Správa sieťových infraštruktúr – Protokol OpenFlow

Úvod

Protokol **OpenFlow** sa používa ako komunikačný protokol medzi kontrolérom a sieťovými zariadeniami. Jeho úlohou je transport dát potrebných pre riadenie dátovej roviny na prepínačoch. Kontrolér dokáže využitím tohto protokolu pridávať, aktualizovať a mazať záznamy v tabuľke tokov, na základe ktorých zariadenia dokážu preposielať dáta. Každý záznam o toku pozostáva z **match** poľa, špecifikujúceho charakteristiky paketu, **counter** poľa uchovávajúceho štatistiku o počte identifikovaných paketov a z poľa **actions**, ktoré určuje množinu akcií, ktoré sa na danom identifikovanom pakete vykonajú. Ak tabuľka tokov neobsahuje záznam pre daný paket, tak sa vykoná štandardná akcia, ktorou je buď odoslanie paketu na kontrolér (správa **Packet_in**), ktorý následne určí, ako daný paket odoslať (správa **Packet_out**), alebo je daný paket zahodený. Každý paket je možné zaradiť do dátového toku na základe viacerých charakteristík, medzi ktoré je možné zaradiť: zdrojovú a cieľovú MAC adresu, zdrojovú a cieľovú IP adresu, zdrojový a cieľový port, číslo protokolu ...

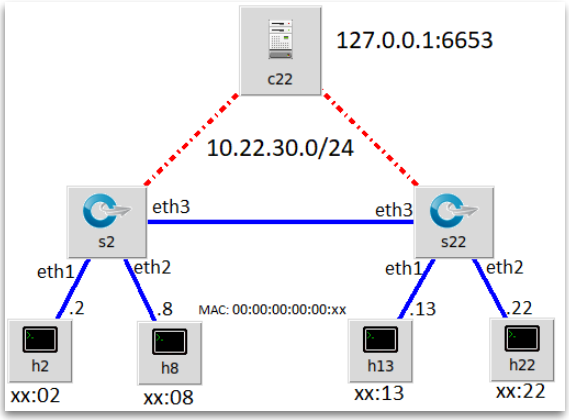
Cieľom tejto kapitoly je na názorných ukážkach vysvetliť ako je možné spravovať sieťovú infraštruktúru využitím práve protokolu OpenFlow. Ukázaný je manuálny prístup využitím terminálových príkazov, prístup využitím grafického rozhrania kontroléra OpenDayLight, prístup využitím nástroja OpenFlow Manager a štandardné využitie nástrojov Postman a Curl. Pre ukážku automatizovanej správy protokolom OpenFlow sú využité jazyky Python a C# (desktopová aplikácia).

Požiadavky pre realizáciu úloh:

- Pripravené virtuálne zariadenie
  - Mininet
  - Kontrolér OpenDayLight a nástroj OpenFlow Manager
  - Postman, Curl
- Prostredie pre prácu s programovacími jazykmi Python a C#

Topológia

Topológia (obrázok 6.1), na ktorej sú realizované experimenty pozostáva z koncových zariadení H2, H8, H13 a H22, nachádzajúcich sa v sieti 10.22.30.0/24, z dvoch OpenFlow prepínačov S2 a S22 a z externého kontroléra C22 (OpenDayLight):



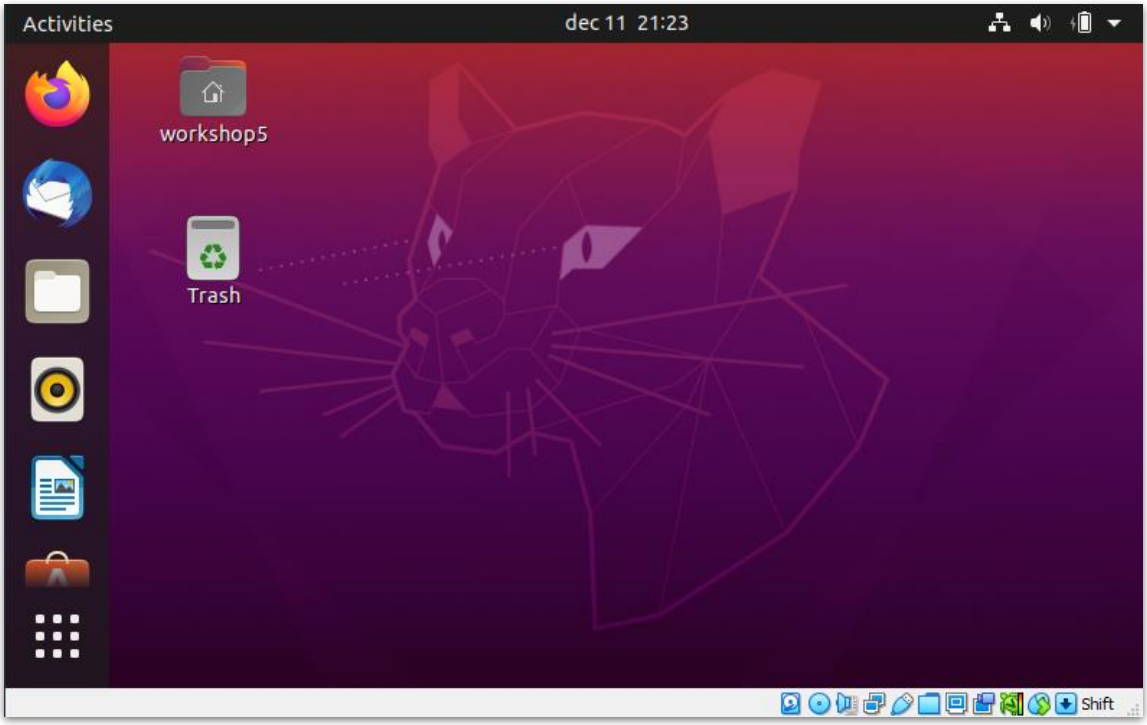
Obrázok 6.1 Navrhnutá topológia

Kontrolér C22 (OpenDayLight) bude spravovať prepínače S2 a S22 využitím komunikačného protokolu OpenFlow. Na kontrolér bude možné odosielať záznamy pre tvorbu tabuliek tokov využitím protokolu HTTP z nástrojov: YangUI (súčasť ODL), OFM, Curl, Postman, Python skript a C# kontrolér.

Úloha 0: Oboznámenie sa s virtuálnym zariadením Workshop_5

Workshop_5 je virtuálne zariadenie (OS: Ubuntu 20.04), na ktorom sú predinštalované všetky nástroje potrebné pre zvládnutie tohto Workshopu. Obsahuje emulátor Mininet, ODL kontrolér s nainštalovaným rozšírením pre web GUI, nástroje OpenFlow Manager, Curl, Postman a Python3. ODL a OFM je možné nájsť v domovskom adresári, odkiaľ ich je možné aj spustiť.

Virtuálne zariadenie je možné importovať do VirtualBox-u, pričom je potrebné sa uistiť, že mu bude vyhradené dostatočné množstvo RAM pamäte (min. 4096 MB) a že mu bude, pre sieťové pripojenie, nastavený Bridge adapter (komunikácia aj z externých zariadení). Po úspešnom spustení a prihlásení [login: **Workshop5**, password: **openflow**] by mala byť zobrazená nasledujúca obrazovka (obrázok 6.2):



Obrázok 6.2 Pracovná plocha virtuálneho zariadenia Workshop_5

Úloha 1: Oboznámenie sa s Mininet-om

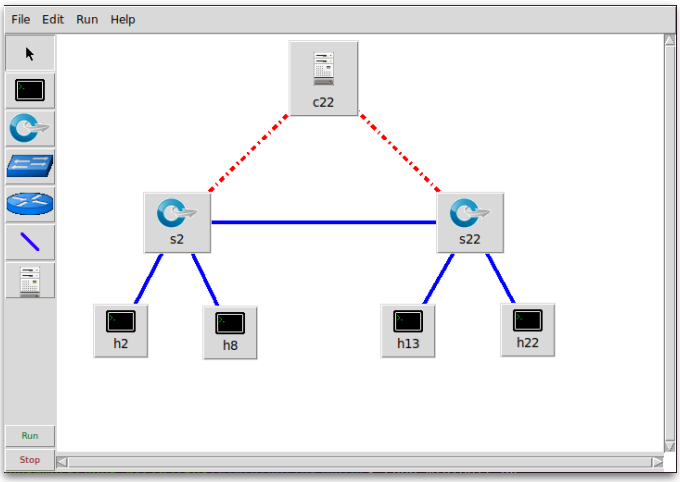
Mininet je nástroj umožňujúci vytvárať virtuálne sieťové topológie využitím kernel-u zariadenia. Má priamu podporu pre OpenFlow prepínače a integráciu aplikačného kódu → prepojenie s kontrolérom. Topológie je možné vytvárať tromi základnými prístupmi: využitím grafického editora MiniEdit, využitím terminálu alebo využitím Python skriptu.

A. **Nástroj MiniEdit:**

MiniEdit je grafické rozhranie, v ktorom je možné ťahaním objektov na plátno (host, prepínač, kontrolér, prepojenia) vytvoriť požadovanú topológiu. Každému objektu je po jeho vložení do topológie možné nastaviť potrebné parametre, ako je názov, adresa, typ a pod. Vytvorenú topológiu je možné spustiť alebo uložiť v podobe .mn súboru alebo Python skriptu. Spustenie je možné nasledovným príkazom z domovského adresára:

```
sudo python3 /mininet/examples/miniedit.py
```

Topológia vytvorená v MiniEdit nástroji by mohla vyzerať nasledovne (obrázok 6.3):



Obrázok 6.3 Navrhnutá topológia vytvorená v nástroji MiniEdit

B. **Spustenie z terminálu:**

Vytvorenie topológie je možné aj priamo z terminálu, no tento prístup je obmedzujúci na špecifikáciu implementačných detailov. Pre kompletnosť je však nižšie uvedený vzorový príkaz, ktorým je možné definovať tvorbu základnej topológie, pozostávajúcej z prepínača, ku ktorému sú pripojené 4 koncové zariadenia:

```
sudo mn –topo=single,4
```

C. **Python skript:**

Tento prístup je odporúčaný a je použitý aj v rámci tejto kapitoly. Skript umožňuje detailné nastavenie topológie spolu so špecifikovaním všetkých parametrov charakterizujúcich jednotlivé zariadenia, medzi ktoré je možné zaradiť: typ zariadení, verziu protokolu, IP adresy, MAC adresy, názov zariadení, vykonanie testov, spustenie terminálu a pod. Spustenie vytvoreného skriptu (mytopo.py) a teda aj celkovej topológie v Mininet-e je možné využitím nasledujúceho príkazu:

```
sudo python3 mytopo.py
```

Nasledujúci zdrojový kód predstavuje skript pre tvorbu topológie potrebnej pre tento manuál:

```
net = Mininet(topo=None, build=False, ipBase='10.22.30.0/24')

info('*** Adding controller\n')
c22 = net.addController(name='c22', controller=RemoteController, ip='127.0.0.1', protocol='tcp',
port=6653)

info('*** Add switches\n')
s2 = net.addSwitch('s2', cls=OVSKernelSwitch, protocols="OpenFlow13")
s22 = net.addSwitch('s22', cls=OVSKernelSwitch, protocols="OpenFlow13")

info('*** Add hosts\n')
h2 = net.addHost('h2', cls=Host, ip='10.22.30.2', mac='00:00:00:00:00:02', defaultRoute=None)
h8 = net.addHost('h8', cls=Host, ip='10.22.30.8', mac='00:00:00:00:00:08', defaultRoute=None)
h13 = net.addHost('h13', cls=Host, ip='10.22.30.13', mac='00:00:00:00:00:13', defaultRoute=None)
h22 = net.addHost('h22', cls=Host, ip='10.22.30.22', mac='00:00:00:00:00:22', defaultRoute=None)

info('*** Add links\n')
net.addLink(h2, s2)
net.addLink(h8, s2)
net.addLink(h13, s22)
```

```
net.addLink(h22, s22)
net.addLink(s2, s22)

info('*** Starting network\n')
net.build()

info('*** Starting controllers\n')
for controller in net.controllers:
    controller.start()

info('*** Starting switches\n')
net.get('s2').start([c22])
net.get('s22').start([c22])

#net.pingAll()
CLI(net)
net.stop()
```

Uvedený skript neobsahuje celkový kód, len sekciu funkcie, pre tvorbu topológie. Pre úplnú funkčnosť je potrebné doplniť chýbajúce importy a *main* funkciu. Tieto časti sa ale nachádzajú vo vzorovom skripte v predpripravenom VM Workshop_5.

