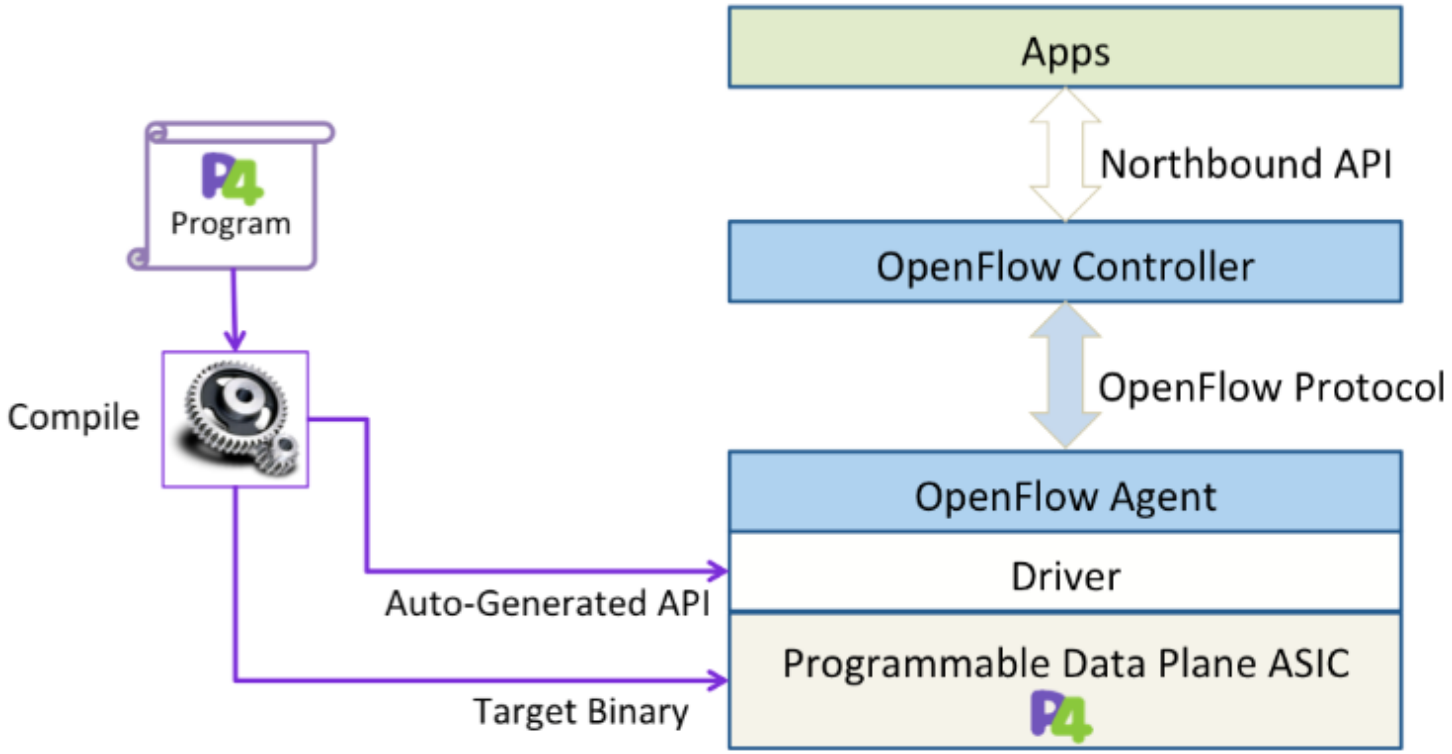
Warsztat 5

mgr inż. Jan Palimąka

Wprowadzenie



P4Runtime

- Protokół między płaszczyzną danych a warstwą sterowania
  - Serwer: switch lub dedykowane oprogramowanie, np. Stratum
  - Klient: dowolna aplikacja płaszczyzny sterowania
- Dedykowany do programów napisanych w P4
- Wspiera praktycznie wszystkie funkcjonalności języka P4
- Niezależny od switcha
- Standardowy port: 9559/tcp
- Brak funkcji związanych z zarządzaniem
  - Nie można np. włączyć/wyłączyć portu, ustawić szybkości bitowej, kodu nadmiarowego, ...
  - Thrift, gNMI, OpenConfig YANG model

Zadanie

- Wykorzystanie dowolnego programu P4 z poprzednich warsztatów
- Uczenie adresów MAC (MAC learning)
  - Powiadomienie płaszczyzny sterowania o nieznanym adresie MAC
  - Podmiana adresu docelowego na podstawie adresu IP
  - Prosta aplikacja w języku Python do dodawania nowych wpisów (jako aplikacji płaszczyzny sterowania)
- Konfiguracja sieci z użyciem P4Runtime
  - Uruchomienie i konfiguracja switcha
  - Dodawanie wpisów do tabel
  - Dowód działania (np. działające polecenie ping)

Sprawozdanie

- Zamieszczony kod źródłowy
- Wpisywane polecenia (wraz z krótkim opisem), które:
  - Nie były wymienione w instrukcji
  - Należało "odkryć" wg instrukcji
- Termin
  - Grupa poniedziałkowa: 29 kwietnia (do końca dnia)
  - Grupa środowa: 24 kwietnia (do końca dnia)

digest - powiadomienie płaszczyzny sterowania

```
void digest<T>(in bit<32> receiver, in T data);
```

