Hochschule Fulda
University of Applied Sciences

Advanced Computer Networks

Lab 5: Programmable Data Plane & P4 Introduction

1 Programmable Data Plane und P4

- Was verstehen Sie unter Software-Defined Networking?
- Was sind die Prinzipien und Konzepte von P4? Wie hängen SDN und P4 zusammen?
- Was sind die Unterschiede von OpenFlow und P4?

Relevante Informationen zu P4 finden Sie auch auf den am Ende des Kapitels genannten Links zu Tutorials und verwandten Lehrveranstaltungen.

Die folgenden Aufgabenstellungen/Inhalte orientieren sich an den P4 Tutorials und am p4-learning Repository der NSG-ETHZ. Für die Bearbeitung und die Projekte innerhalb Ihrer Gruppe erhalten Sie wie schon zuvor beim Mininet/SDN-Praktikum Zugriff auf eine vorbereitete virtuelle Maschine (Firecracker microVM). Ihre VMs erreichen Sie nun über:

`https://prona.informatik.hs-fulda.de/guacamole/`

Melden Sie sich mit ihrem Gruppen-Account an der VM mit der Endung P4 an (User: acn-groupXY, Passwort: netlab, VM: GruppeXY_P4).

Greifen Sie über den "learn-sdn-hub"Link auf dem Desktop auf die web-basierte P4-Umgebung für das Praktikum zu und loggen Sie sich mit Username: p4 und Password: p4 ein.

2 P4 Beispiel (1): Repeater

Deployen Sie das Example1 und starten Sie das Assignment. Lassen Sie kurz das Repeater-Beispiel aus den Folien Revue passieren, indem Sie den P4 Code des Assignments anschauen und dann unter Terminals im BASH Terminal einen ping von h1 zu h2 starten:

```
mininet> h1 ping h2
```

- An welcher Stelle im Programm wird die Logik des Repeaters umgesetzt?

Überprüfen Sie Ihre Vermutung, indem Sie den Code verändern und anschließend auf Deploy klicken, um die Änderungen in der Umgebung zu speichern. Staren Sie danach mininet neu mit:

```
mininet> exit
p4@gruppeXY:~/p4-boilerplate/Example1-Repeater$ make clean
p4@gruppeXY:~/p4-boilerplate/Example1-Repeater$ make
```

Hierbei wird ein Makefile in Anlehnung an die P4 tutorials verwendet, das Sie ebenfalls in einem Tab des Editors geöffnet haben. Das Makefile verwendet die Topologie in topology.json für die Anlegung der Hosts, Switches und Links dazwischen in mininet. Die Initialisierung des Switches erfolgt in s1-runtime.json und ist im Repeater Beispiel weitgehend leer. Sie können die Dateien auch im Terminal sehen, z.B. wenn Sie im BASH2 Terminal ls verwenden:

```
p4@gruppeXY:~/p4-boilerplate/Example1-Repeater$ ls
Makefile  Makefile~  build  logs  pcaps  pod-topo  prona-repeater.p4
p4@gruppeXY:~/p4-boilerplate/Example1-Repeater$ ls pod-topo
s1-runtime.json  topology.json
```

3 P4 Beispiel (2): Minimalistischer Layer 2 Switch

Klicken Sie in der Menüleiste auf Assignments, deployen Sie das Example2 und starten Sie das Assignment. Lassen Sie kurz das Minimalistic-Switch-Beispiel aus den Folien Revue passieren, indem Sie den P4 Code des Assignments anschauen und dann unter Terminals im BASH Terminal einen ping von h1 zu h2 starten:

```
mininet> h1 ping h2
```

- Warum funktioniert der Ping nicht?
- Was fehlt noch, obwohl der P4 Code für den Switch vollständig ist?
- Wir beheben das Problem in Aufgabe 1 und 2.

3.1 Aufgabe 1: Hinzufügen von Tabelleneinträgen für die MAC-Adressen der Hosts

Fügen Sie Tabellen-Einträge für die MAC-Adressen der Hosts ein. Tabelleneinträge müssen zur Laufzeit (P4Runtime) z.B. durch die Control Plane (per CLI, Python API, gRPC) hinzugefügt werden. Sie können hierfür die simple_switch_CLI im BASH2 Terminal verwenden.

```
p4@gruppeXY:./Example2-MinimalisticSwitch$ simple_switch_CLI
RuntimeCmd: ?
RuntimeCmd: help table_add
```

Beenden können Sie die simple_switch_CLI mit CTRL+C. Fügen Sie in der Tabelle MyIngress.dstMacAddr für jeden Host einen Eintrag hinzu, und geben Sie eine Action (forward) und Parameter (egress port) an. Sie können mit TAB vervollständigen. Die MAC-Adressen der Hosts sehen Sie im Editor unter topology.json. Alternativ können Sie diese auch in mininet auf den Hosts nachschauen (z.B. "h1 ifconfig eth0"). Die erforderlichen Befehle stehen auch auf den Folien zur Lehrveranstaltung.

```
RuntimeCmd: table_add ...
```

Testen Sie danach erneut den Ping.