Obrázok 6.4 znázorňuje spustenie Mininet prostredia a použitie základných príkazov pre prácu a overenie topológie:

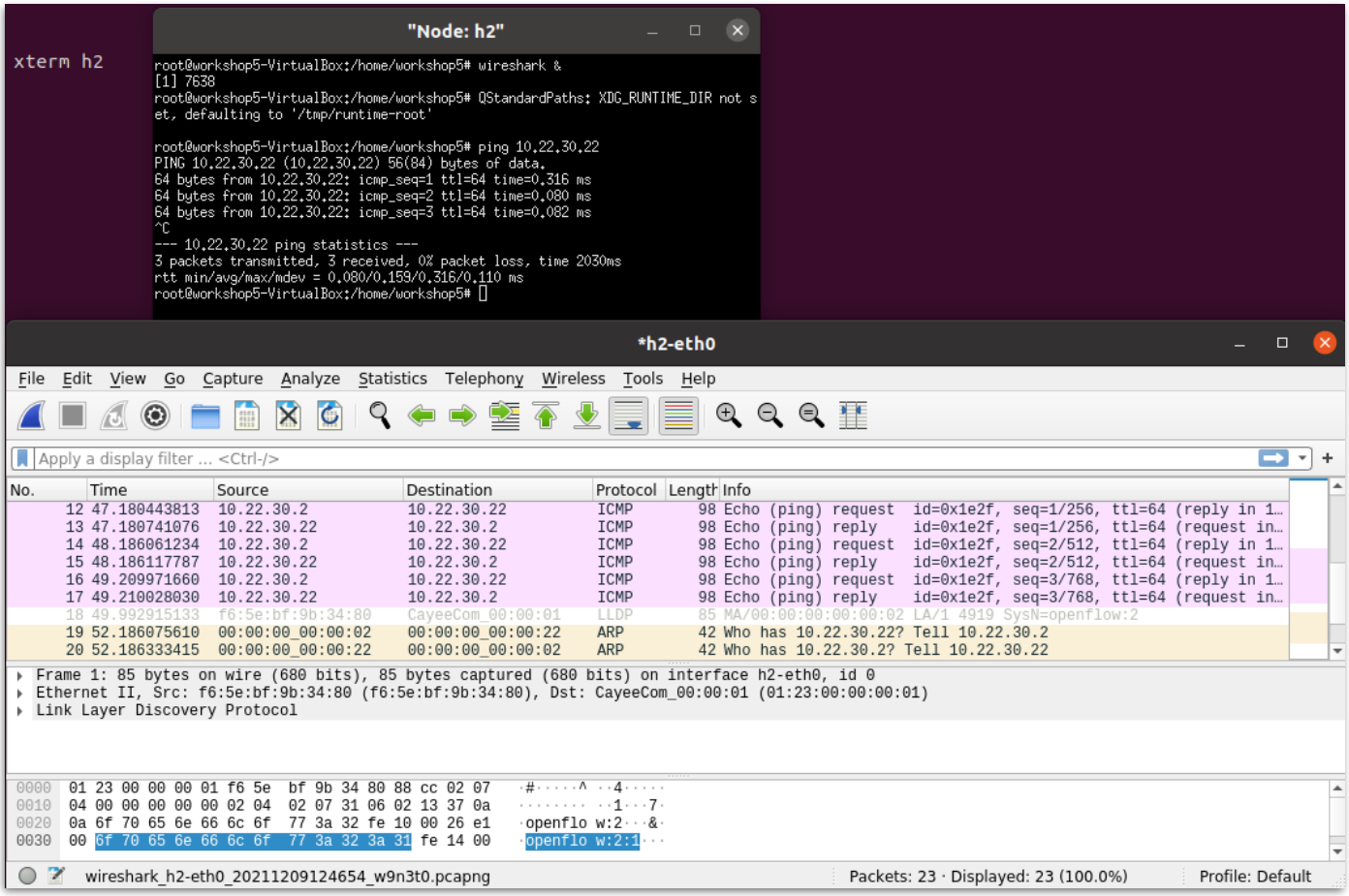
```
workshop5@workshop5-VirtualBox:~$ sudo python3 mytopo.py
[sudo] password for workshop5:
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h2 h8 h13 h22
*** Starting controllers
*** Starting switches
*** Starting CLI:
mininet>
mininet> nodes
available nodes are:
c22 h13 h2 h22 h8 s2 s22
mininet>
mininet> net
h2 h2-eth0:s2-eth1
h8 h8-eth0:s2-eth2
h13 h13-eth0:s22-eth1
h22 h22-eth0:s22-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:h8-eth0 s2-eth3:s22-eth3
s22 lo:  s22-eth1:h13-eth0 s22-eth2:h22-eth0 s22-eth3:s2-eth3
c22
mininet>
mininet> links
h2-eth0<->s2-eth1 (OK OK)
h8-eth0<->s2-eth2 (OK OK)
h13-eth0<->s22-eth1 (OK OK)
h22-eth0<->s22-eth2 (OK OK)
s2-eth3<->s22-eth3 (OK OK)
mininet>
mininet> dump
<Host h2: h2-eth0:10.22.30.2 pid=5265>
<Host h8: h8-eth0:10.22.30.8 pid=5267>
<Host h13: h13-eth0:10.22.30.13 pid=5269>
<Host h22: h22-eth0:10.22.30.22 pid=5271>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=5256>
<OVSSwitch s22: lo:127.0.0.1,s22-eth1:None,s22-eth2:None,s22-eth3:None pid=5260>
<RemoteController c22: 127.0.0.1:6653 pid=5249>
mininet>
mininet> h2 ping h13
PING 10.22.30.13 (10.22.30.13) 56(84) bytes of data.
64 bytes from 10.22.30.13: icmp_seq=1 ttl=64 time=0.432 ms
```

Obrázok 6.4 Spustenie nástroja Mininet a prieskum topológie

Mininet umožňuje pracovať so všetkými zariadeniami ako so štandardnými linuxovými systémami, kde je možné po zadaní názvu zariadenia zadať terminálový príkaz, ktorý sa má vykonať:

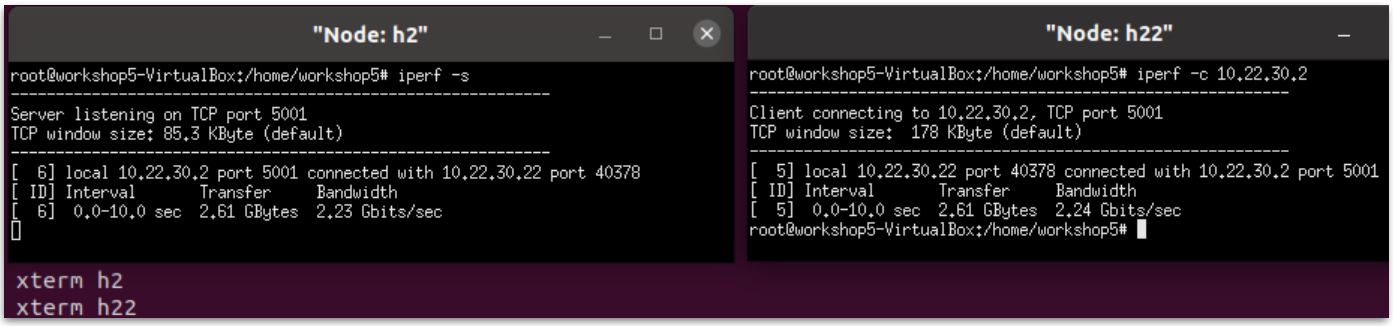
```
mininet> h2 ifconfig
mininet> h2 python -m http.server 80 &
mininet> h22 curl h2
mininet> xterm h2
```

Využitím posledného zmieneného príkazu je možné otvoriť terminál zariadenia, s ktorým je možné pracovať bežným spôsobom. Obrázok 6.5 dokumentuje spustenie nástroja *Wireshark*:



Obrázok 6.5 Spustenie nástroja Wireshark

Testovanie priepustnosti je možné realizovať nástrojom `iperf`, ako znázorňuje obrázok 6.6:



Obrázok 6.6 Testovanie priepustnosti

Užitočný príkaz je `pingall`, ktorým je možné overiť konektivitu medzi všetkými koncovými zariadeniami, čo dokumentuje obrázok 6.7:

```
mininet> pingall
*** Ping: testing ping reachability
h2 -> h8 h13 h22
h8 -> h2 h13 h22
h13 -> h2 h8 h22
h22 -> h2 h8 h13
*** Results: 0% dropped (12/12 received)
```

Obrázok 6.7 Použitie príkazu pingall

Pozn.: V prípade, že je v topológii prístupný kontrolér alebo topológia s kontrolérom vôbec nepočíta, tak po spustení automaticky funguje dostupnosť všetkých zariadení. Konektivita je zabezpečená tak, že v prípade existencie kontroléra sú automaticky vložené záznamy o tokoch do tabuliek tokov, ktoré umožňujú komunikáciu medzi kontrolérom a prepínačom a zároveň posielajú štandardnú komunikáciu cez všetky rozhrania (neefektívne). Ak však kontrolér v topológii nie je definovaný, tak sa prepínač správa ako bežný L2 prepínač. Ak je kontrolér definovaný, ale nie je dostupný, tak medzi zariadeniami komunikácia fungovať nebude.

Nasledujúce sekcie sa budú venovať ukážkam rôznych prístupov, ako spravovať tabuľky tokov na OpenFlow prepínačoch, za predpokladu existencie ODL kontroléra.

Úloha 2: Riadenie tokov – manuálny prístup

OpenFlow prepínače je možné spravovať bežným spôsobom využitím sady terminálových príkazov. Z prostredia Mininet je možné spustiť príkaz shell-u OpenFlow prepínača príkazom:

```
mininet> sh ovs-ofctl
```

Ukážme si to na príklade: Najskôr sa je potrebné oboznámiť s prepínačmi, ktoré máme k dispozícii. Najdôležitejšími vlastnosťami sú ich rozhrania, pričom potrebujeme identifikovať ich čísla. Príkaz **`show`** umožňuje zobrazenie základných informácií o prepínači. Jeho výstup je znázornený na obrázku 6.8:

```
mininet> sh ovs-ofctl show s2 -O OpenFlow13
OFPT_FEATURES_REPLY (OF1.3) (xid=0x2): dpid:0000000000000002
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS QUEUE_STATS
OFPST_PORT_DESC reply (OF1.3) (xid=0x3):
 1(s2-eth1): addr:a2:40:51:8e:c9:15
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 2(s2-eth2): addr:0e:7e:ad:b8:b0:c3
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 3(s2-eth3): addr:96:40:ec:6c:c7:e3
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(s2): addr:86:d5:e3:24:dd:43
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.3) (xid=0x9): frags=normal miss_send_len=0
```

Obrázok 6.8 Výstup príkazu show pre výpis základných informácií o prepínači

Upozornenie: Štruktúra príkazu je nasledovná. Najskôr je potrebné zadať kľúčové slovo **sh ovs-ofctl**. Nasleduje príkaz, ktorý chceme vykonať – v našom prípade príkaz **show**. Za príkazom nasleduje špecifikácia zariadenia, na ktoré chceme príkaz aplikovať. Nakoniec je ešte potrebné špecifikovať verziu protokolu OpenFlow príkazom **-O OpenFlow13**. V tomto dokumente sa pracuje s verziou 1.3 (potrebné pre kompatibilitu s verziou kontroléra).