- Funkcja - wystarczy wywołanie
- `T` – typ wysyłanych danych, np. nazwa struktury (będzie użyta jako nazwa instancji obiektu typu `digest`)
- `receiver` – ignorowane przez BMv2
- `data` - wysyłane dane, np.:
  - `{f1, f2, f3}`
  - `meta.mac_digest`
- Można wywoływać tylko w bloku ingress

CPU_PORT - wysłanie pakietu do płaszczyzny sterowania

- Pozwala na bezpośrednie wysłanie/odebranie pakietu do/z płaszczyzny sterowania
- Stosowane, gdy pakietu nie można obsłużyć w płaszczyźnie przekazu danych
  - Protokoły routingu, kontrolne, ...
- W płaszczyźnie przekazu danych widoczny jako zwykły port
- simple_switch nie wspiera tej funkcji
- simple_switch_grpc wymaga opcji --cpu_port
  - 9 bitów (dla v1model)
  - 0 nie jest poprawną wartością
  - 511 zarezerwowane dla DROP_PORT

Instalacja zależności

- p4runtime-shell - klient P4Runtime
  https://github.com/p4lang/p4runtime-shell

```
pip3 install --upgrade \
git+https://github.com/p4lang/p4runtime-shell.git#egg=httpie
```

Modyfikacja pliku p4_mininet.py

```
diff --git a/mininet/p4_mininet.py b/mininet/p4_mininet.py
index 0cb0c78..33c3c78 100755
--- a/mininet/p4_mininet.py
+++ b/mininet/p4_mininet.py
@@ -124,7 +124,7 @@ class P4Switch(Switch):
             args.extend(['--nanolog', self.nanomsg])
         args.extend(['--device-id', str(self.device_id)])
         P4Switch.device_id += 1
-        args.append(self.json_path)
+        args.append("--no-p4")
         if self.enable_debugger:
             args.append("--debugger")
         if self.log_console:
```

Uruchomienie sieci

```
1sw_demo.py                                                    \
    --behavioral-exe /usr/bin/simple_switch_grpc \
    --json out/main.json
```

- Zostanie uruchomiony serwer P4Runtime pod adresem 0.0.0.0:9559
  - Trzeba dodać opcję `--grpc-server-addr` aby zmienić
- Nadal można korzystać z protokołu Thrift (simple_switch_CLI), ale ma to status eksperymentalny

Kompilacja

```
p4c                                     \
    --target bmv2                       \
    --arch v1model                      \
    --p4runtime-files p4info.txt        \
    -o out/                             \
    plik.p4
```

Klient P4Runtime

```
python3                                 \
    -m p4runtime_sh                     \
    --grpc-addr localhost:9559          \
    --device-id 0                       \
    --election-id 0,1                   \
    --config p4info.txt,out/main.json
```

- Opcja `--device-id` wskazuje który switch docelowy (serwer może obsługiwać wiele switchy jednocześnie)
- Opcja `--election-id` służy do wyboru klienta z prawem do zapisu (największe `election-id`)
  - Może być podłączonych wielu klientów, ale tylko jeden z nich może modyfikować stan switcha
  - Liczba 128-bitowa zapisana jako para liczb 64-bitowych
- Opcja `--config` instaluje program w switchu (nie ma potrzeby restartowania Minineta, aby wgrać nowy program P4)
  - Nie jest obowiązkowa - bez niej klient pobierze z serwera konfigurację switcha
- Interpreter Pythona, działa tabulator

P4Runtime - operacje

- Dodanie wpisu do tabeli

  Wg dokumentacji: https://github.com/p4lang/p4runtime-shell
- Odczytanie digestu

  https://github.com/p4lang/p4runtime-shell/pull/102

Źródło: https://opennetworking.org/wp-content/uploads/2018/12/ONOS-support-for-P4.pdf

ONOS

"Pipeconf" - Bring your own pipeline!

- **Package together everything necessary to let ONOS understand, control, and deploy an arbitrary pipeline**
- **Provided to ONOS as an app**
  - Can use .oar format for distribution

1. **Pipeline model**
   - Description of the pipeline understood by ONOS
   - Automatically derived from P4Info
2. **Target-specific binaries to deploy pipeline to device**
   - E.g. BMv2 JSON, Tofino binary, FPGA bitstream, etc.
3. **Pipeline-specific driver behaviors**
   - E.g. "Pipeliner" implementation: logic to map FlowObjectives to P4 pipeline

Przydatne materiały

- Wszystkie materiały z poprzednich zajęć
- Specyfikacja protokołu P4Runtime: https://p4.org/p4-spec/p4runtime/v1.3.0/P4Runtime-Spec.html
- P4Runtime-shell: https://github.com/p4lang/p4runtime-shell
- Stratum: https://github.com/stratum/stratum/

Przygotowanie RPi do projektu

- Zainstalowanie obrazu Raspberry Pi OS
- Przygotowanie dostępu zdalnego i dostępu do sieci
- Instalacja switcha BMv2