3.2 Aufgabe 2: Hinzufügen einer Multicast-Gruppe

Fügen Sie auf dem Switch eines Multicast-Gruppe hinzu, um auch Broad- und Multicasts zu unterstützen. Erzeugen Sie die Multicast-Gruppe 1 (std_meta.mcast_grp = 1). Standardmäßig weiß der Switch nicht welche Ports zu welcher Broadcast Domain (z.B., VLAN) gehören sollen. Die erforderlichen Befehle stehen auch auf den Folien zur Lehrveranstaltung.

```
RuntimeCmd: help mc_mgrp_create
```

Erzeugen Sie einen Node 0 und fügen Sie alle Ports (1, 2) zu ihm hinzu. Assoziieren Sie anschließend den Node mit der zuvor angelegten Multicast-Gruppe 1.

```
RuntimeCmd: help mc_node_create
RuntimeCmd: help mc_node_associate
```

Nachdem Sie die Multicast-Gruppe den Node und die Assoziation angelegt haben, können Sie alle Einträge aus der Tabelle dstMacAddr entfernen, und der Ping sollte nun trotzdem noch funktionieren, da unbekannte MAC-Adressen per Default als broadcast gesendet werden (default_action der Tabelle dstMacAddr).

3.3 Diskussion und nächste Schritte

Der Ping von h1 zu h2 funktioniert und wir haben einen static Layer 2 switch realisiert. Neben Flooding (per Broadcast) kann dieser durch statische Einträge für die MAC-Addressen der Hosts auch Filtering für das Forwarding in der Data Plane umsetzen.

Wie können wir ein dynamisches Lernen und Füllen der MAC-Adresstabelle erreichen und so die Funktion eines typischen realen Layer 2 Switches (flood & filter) implementieren?

4 P4 Beispiel (3): Learning Layer 2 Switch

Klicken Sie in der Menüleiste auf Assignments, deployen Sie das Example3 und starten Sie das Assignment. Lassen Sie kurz das Learning-Switch-Beispiel aus den Folien Revue

passieren, indem Sie den P4 Code des Assignments anschauen und dann unter Terminals im BASH Terminal einen ping von h1 zu h2 starten:

```
mininet> h1 ping h2
```

- Warum funktioniert der Ping nicht?
- Was fehlt noch, obwohl der P4 Code für den Switch vollständig ist?
- Wir beheben das Problem in Aufgabe 1 und 2.

4.1 Aufgabe 1: Starten des Controllers

Starten Sie den Controller im BASH2 Terminal.

```
p4@gruppeXY:./Example3-LearningSwitch$ \
 sudo python learning_switch_controller_app.py s1
```

Sie finden den Code des Controllers im Editor Tab learning_switch_controller_app.py. Der Parameter s1 gibt den Namen des Switches an den der Controller verwalten soll. Beobachten Sie im Terminal BASH2 die Aktionen, die der Controller auf dem Switch s1 durchführt.

- Wie hängt dies mit Ihrer Verwendung von simple_switch_CLI beim Minimalistic-Switch zusammen?
- Welche Vorteile entstehen hier im Vergleich zur Verwendung von simple_switch_CLI?

4.2 Aufgabe 2: Neustart der Umgebung und Betrachtung der Wirkung von Flooding & Filtering

Stoppen Sie mininet im BASH Terminal und starten Sie die Umgebung neu:

```
mininet> exit
p4@gruppeXY:./Example3-LearningSwitch$ sudo p4run
```

Starten Sie anschließend den Controller im BASH2 Terminal erneut:

```
p4@gruppeXY:./Example3-LearningSwitch$ \
sudo python learning_switch_controller_app.py s1
```

Starten Sie im BASH3 Terminal tcpdump (simple Alternative zu Wireshark auf der CLI) auf h3. Der Befehl mx ist eine praktische Erweiterung aus der p4-learning Umgebung der ETHZ. Damit können Sie direkt im Terminal Befehle auf den Hosts im laufenden mininet ausführen. Die Umgebung bietet auch noch mxexec zum Ausführen von Befehlen in mininet.

```
p4@gruppeXY:./Example3-LearningSwitch$ mx h3 tcpdump
```

Starten Sie in mininet im BASH Terminal einen Ping von h1 zu h2.