OpenFlow prepínač sa na začiatku automaticky naučí sadu záznamov o tokoch, ktoré umožnia komunikáciu s kontrolérom a zároveň smerovanie bežnej prevádzky každým rozhraním. Zobrazenie tabuľky tokov (obrázok 6.9) je možné vykonať príkazom:

```
mininet> sh ovs-ofctl dump-flows s2 -O OpenFlow13
```

```
mininet> sh ovs-ofctl dump-flows s2 -O OpenFlow13
 cookie=0x2b00000000000007, duration=244.098s, table=0, n_packets=51, n_bytes=4437, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2b00000000000006, duration=240.096s, table=0, n_packets=10, n_bytes=756, priority=2,in_port="s2-eth1" actions=output:"s2-eth2",output:"s2-eth3",CONTROLLER:65535
 cookie=0x2b00000000000007, duration=240.080s, table=0, n_packets=4, n_bytes=280, priority=2,in_port="s2-eth2" actions=output:"s2-eth1",output:"s2-eth3",CONTROLLER:65535
 cookie=0x2b00000000000008, duration=240.077s, table=0, n_packets=19, n_bytes=1534, priority=2,in_port="s2-eth3" actions=output:"s2-eth1",output:"s2-eth2"
 cookie=0x2b00000000000007, duration=244.098s, table=0, n_packets=47, n_bytes=4023, priority=0 actions=drop
```

Obrázok 6.9 Zobrazenie tabuľky tokov

Každý záznam v tabuľke tokov pozostáva zo štandardných štatistických údajov ako trvanie (ako dlho je daný záznam v tabuľke), identifikátor tabuľky, v ktorej sa nachádza (štandardne tab. 0) a množstva prenesených paketov a bajtov daným tokom. Každý tok tiež obsahuje informáciu o priorite (rozsah od <0-65,535>. Vyššia hodnota je preferovanejšia. Pri nešpecifikovaní sa jedná o prioritu 32,768). Nasledujú **match** výrazy, medzi ktoré patria: vstupný port [**in_port**], L2 údaje [**dl_type/src/dst**], L3 údaje [**nw_proto/src/dst**] a L4 údaje [**tp_src/dst**]. Poslednou časťou je sekcia **actions**, ktorá slúži na definovanie procedúry, ktorá sa vykoná na identifikovaných paketoch. Najčastejšie je to akcia **output** definujúca výstupné rozhranie, ktorým treba paket poslať. Tiež sa môže jednať o akciu **flood**, ktorá je vhodná napr. pre protokol ARP. Štandardné správanie je možné spustiť akciou **normal/drop** (zahoď).

Ukázať si však chceme postupné budovanie tabuľky tokov a preto je najskôr potrebné vymazať automaticky vložené záznamy. Tým, že sa zmaže aj záznam, ktorý povoľuje posielanie požiadaviek na kontrolér, tak po tomto manuálnom odstránení tabuľky, sa už prepínač nebude môcť naučiť od kontroléra žiadne záznamy. Príkaz na zmazanie tabuľky tokov je potrebné zadať na oboch prepínačoch, pričom variant príkazu pre prepínač s2 vyzerá nasledovne:

```
mininet> sh ovs-ofctl del-flows s2 -O OpenFlow13
```

Začnime ukážkou manuálneho vloženia dátových tokov, ktoré identifikujú prenášané správy len na základe L2 informácií. Vzorová sada príkazov pre povolenie komunikácie medzi zariadením h13 a h22 by mohla vyzerať nasledovne:

```
mininet> sh ovs-ofctl add-flow s22 priority=300,dl_type=0x806,actions=flood -O
OpenFlow13

mininet> sh ovs-ofctl add-flow s22
priority=300,dl_dst=00:00:00:00:00:22,actions=output:2 -O OpenFlow13

mininet> sh ovs-ofctl add-flow s22
priority=300,dl_dst=00:00:00:00:00:13,actions=output:1 -O OpenFlow13
```

Prvý príkaz špecifikuje typ správ s kódom 0x806 (ARP protokol), ktorého správy budú posielané všetkými rozhraniami (Flood). Tento príkaz je kľúčový, pretože bez neho by h13 a h22 nedokázali získať MAC adresu cieľa.

Druhý príkaz hovorí, že pokiaľ v pakete/rámci bude cieľová MAC adresa (00:00:00:00:00:22), čo je adresa h22, tak tento paket/rámec bude odoslaný cez výstupné rozhranie 2 (output:2), z prvého výpisu v tejto sekcii je možné nájsť reálny názov rozhrania, ktorým je: s22-eth2.

Tretí príkaz v poradí definuje, že správy s cieľovou MAC adresou končiacou na :13 budú odoslané výstupným rozhraním 1.

Podobný prístup je možné využiť aj pri identifikovaní paketu na základe L3 informácií. Nasledujúci výpis predstavuje skript pre vloženie tokov na zabezpečenie plnej dostupnosti všetkých zariadení. Aby bolo možné z výstupu priamo kopírovať príkazy, tak v nich nebude časť prompt-u:

```
sh ovs-ofctl add-flow s2 priority=300,dl_type=0x806,actions=flood -O OpenFlow13
sh ovs-ofctl add-flow s2 dl_type=0x800,nw_dst=10.22.30.2,actions=output:1 -O OpenFlow13
sh ovs-ofctl add-flow s2 ip,nw_dst=10.22.30.8,actions=output:2 -O OpenFlow13
sh ovs-ofctl add-flow s2 priority=500,ip,nw_dst=10.22.30.0/24,actions=output:3 -O OpenFlow13

sh ovs-ofctl add-flow s22 priority=300,dl_type=0x806,actions=flood -O OpenFlow13
sh ovs-ofctl add-flow s22 ip,nw_dst=10.22.30.13,actions=output:1 -O OpenFlow13
sh ovs-ofctl add-flow s22 ip,nw_dst=10.22.30.22,actions=output:2 -O OpenFlow13
sh ovs-ofctl add-flow s22 priority=500,ip,nw_dst=10.22.30.0/24,actions=output:3 -O OpenFlow13
```

Práca je v tomto prípade podobná. Dôležité je poznamenať, že typ protokolu je možné zadať v hexa kóde, ale aj v tvare kľúčového slova – (**dl_type=0x800** je to isté ako **ip**). Cieľovú adresu je možné špecifikovať ako konkrétneho **hosta**, ale aj ako rozsah adries využitím **/mask** časti. V tomto prípade je ale dôležité, aby menej špecifický záznam mal nižšiu prioritu. V našom prípade to splnené je, pretože priorita 500 je menej ako štandardná hodnota 32,768.

Pre identifikovanie toku na základe L4 záhlavia by príkaz mohol vyzerať nasledovne:

```
sh ovs-ofctl add-flow s22 priority=300,dl_type=0x806,actions=flood -O OpenFlow13
sh ovs-ofctl add-flow s22 ip,nw proto=6,tp dst=80,actions=output:2 -O OpenFlow13
```

Dôležité si je uvedomiť, že aj napriek tomu, že správy definujeme na základe L4 charakteristík, tak nesmieme zabudnúť na povolenie ARP komunikácie. ARP funkcionalitu je možné povoliť príkazom:

```
sh ovs-ofctl add-flow s22 arp,actions=normal
```

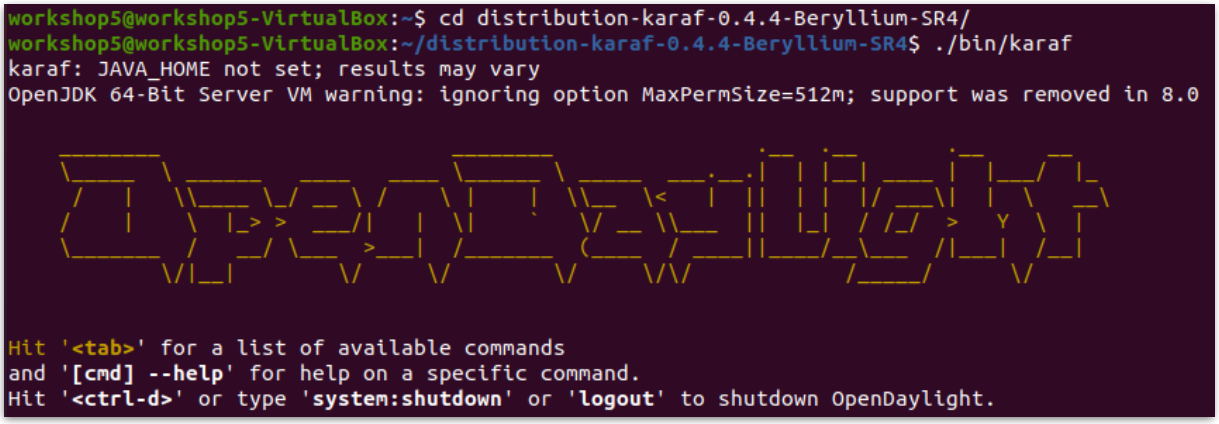
Funkčnosť špecifikácie L4 dátového toku overíme spustením HTTP servera, ktorého dostupnosť, z pohľadu aplikačnej vrstvy, je možné overiť nástrojom Curl. Príklad použitia je znázornený v nasledujúcom výpise:

```
mininet> h22 python -m SimpleHTTPServer 80 &
mininet> h13 curl -i h22
```

Pri záznamoch o tokoch je tiež možné definovať dĺžku ich existencie v tabuľke tokov a to atribútmi **idle_timeoute** (doba neaktivity) a **hard_timeoute** (fixná doba od naučenia).

Úloha 3: Riadenie tokov – OpenDayLight (ODL)

OpenDayLight je jeden z najkomplexnejších verejne dostupných kontrolérov pre prácu s OpenFlow protokolom. Spustiť ho je možné postupom znázorneným na obrázku 6.10:



Obrázok 6.10 Spustenie kontroléra OpenDayLight

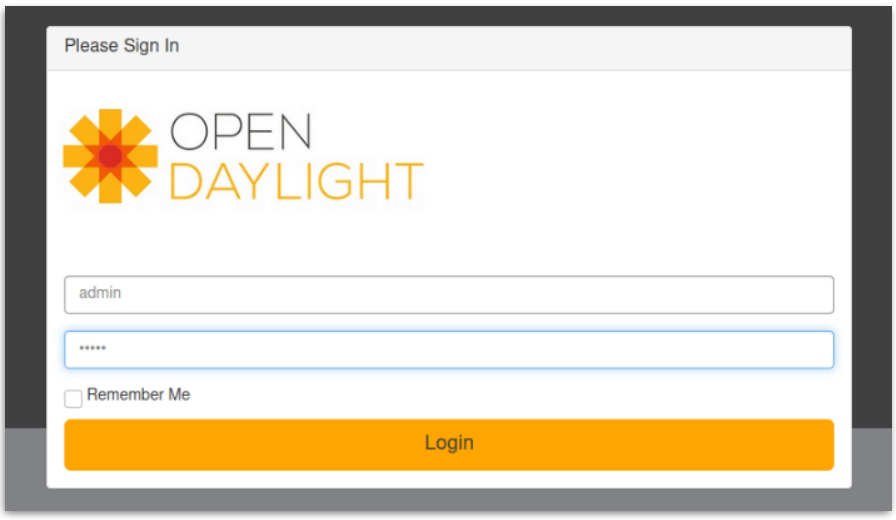
V štandardnom nastavení kontrolér vykonáva len základné činnosti. V prípade potreby pokročilejších funkcionalít, je potrebné ich doinštalovať. Predpripravený kontrolér je plne nainštalovaný. Pre úplnosť je uvedená množina potrebných príkazov pre inštaláciu GUI (dlux) a pod.:

```
feature:install odl-mdsal-clustering

feature:install odl-dlux-core odl-dlux-node odl-dlux-yangui odl-dlux-yangvisualizer

feature:install odl-l2switch-all odl-restconf-all odl-openflowplugin-all odl-yangtools-
common odl-mdsal-all
```

