzeptionell in Form von WLAN Controllern schon seit langer Zeit
- SDN kann ein Overlay erfordern oder auch nicht
- Für den User lässt ein SDN Controller ein Netzwerk wie einen einzigen Switch erscheinen
- APIs

Software Defined Networking

OpenFlow

- Protokoll
- Controller & OpenFlow Switches
- Network Processing Unit (NPU) Interna/Pipeline: Tables, Actions, ...

Software Defined Networking

OpenFlow Protokoll

- Regelt Kommunikation zwischen Controller und Switches
- TCP basiert, Port 6653, wahlweise TLS verschlüsselt
- Kann Table Einträge hinzufügen, ändern und löschen
- Drei Message Typen
  - **Symmetric** Nachrichten für Verbindungsaufbau, Errors und Experimentelle Features
  - **Controller to Switch** Ändern von Flow Table Entries, Features & Statistiken abfragen, ...
  - **Asynchronous** Switch zum Controller, Unbekannte Pakete, Änderungen am Flow Table, Port Status, ...

- **Ports** Physical, Logical, Reserved
- **Tables**
  - **Flow** Match Selector mappt Flows auf „Instructions" (Aktionen)
  - **Group** Gruppiert Ports
  - **Meter** QoS, Shaper, ...
- Table Miss

- Neuer Blickwinkel
- Weniger Komplexität?
- Bessere Skalierung?
- Aus Administrator-Sicht: Mehr Flexibilität (z.B. MLAG)

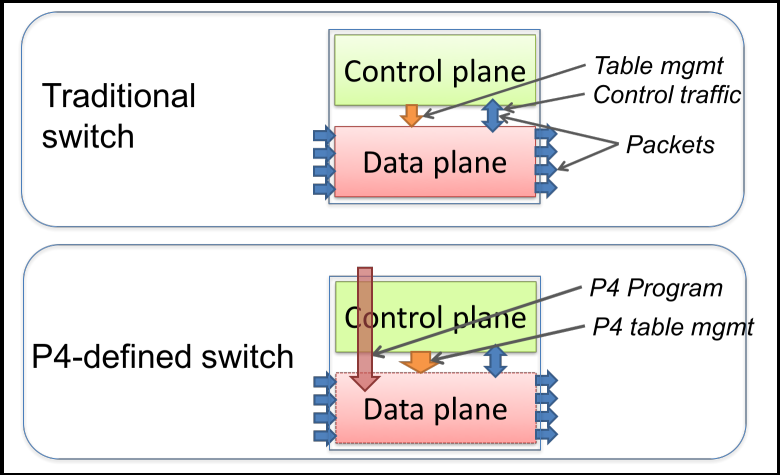
Software Defined Networking

Evolution

- Mehr Funktionalität $\Rightarrow$ mehr Tables (von 12 zu aktuell 40)
- Langsame Iteration
- Protokoll Design: unflexibel

- Okösystem
  - Sprache
  - Compiler
  - Targets
  - Runtime
- Flexibel
- Einfaches Prototyping
- Traditionelles Netzwerk kann mit P4 implementiert werden
- OpenFlow kann mit P4 implementiert werden, aber nicht umgekehrt
- Aktuelle Version: $p4_{16}$

Unterschied zu SDN: Static vs. User-defined Dataplane



P4

Beispiel Projekte

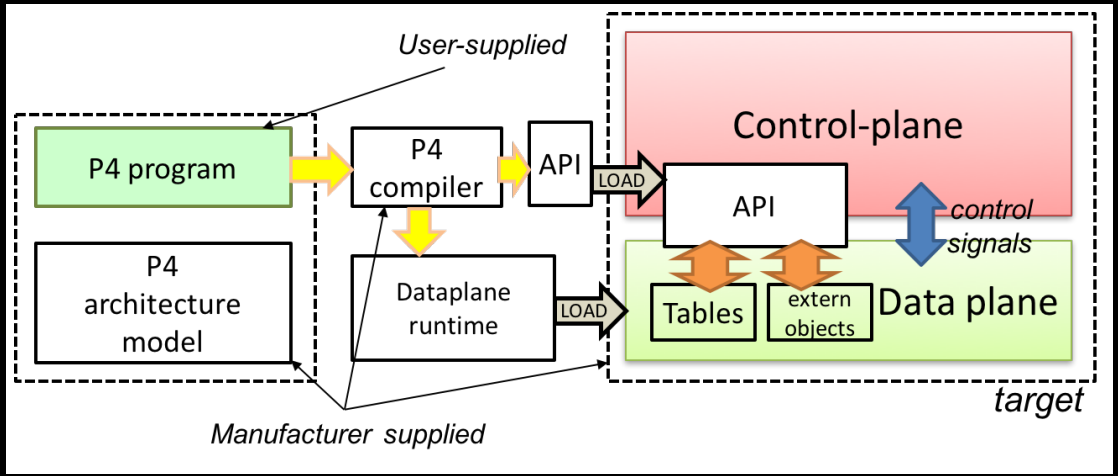
- calc.exe :)
- Explicit Congestion Notification
- Low Latency Congestion Control
- In-band Network Telemetry
- In-Network caching and coordination
- Aggregation for MapReduce Applications
- Layer 4 Load Balancer - SilkRoad, S.H.E.L.L., ...
- Traditioneller Switch - switch.p4
- SDN Switch - fabric.p4