- Welche Pakete vom Ping zwischen h1 und h2 sehen Sie auch an h3 im tcpdump im BASH3 Terminal? Warum?

4.3 Diskussion und nächste Schritte

Der Switch leitet Layer 2 Frames basierend auf ihrer Destination-MAC-Adresse weiter. Die Lösung implementiert die typische "flood & filter"Funktion eines realen Layer 2 Switches. Broadcasts und Frames an unbekannte MAC-Adressen (unknown unicast) werden an alle Ports geflutet, außer dem Port über den das Frame am Switch empfangen wurde (einfache Vermeidung von Loops). Sie können die table_add Zeilen im Controller auskommentieren, den Controller neu starten und sehen, dass dann alle Frames geflutet werden. Das Gleiche passiert, wenn die Tabelle voll ist, und deren default_action greift. Genau das Gleiche passiert auch bei realen Switches. Frames an gelernte Ziel-MAC-Adressen werden vom Switch nur an die Ports weitergeleitet auf denen diese MAC-Adressen zuvor als Quelle gesehen wurde.

Die Implementierung ist noch immer stark vereinfacht. Z.B. wird in der Lösung kein Aging bzw. Vergessen von MAC-Adressen unterstützt. Außerdem keine VLANs, kein STP usw. Trotzdem haben wir die essentiellen Bestandteile der Implementierung der Data Plane und Control Plane eines Layer 2 Switches explorativ kennengelernt. Basierend darauf können Sie fortgeschrittenere Network Elements und deren Funktionen implementieren. Sie können dafür z.B. die in Example1 und Example2 verwendete Umgebung der P4 Tutorials verwenden. Oder die p4-learning Umgebung, die in Example3 verwendet wurde. Eine umfassende integrierte P4-Entwicklungsumgebung bietet auch das p4environment, das im Example-p4env verwendet wird.

5 Direkte Verwendung der P4 Tools in der VM

Wir haben die web-basierte Umgebung learn-sdn-hub verwendet, um uns der Entwicklung von P4 Code anzunähern. Die Umgebung kapselt und abstrahiert die Entwicklung von P4 Code und die Verbindung per SSH zu einem Target Host, der über die erforderliche P4 Toolchain verfügt, um den Einstieg zu vereinfachen. Sie können für Ihre weiteren Experimente auch direkt, ohne die Web-Oberfläche, arbeiten. Auf dem Desktop finden Sie einen Link zu einem Terminal als Benutzer p4 (Passwort: p4). In /p4-boilerplate (bzw. /home/p4-boilerplate) wurden die o.g. Example1, Example2 und Example3 ausgeführt. In /tutorials finden Sie die kompletten Beispiele des offiziellen P4 Tutorials von p4lang. In /p4-learning ist die komplette p4-learning Umgebung der NSG der ETHZ zu finden. In /p4environment finden Sie das p4environment der Hochschule Fulda. Unter /p4-guide finden Sie den P4 Guide. Weiterführende Links hierzu finden Sie im nächsten Abschnitt des Übungsblatts.

6 Weiterführende Informationen

- Verwendete Code-Beispiele (ProNA P4 boilerplate):

  `https://github.com/prona-p4-learning-platform/p4-boilerplate`

- Verwendete Plattform (ProNA learn-sdn-hub):

  `https://github.com/prona-p4-learning-platform/learn-sdn-hub`

- P4 Guide:

  `https://github.com/jafingerhut/p4-guide`

- P4 Tutorials:

  `https://github.com/p4lang/tutorials`

- p4-learning Umgebung der NSG der ETHZ:

  `https://github.com/nsg-ethz/p4-learning`

- p4environment des NetLab der Hochschule Fulda:

  `https://gitlab.cs.hs-fulda.de/flow-routing/cnsm2020/p4environment`

7 Eigene VM mit P4 Toolchain

Sie können auch eine eigene Linux VM installieren, die alle für die Entwicklung von P4 erforderlichen Tools (p4c, bmv2, pi, p4runtime) beinhaltet. Hierfür können Sie z.B. die Install-Skripte aus dem P4-Guide zusammen mit Ubuntu 18.04 verwenden: `https://github.com/jafingerhut/p4-guide/blob/master/bin`.

Wir empfehlen aktuell die Verwendung der v2: `https://github.com/jafingerhut/p4-guide/blob/master/bin/install-p4dev-v2.sh`

Alternativ finden Sie im Web zahlreiche Anleitungen und Tutorials sowie VMs, wie z.B. bei den P4 Tutorials `https://github.com/p4lang/tutorials` (siehe Abschnitt "Obtaining required software") oder p4-learning `https://github.com/nsg-ethz/p4-learning` (siehe Abschnitt "Required Software").