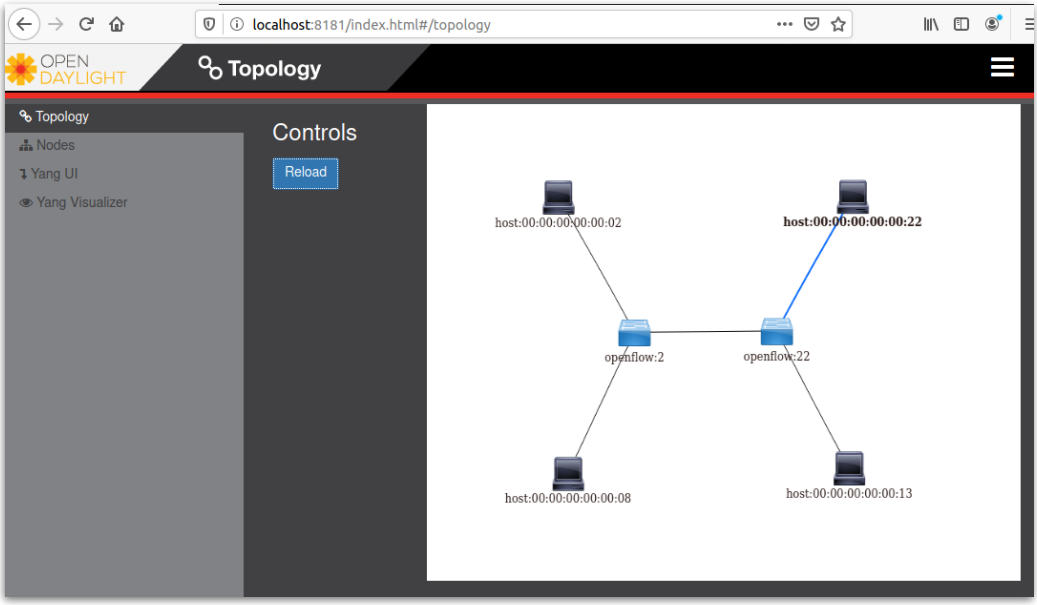
V tejto chvíli sa vieme pripojiť na grafickú nadstavbu kontroléra ODL využitím Webového prehliadača, kde stačí zadať nasledujúcu URL a následne sa prihlásiť menom a heslom **admin**.

```
http://localhost:8181/index.html#/login
```

Obrázky 6.11 a 6.12 znázorňujú prihlasovacie okno a základné prostredie zobrazujúce topológiu:

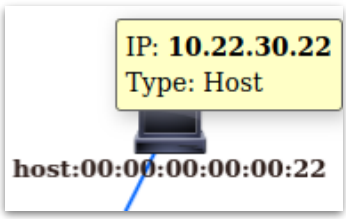


Obrázok 6.11 Prihlasovacie okno OpenDayLight kontroléra vo webovom prehliadači

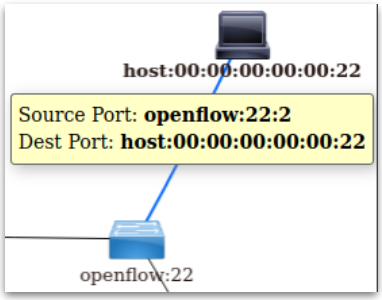


Obrázok 6.12 *Prostredie kontroléra znázorňujúce zapojenú topológiu*

Prechodom kurzoru nad objektami je možné zobraziť ich základné údaje ako je MAC a IP adresa, čo zobrazujú obrázky 6.13 a 6.14:

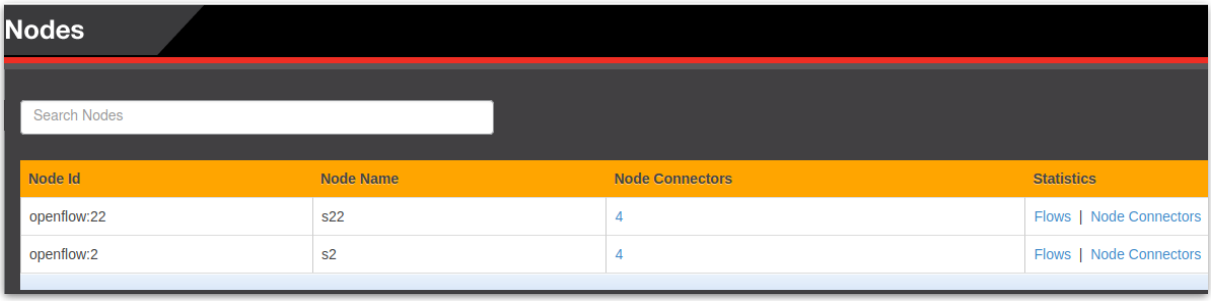


Obrázok 6.13 *Zobrazenie IP adresy*

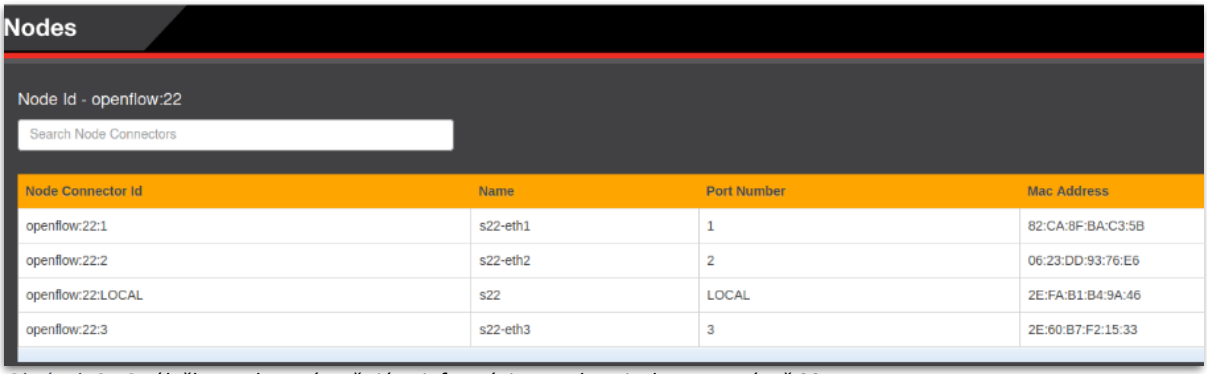


Obrázok 6.14 *Zobrazenie MAC adresy*

Záložka **Nodes** v paneli na ľavej strane, obsahuje zoznam všetkých monitorovaných OpenFlow prepínačov a po rozkliknutí aj detailné informácie o ich rozhraniach (obrázky 6.15 a 6.16):



Obrázok 6.15 *Záložka Nodes znázorňujúca zoznam všetkých monitorovaných OpenFlow prepínačov*

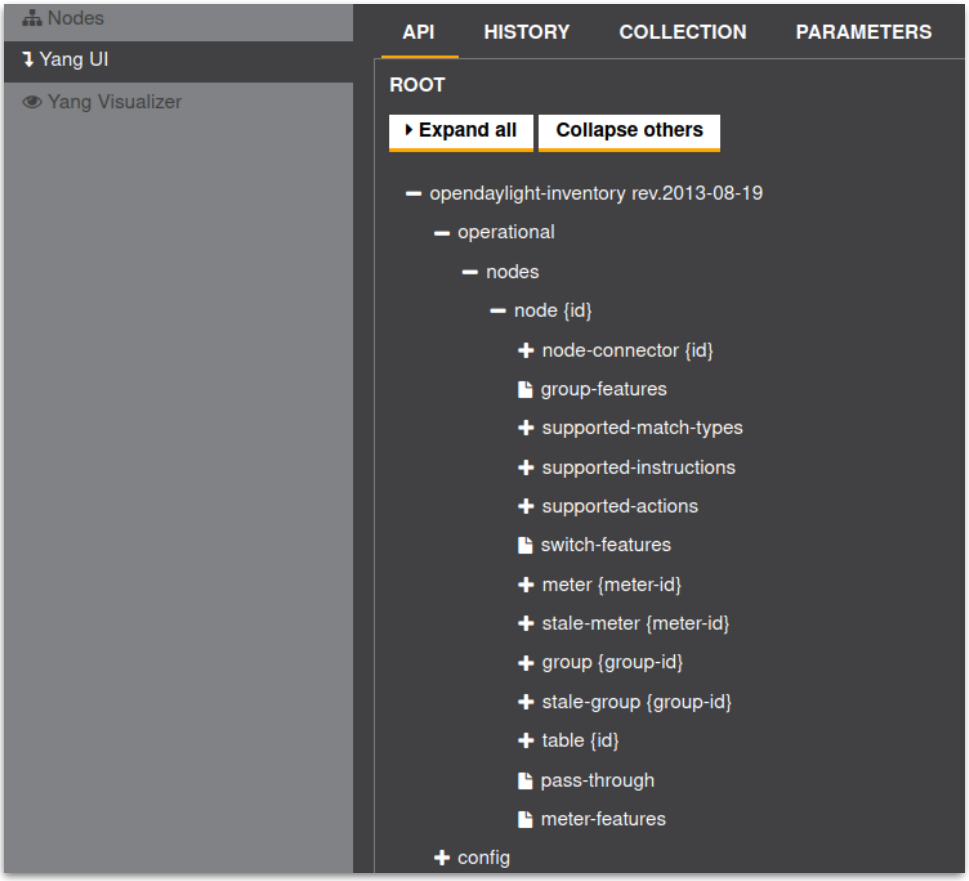


Obrázok 6.16 *Záložka Nodes znázorňujúca informácie o rozhraniach pre prepínač 22*

Najdôležitejšou záložkou pre správu monitorovanej OpenFlow infraštruktúry je záložka **Yang UI**. Táto záložka graficky znázorňuje štruktúru YANG modelov, ktorú je možné využívať na vyčítavanie, zmenu a mazanie záznamov o tokoch na spravovaných OpenFlow prepínačoch. Najvhodnejším YANG modelom je: **opendaylight-inventory:nodes**, ktorý obsahuje sekciu **operational** pre čítanie údajov a sekciu **config** pre ich zmenu.