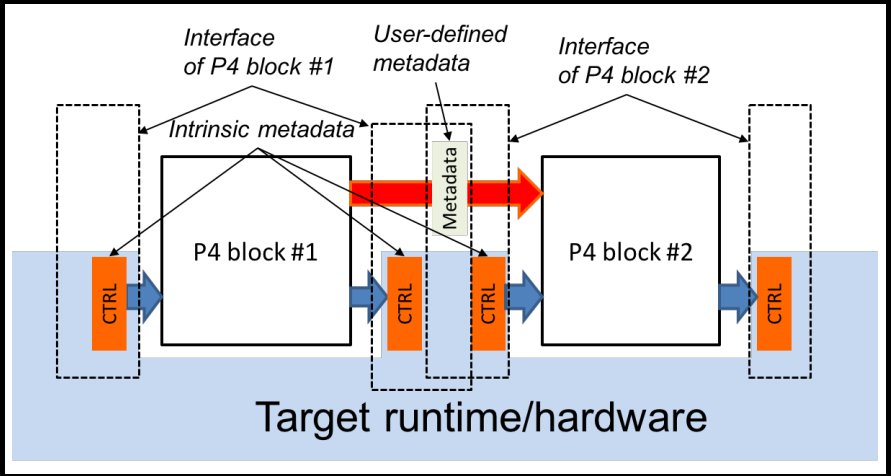
P4

Welche Platformen werden programmiert?

- **PISA:** Flexible Match+Action ASICs
  z.B.: Intel Flexpipe, Cisco Doppler, Cavium (Xpliant), Barefoot Tofino, ...
- **NPU:** z.B.: EZchip, Netronome, ...
- **CPU:** Open Vswitch, eBPF, DPDK, VPP...
- **FPGA:** Xilinx, Altera, ...

Bottom Up vs. Top Down


User-supplied
P4 program
P4 compiler
API
LOAD
Control-plane
API
control signals
P4 architecture model
Dataplane runtime
LOAD
Tables
extern objects
Data plane
target
Manufacturer supplied


Interface
of P4 block #1
User-defined
metadata
Interface
of P4 block #2
Intrinsic metadata
Metadata
P4 block #1
P4 block #2
CTRL
CTRL
CTRL
CTRL
Target runtime/hardware

P4
Primitiven

- Header Typen
- Parsers
- Tables
- Actions
- Match-Action Blöcke
- Kontrollfluss
- Metadaten (Pro Paket)

- Match-Action Blöcke:
  - Lookup Keys erzeugen aus Paket Feldern oder Metadaten
  - Lookup Erzeugen $\Rightarrow$ eine Action wird gewählt
  - Die Action wird ausgeführt
- Metadaten (Pro Paket)
  - User-defined: Benutzer definierte Metadaten (z.B. Routing Table ID)
  - Intrinsic: Metadaten von der Architektur (z.B. Ingress Port)

- Kommunikation zwischen Control und Dataplane
- gRPC
- Der Compiler erzeugt:
  - ein Kompilat mit der Forwarding Logik, welche in der Dataplane ausgeführt wird
  - ein API Definition zum Managen der Objekte in der Dataplane

- V1Model
- Protocol Independent Switch Architecture (PISA) $\Leftarrow$ (nur $p4_{14}$)
- Portable Switch Architecture (PSA)

Target für Tests und Entwicklung $\Rightarrow$ `p4-bmv2 / p4/behavioural-model`

Batteries included?

Architekturspezifisch, V1Model:

- Parser
- VerifyChecksum
- Ingress
- Egress
- ComputeChecksum
- Deparser

- Für Setup einer Testumgebung siehe: `https://github.com/jafingerhut/p4-guide/blob/master/bin/README-install-troubleshooting.md`
- scapy

Demo!

Danke für die
Aufmerksamkeit!