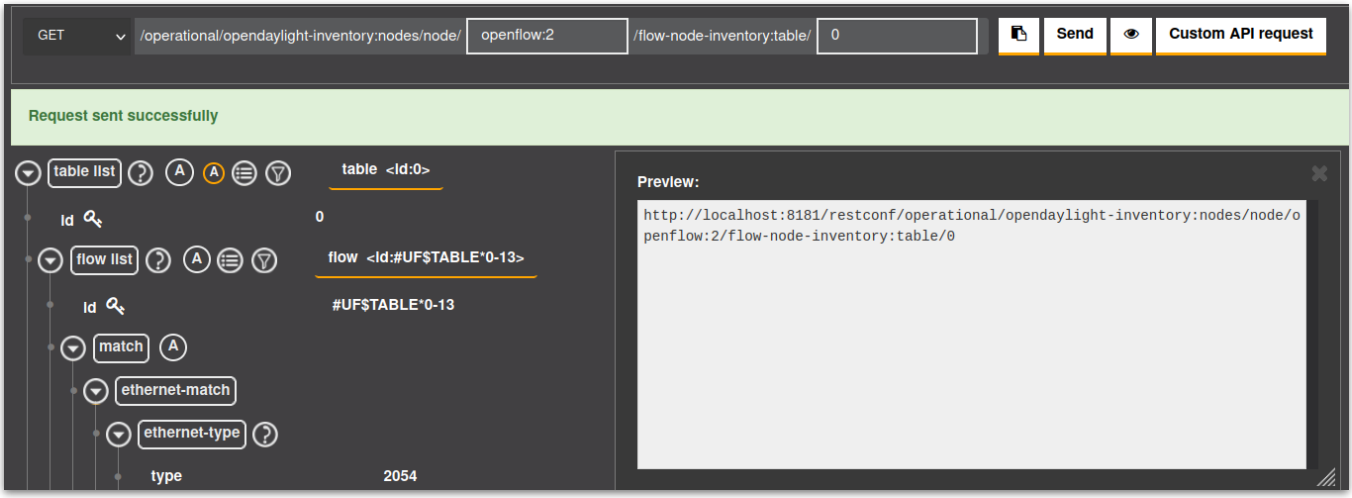
```
http://localhost:8181/restconf/operational/opendaylight-
inventory:nodes/node/openflow:2/flow-node-inventory:table/0

http://localhost:8181/restconf/config/opendaylight-
inventory:nodes/node/openflow:22/flow-node-inventory:table/0
```

Ako je možné vidieť z uvedených URL adries, grafické rozhranie komunikuje s kontrolérom využitím protokolu RESTCONF. K uvedeným zdrojom sa dostaneme cez Yang UI, čo vidíme na obrázku 6.17:



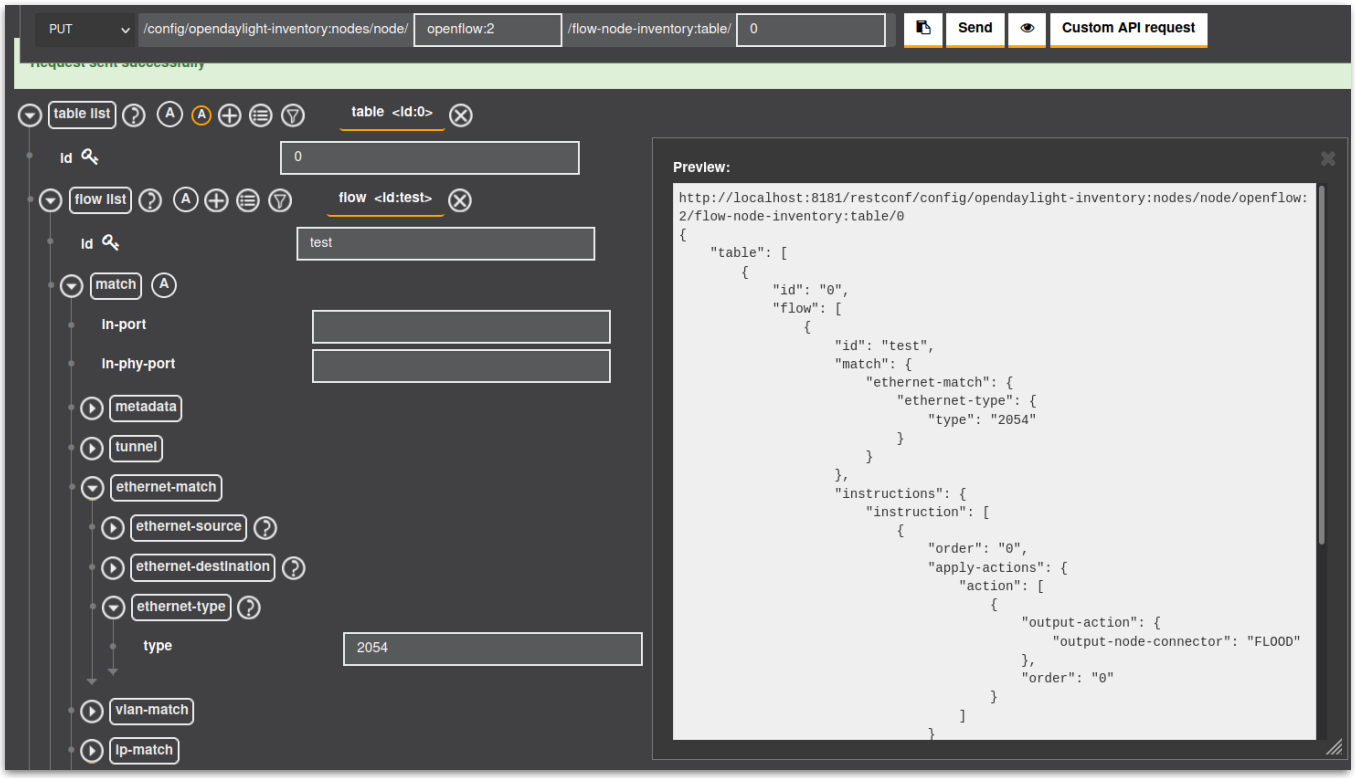
Obrázok 6.17 Zobrazenie stromovej štruktúry modulu inventory

Po zvolení príslušného YANG modelu sa v spodnej časti zobrazí prehliadač, v ktorom je možné zvoliť akú požiadavku chceme a zároveň je možné vyplniť príslušné atribúty. Dôležitá poznámka je, že názov zariadenia je potrebné vkladať v tvare – napr. pre zariadenie s2 je názov **openflow:2**. Zároveň je po odoslaní možné zobraziť show preview, ktoré v sebe obsahuje URL a telo HTTP požiadavky, čo je možné vidieť na obrázku 6.18:



Obrázok 6.18 Vykonanie GET požiadavky

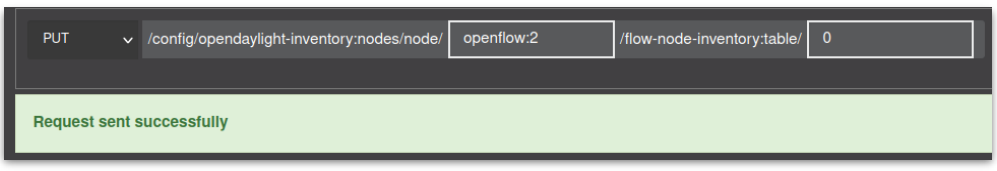
V prípade sekcie **config** a metódy **PUT** je možné údaje o dátových tokoch aj meniť, čo je možné vidieť na obrázku 6.19. Takto je možné nastaviť čo potrebujeme a odoslať požiadavku tlačidlom **send**. Ako je možné vidieť, tak telo požiadavky je vo formáte JSON, s ktorým je neskôr tiež možné pracovať využitím programovacích jazykov pri tvorbe automatizačných nástrojov:



Obrázok 6.19 Vykonanie PUT požiadavky

Pri tvorbe takejto požiadavky je potrebné vyplniť všetky potrebné atribúty, medzi ktoré patrí napr. číslo tabuľky, názov toku a podobne.

Po úspešnom vykonaní operácie sa zobrazí kontrolná správa (obrázok 6.20):



Obrázok 6.20 Výpis o úspešnom odoslaní PUT požiadavky

Pozn.: YANG model pre zobrazenie štatistík rozhraní je uvedený v nasledujúcom výpise:

```
http://localhost:8181/restconf/operational/opendaylight-
inventory:nodes/node/openflow:2/node-connector/openflow:2:2/opendaylight-port-
statistics:flow-capable-node-connector-statistics
```

Úloha 4: Riadenie tokov – OpenFlow Manager (OFM)

OpenFlow Manager je externá aplikácia, ktorú je tiež možné využiť na posielanie RESTCONF GET a PUT požiadaviek na ODL kontrolér, ktorý následne tieto požiadavky odosiela na OpenFlow prepínače. Spustenie nástroja realizuje príkaz **grunt**, ako je možné vidieť na obrázku 6.21:

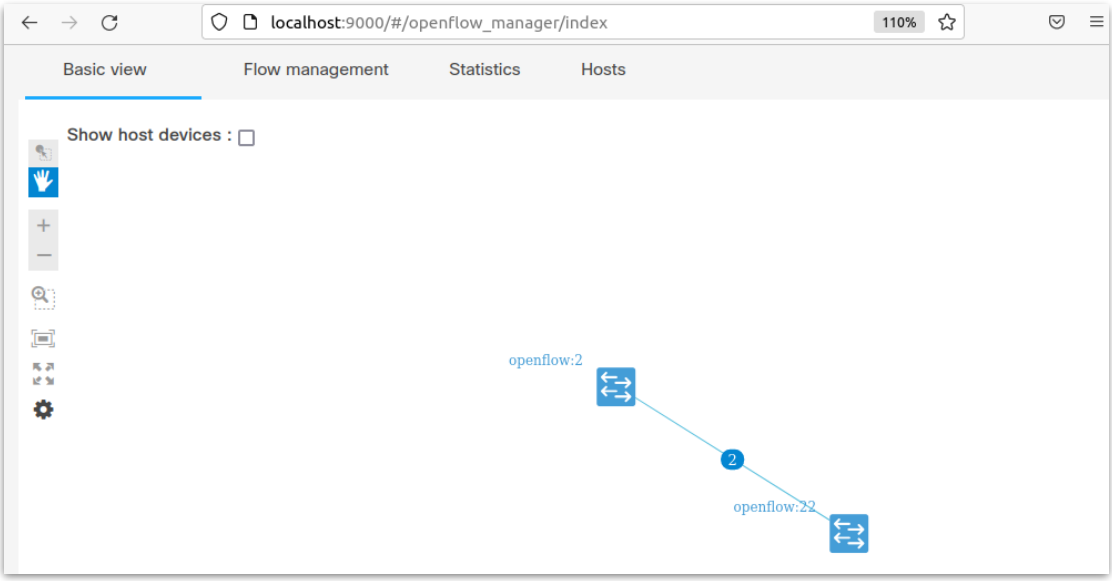
```
workshop5@workshop5-VirtualBox:~$ cd OpenDayLight-OpenFlow-App/
workshop5@workshop5-VirtualBox:~/OpenDayLight-OpenFlow-App$ grunt
Running "connect:def" (connect) task
Waiting forever...
Started connect web server on http://localhost:9000
```

Obrázok 6.21 Spustenie nástroja OFM

Tento nástroj predstavuje webovú aplikáciu, ktorú je možné otvoriť využitím nasledujúcej URL:

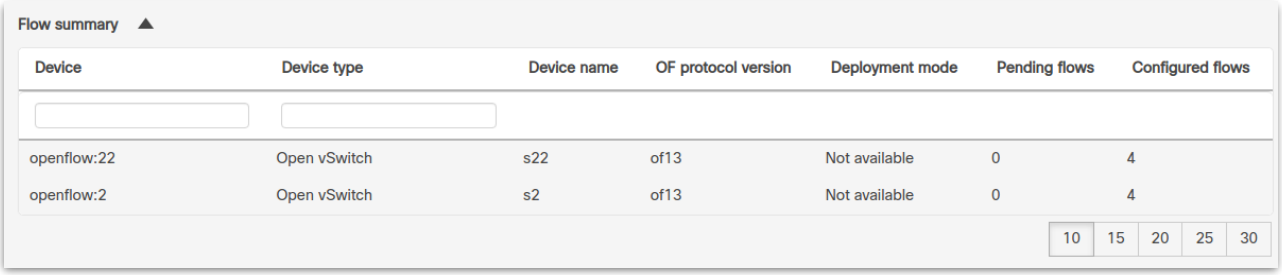
```
http://localhost:9000/#/openflow_manager/index
```

Grafické rozhranie webovej aplikácie vyzerá nasledovne (obrázok 6.22):



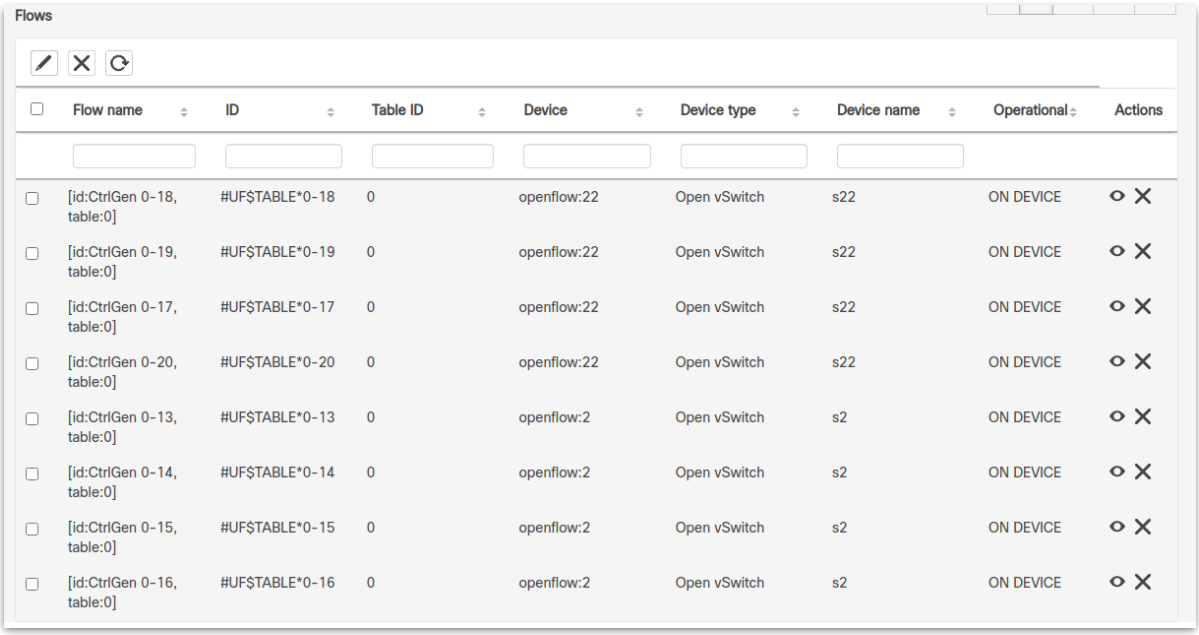
Obrázok 6.22 Zobrazenie spravovaných zariadení

Najdôležitejšou záložkou je záložka **Flow management**, cez ktorú je možné kompletne monitorovať a spravovať záznamy tokov. Po otvorení tejto záložky sa zobrazí sumárna informácia o spravovaných zariadeniach a počte existujúcich tokov (obrázok 6.23):



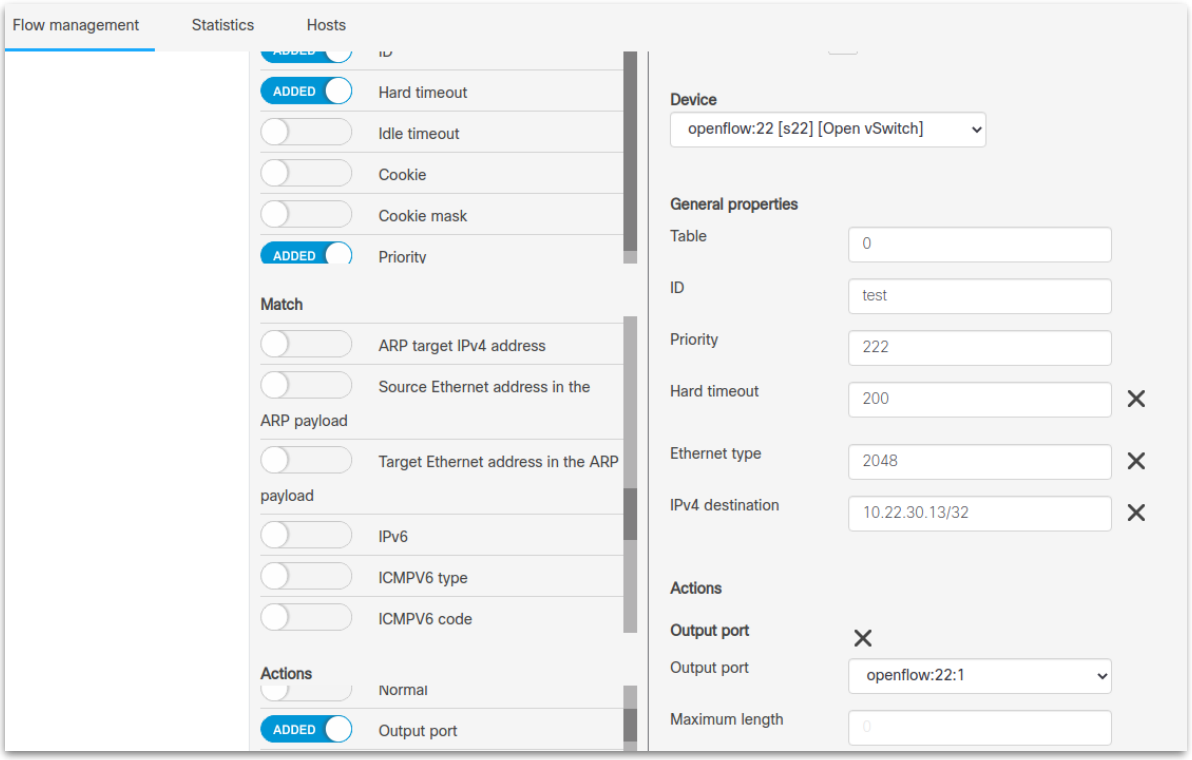
Obrázok 6.23 Zobrazenie tabuľky spravovaných zariadení

V spodnej časti tejto záložky je zoznam existujúcich tokov, ktoré je možné detailne zobraziť, upraviť prípadne vložiť úplne nový dátový tok. Obrázok 6.24 znázorňuje zoznam existujúcich tokov:



Obrázok 6.24 Zobrazenie tabuľky existujúcich tokov

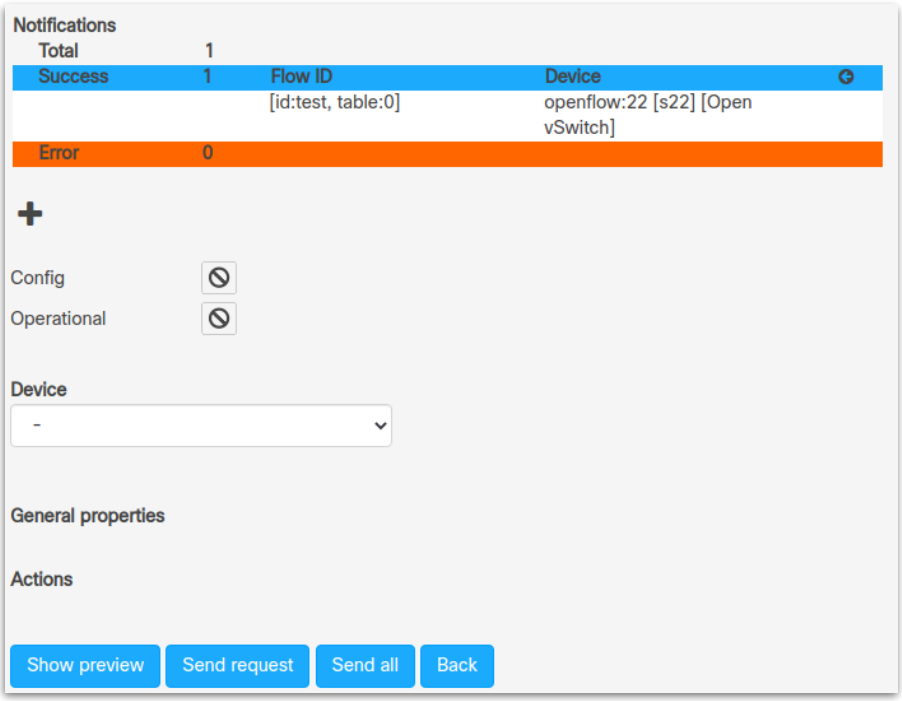
Po zobrazení náhľadu niektorého z tokov, sa zobrazí formulár, v ktorom je možné vidieť všetky atribúty toku. Pri vytváraní nového toku, alebo pri jeho editácii, je možné tieto atribúty meniť a upravený dátový tok odoslať na kontrolér:



Obrázok 6.25 Formulár pre úpravu atribútov dátového toku

V uvedenom formulári na obrázku 6.25, si v ľavej časti stačí jednoducho vyplniť nastavenia, ktoré chceme použiť a v pravej časti im následne zadať konkrétne požadované hodnoty.

Po odoslaní požiadavky sa v grafickom rozhraní zobrazí notifikačná správa o úspešnosti operácie, čo je možné vidieť na obrázku 6.26:



Obrázok 6.26 Potvrdenie úspešnosti vykonania operácie

Podobne ako z ODL Yang UI, tak aj toto rozhranie ponúka zobrazenie odoslanej požiadavky, kde sa po kliknutí na tlačidlo **show preview** zobrazí odoslaná štruktúra v JSON formáte, tak ako na obrázku 6.27:

Obrázok 6.27 Zobrazenie dátového toku v JSON formáte

Táto sekcia uzatvára možnosti správy OpenFlow prepínačov využitím kontroléra ODL a nástroja OFM. Avšak je potrebné si uvedomiť, že tieto nástroje sme museli ručne vyplniť. Ich jedinou výhodou je, že ponúkajú používateľsky prívetivé rozhranie. Nasledujúce sekcie sa venujú ukážke práce s nástrojmi, ktoré je možné viac prepojiť s inými automatizačnými prostriedkami.

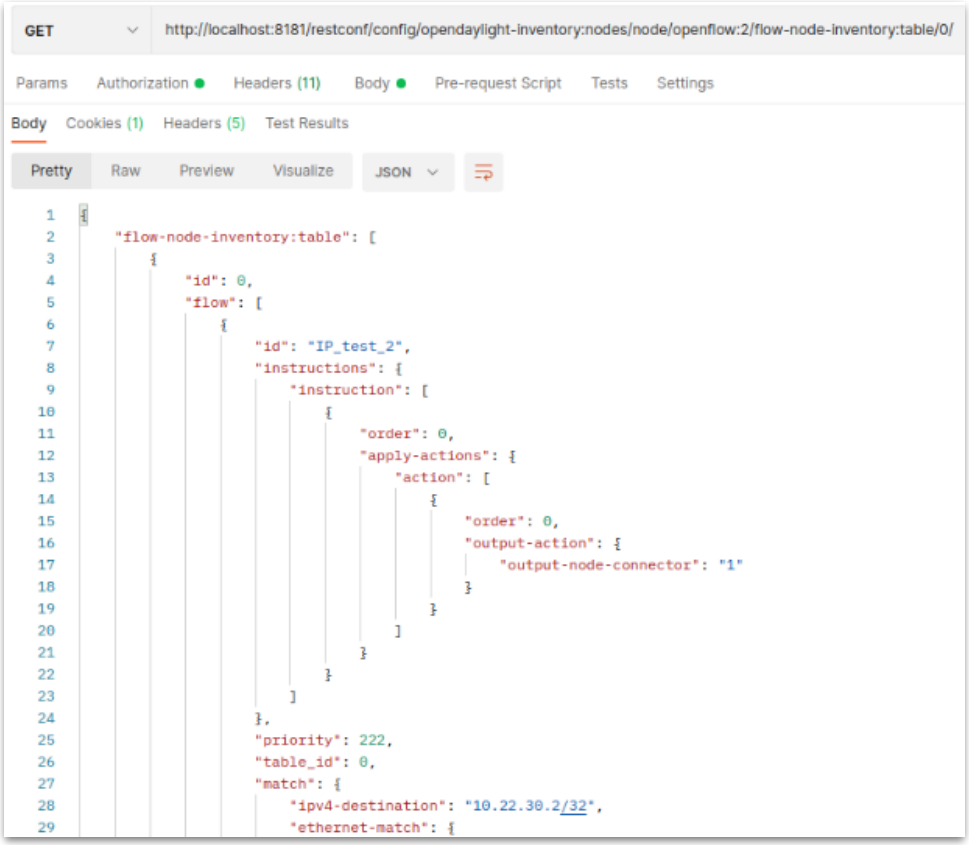
Úloha 5: Riadenie tokov – Postman

Nástroj Postman síce nepredstavuje automatizačný nástroj, no pri vzdelávacom procese je vhodným nástrojom pre začatie práce a zorientovanie sa v danej problematike. Týmto nástrojom je možné jednoducho vytvoriť HTTP požiadavku na základe údajov získaných z ODL a OFM. Výstupom sú dáta v JSON formáte, ktoré je neskôr možné využívať pri písaní kódu. Samotné rozhranie nástroja Postman zároveň umožňuje generovanie kódu pre tvorbu danej HTTP požiadavky pre rôzne programovacie jazyky. Nasledujúca sekcia znázorňuje ako jednoducho je možné vytvoriť a odoslať **GET** požiadavku. Jediným skrytým problémom môže byť autentifikácia. Tým, že sú požiadavky posielané na kontrolér ODL, tak autentifikačné údaje pri *basic Auth* type sú [admin, admin].

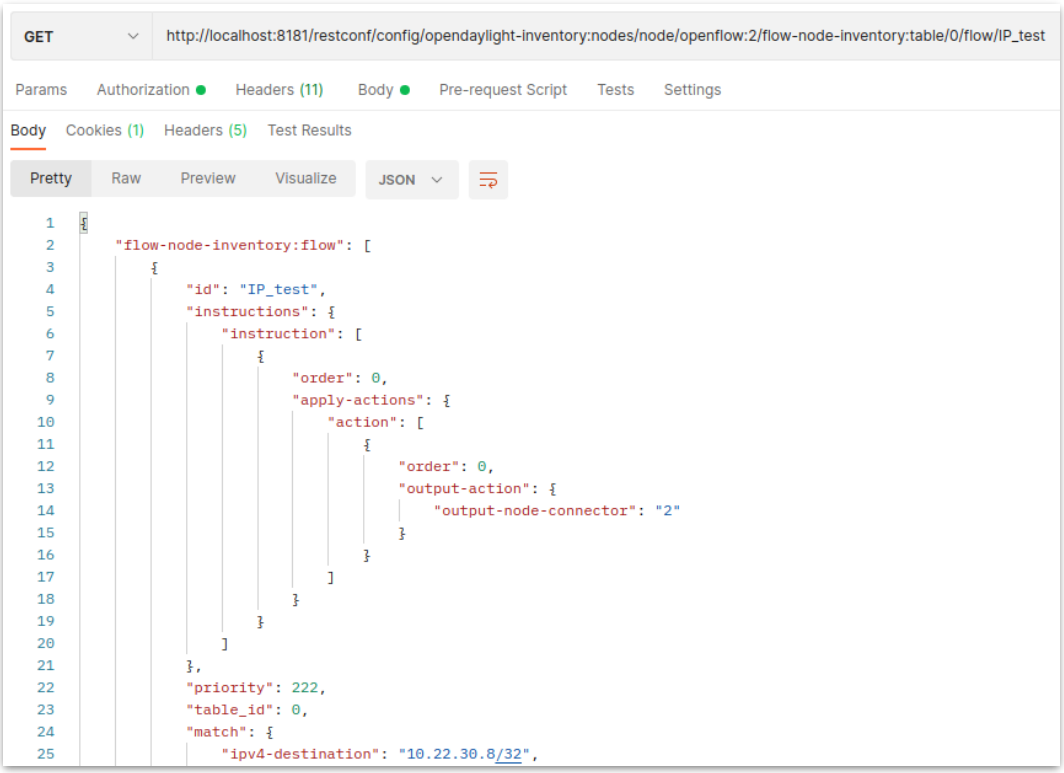
Nasledujúci výpis znázorňuje dve URL adresy, ktoré je možné využiť na načítanie celej tabuľky tokov a následne daný výpis filtrovať a zobraziť len údaje o jednom konkrétnom zvolenom toku:

```
http://10.10.10.18:8181/restconf/config/opendaylight-
inventory:nodes/node/openflow:2/flow-node-inventory:table/0/

http://10.10.10.18:8181/restconf/config/opendaylight-
inventory:nodes/node/openflow:2/flow-node-inventory:table/0/flow/IP_test
```

Pre programové spracovanie, pridávanie, mazanie a aktualizáciu záznamov o dátových tokoch je vhodnejšie využívať prístup k špecifickému toku. Tento prístup prináša efektívnejšie spracovanie, keďže nie je potrebné pracovať s veľkým množstvom dát, ale len s JSON štruktúrou pre jeden tok. Pri potrebe LEN načítavania dát je vhodnejší prístup stiahnutia celej tabuľky tokov (obrázok 6.28), z ktorej je možné zistiť názvy jednotlivých tokov, ktoré môžeme neskôr využiť na tvorbu filtrov (obrázok 6.29):

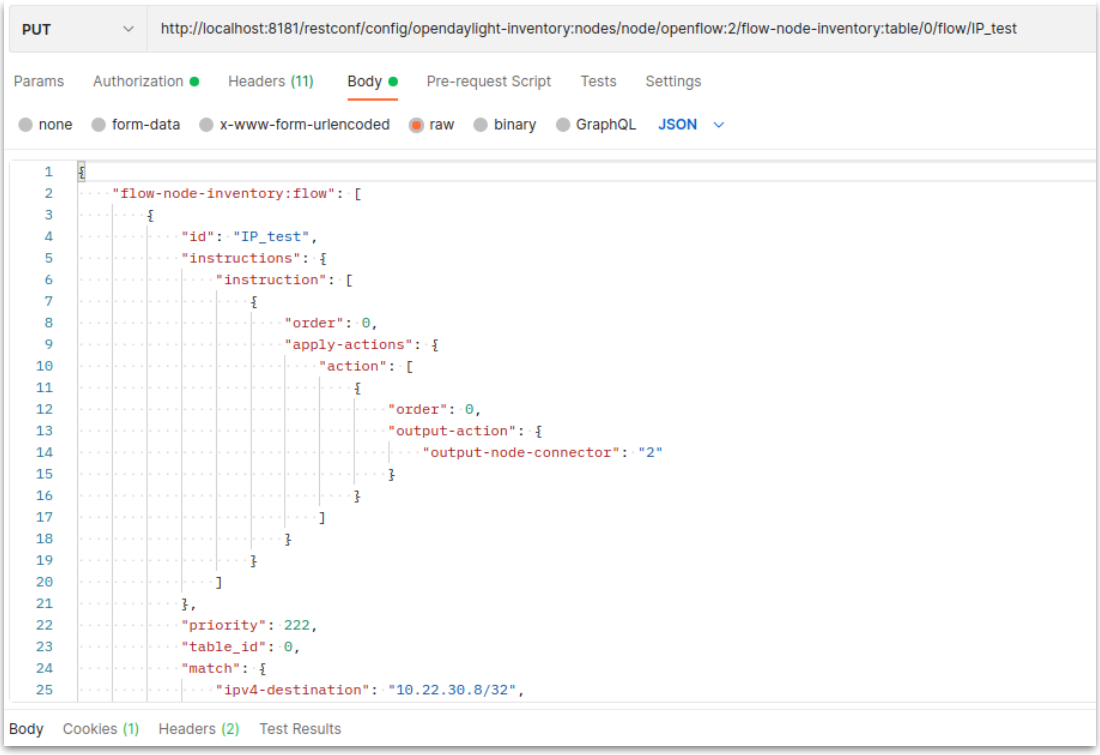


Obrázok 6.28 Zobrazenie výpisu tabuľky tokov v aplikácii Postman



Obrázok 6.29 Zobrazenie výpisu konkrétneho toku v aplikácii Postman

Pre vytvorenie/aktualizáciu záznamu o dátovom toku stačí zmeniť operáciu z GET na PUT a vyplniť pole **Body** (raw). Telo je ideálne vyplniť rovnakou JSON štruktúrou, akú sme dostali v predchádzajúcom kroku, pričom len mierne upravíme názov a URL (v prípade vytvárania nového toku). Zároveň je možné upraviť polia, ktoré chceme aktualizovať. Obrázok 6.30 znázorňuje vloženie toku s názvom `IP_test`:



Obrázok 6.30 Zmena konkrétneho toku aplikáciou Postman

Úloha 6: Riadenie tokov – Curl

Nástroj Curl je jednoduchý terminálový nástroj na odosielanie dát. Príklady formátu Curl požiadaviek z príkazového riadku pre HTTP GET (obrázok 6.31) a PUT metódy sú znázornené v nasledujúcich výpisoch:

```
curl --location --request GET 'http://localhost:8181/restconf/config/opendaylight-
inventory:nodes/node/openflow:2/flow-node-inventory:table/0/flow/IP_test' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic YWRtaW46YWRtaW4='
```

```
workshop5@workshop5-VirtualBox:~$ curl --location --request GET 'http://localhost:8181/restcon
f/config/opendaylight-inventory:nodes/node/openflow:2/flow-node-inventory:table/0/flow/IP_test
' \
> --header 'Content-Type: application/json' \
> --header 'Authorization: Basic YWRtaW46YWRtaW4='
{"flow-node-inventory:flow":[{"id":"IP_test","instructions":{"instruction":[{"order":0,"apply-
actions":{"action":[{"order":0,"output-action":{"output-node-connector":"2"}}]}}}]},"priority":
222,"table_id":0,"match":{"ipv4-destination":"10.22.30.8/32","ethernet-match":{"ethernet-type"
:{"type":2048}}}}]}workshop5@workshop5-VirtualBox:~$
```

Obrázok 6.31 Vykonanie GET požiadavky nástrojom Curl

```
curl --location --request PUT 'http://localhost:8181/restconf/config/opendaylight-
inventory:nodes/node/openflow:2/flow-node-inventory:table/0/flow/IP_test' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic YWRtaW46YWRtaW4=' \
--data-raw 'Tu vlozit JSON kod'
```

Úloha 7: Riadenie tokov – Python skript

V rámci programovacieho jazyka Python je na prácu s OpenFlow protokolom možné využiť knižnicu `requests`, ktorá umožňuje posielať RESTCONF požiadavky na ODL kontrolér. Vzorový zdrojový kód pre čítanie (metóda **GET**) dátového toku s názvom IP_test je uvedený v nasledujúcom výpise:

```
import requests
import json

url = "http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:2/flow-
node-inventory:table/0/flow/IP_test"

payload = ""
headers = {
  'Content-Type': 'application/json',
  'Authorization': 'Basic YWRtaW46YWRtaW4='
}
response = requests.request("GET", url, headers=headers, data=payload)
print(response.text)
```

Očakávaný výstup programu je zobrazený na obrázku 6.32:

```
workshop5@workshop5-VirtualBox:~$ python3 test_GET.py
{"flow-node-inventory:flow":[{"id":"IP_test","instructions":{"instruction":[{"order":0,"apply-actions":{"action":[{"order":0,
"output-action":{"output-node-connector":"2"}}]}}}]},"priority":222,"table_id":0,"match":{"ipv4-destination":"10.22.30.8/32","
ethernet-match":{"ethernet-type":{"type":2048}}}}]}
```

Obrázok 6.32 Zobrazenie výstupu pre vytvorený skript

Pre operáciu **PUT** sa kód veľmi nemení. Ukážka vzorového kódu je v nasledujúcom výpise:

```
import requests
import json

url = "http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:2/flow-
node-inventory:table/0/flow/IP_test"

payload = json.dumps({Tu vlozit flow JSON objekt})
headers = {
  'Content-Type': 'application/json',
  'Authorization': 'Basic YWRtaW46YWRtaW4=',
  'Cookie': 'JSESSIONID=1t8zf9r4j64vc1fzgl2p7cvuoz'
}
response = requests.request("PUT", url, headers=headers, data=payload)
print(response)
```

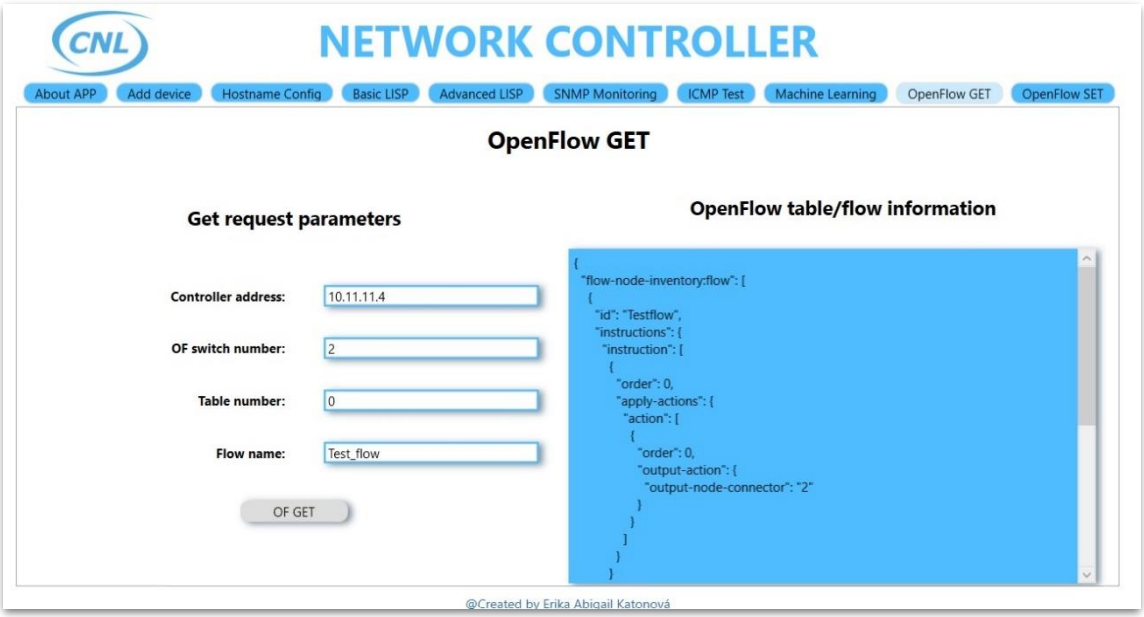
Očakávaný výstup programu je zobrazený na obrázku 6.33:

```
workshop5@workshop5-VirtualBox:~$ python3 test_PUT.py
<Response [200]>
```

Obrázok 6.33 Zobrazenie výstupu pre vytvorený skript

Úloha 8: Riadenie tokov – C# (desktopová aplikácia)

Táto záverečná sekcia ukáže, akým spôsobom je možné do kontroléra, ktorý je implementovaný v podobe desktopovej aplikácie v jazyku C#, technológiou .NET a rámcom WPF, pridať podporu pre prácu s protokolom OpenFlow. Používateľské rozhranie pre operáciu GET by mohlo vyzerať nasledovne (obrázok 6.34):



Obrázok 6.34 Vzorové používateľské rozhranie po vykonaní GET požiadavky

Ako je možné vidieť, tak v ľavej časti používateľ špecifikuje adresu kontroléra, číslo OpenFlow prepínača, číslo tabuľky a voliteľne názov dátového toku. V pravej časti sa následne vypíše prijatý JSON objekt. Pri špecifikácii názvu toku, sa vypíše JSON pre konkrétny tok, ak názov toku špecifikovaný nebol, tak sa vypíše JSON pre celú tabuľku, ktorá obsahuje záznam o každom dátovom toku.

Nasledujúce výpisy obsahujú zdrojový kód pre vytvorenie používateľského rozhrania:

```
<TabItem Header="OpenFlow GET">
   <Grid>
      <Grid.RowDefinitions>
         <RowDefinition Height="1*"/>
         <RowDefinition Height="8*"/>
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
         <ColumnDefinition Width="1*"/>
         <ColumnDefinition Width="1*"/>
      </Grid.ColumnDefinitions>
      <Label Content="OpenFlow GET" Grid.Row="0" Grid.ColumnSpan="2" HorizontalAlignment="Center"
VerticalAlignment="Center" FontWeight="Bold" FontSize="22"/>

      <Grid Grid.Row="1" Grid.Column="0">
         <Grid.RowDefinitions>
            <RowDefinition Height="2*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="2*"/>
         </Grid.RowDefinitions>
         <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
         </Grid.ColumnDefinitions>
         <Label Content="Get request parameters" Grid.Row="0" Grid.ColumnSpan="2"
HorizontalAlignment="Center" VerticalAlignment="Center" FontWeight="Bold" FontSize="18"/>

         <Label Content="Controller address: " Grid.Row="1" Grid.Column="0"
HorizontalAlignment="Right" VerticalAlignment="Center" FontWeight="Bold"/>
         <TextBox Name="of_controller_address" Grid.Row="1" Grid.Column="1" Height="20" Width
="200" HorizontalAlignment="Center" VerticalAlignment="Center" BorderBrush="#4ebcff"
BorderThickness="2">
            <TextBox.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </TextBox.Effect>
         </TextBox>

         <Label Content="OF switch number: " Grid.Row="2" Grid.Column="0"
HorizontalAlignment="Right" VerticalAlignment="Center" FontWeight="Bold"/>
         <TextBox Name="of_node_number" Grid.Row="2" Grid.Column="1" Height="20" Width ="200"
HorizontalAlignment="Center" VerticalAlignment="Center" BorderBrush="#4ebcff" BorderThickness="2">
            <TextBox.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </TextBox.Effect>
         </TextBox>

         <Label Content="Table number: " Grid.Row="3" Grid.Column="0" HorizontalAlignment="Right"
VerticalAlignment="Center" FontWeight="Bold"/>
         <TextBox Name="of_table_number" Grid.Row="3" Grid.Column="1" Height="20" Width ="200"
HorizontalAlignment="Center" VerticalAlignment="Center" BorderBrush="#4ebcff" BorderThickness="2">
            <TextBox.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </TextBox.Effect>
         </TextBox>

         <Label Content="Flow name: " Grid.Row="4" Grid.Column="0" HorizontalAlignment="Right"
VerticalAlignment="Center" FontWeight="Bold"/>
         <TextBox Name="of_flow_name" Grid.Row="4" Grid.Column="1" Height="20" Width ="200"
HorizontalAlignment="Center" VerticalAlignment="Center" BorderBrush="#4ebcff" BorderThickness="2">
            <TextBox.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </TextBox.Effect>
         </TextBox>

         <Button Content="OF GET" Grid.Row="5" Grid.ColumnSpan="2" Height="20" Width="100"
BorderThickness="0" HorizontalAlignment="Center" VerticalAlignment="Top" Click="OpenFlow_Get"
Margin="0,20,0,0">
            <Button.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
```

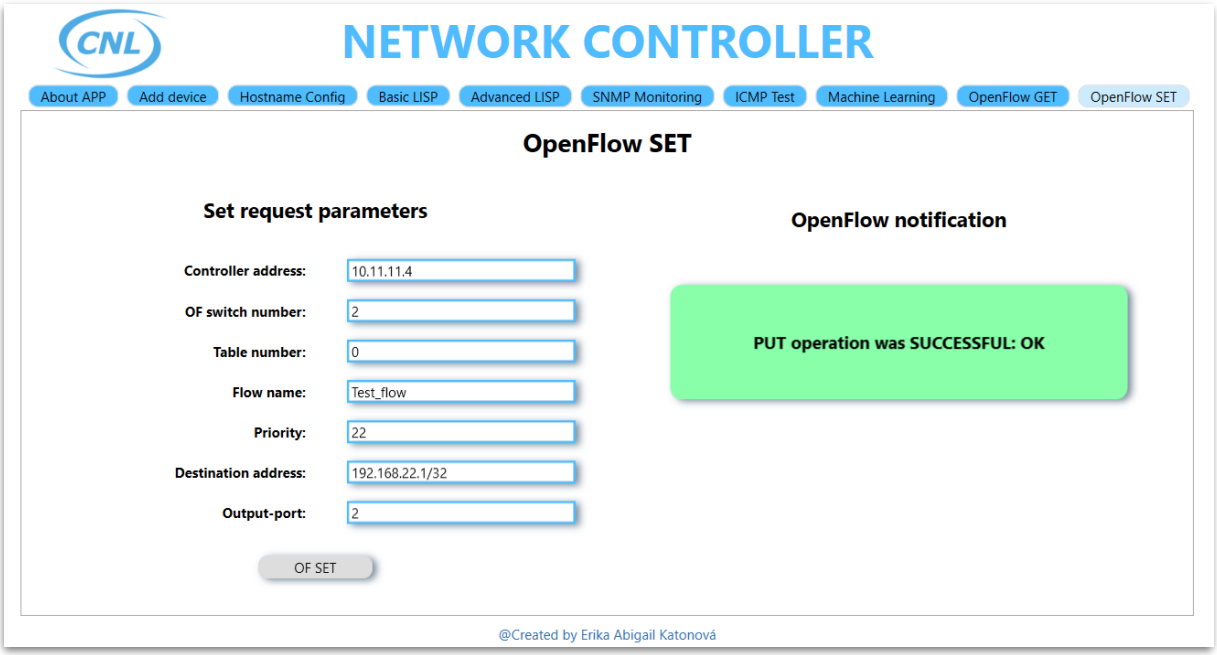
```
            </Button.Effect>
            <Button.Resources>
               <Style TargetType="Border">
                  <Setter Property="CornerRadius" Value="8"/>
               </Style>
            </Button.Resources>
         </Button>
      </Grid>

      <Grid Grid.Row="1" Grid.Column="1">
         <Grid.RowDefinitions>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="4*"/>
         </Grid.RowDefinitions>
         <Label Content="OpenFlow table/flow information" Grid.Row="0" HorizontalAlignment="Center"
VerticalAlignment="Center" FontWeight="Bold" FontSize="18"/>

         <ScrollViewer Grid.Row="3" Grid.ColumnSpan="2" VerticalScrollBarVisibility="Auto"
Margin="0,0,20,0">
            <StackPanel Name="OF_Get_show" Background="#ccebff"/>
            <ScrollViewer.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </ScrollViewer.Effect>
            <ScrollViewer.Resources>
               <Style TargetType="Border">
                  <Setter Property="CornerRadius" Value="10"/>
               </Style>
            </ScrollViewer.Resources>
         </ScrollViewer>
      </Grid>
   </Grid>
</TabItem>
```

Zdrojový kód pre vykonanie HTTP GET požiadavky a výpis je znázornený v nasledujúcom výpise:

```
private void OpenFlow_Get(object sender, RoutedEventArgs e) {
   //vyčistenie poľa
   OF_Get_show.Children.Clear();

   //definovanie parametrov načítaných od používateľa
   string address = of_controller_address.Text;
   string device_number = of_node_number.Text;
   string table_number = of_table_number.Text;
   string flow_name = of_flow_name.Text;

   //vytvorenie URL podľa zadaného vstupu
   string url;
   //ak bol zadaný aj názov dátového toku tak daná URL obsahuje doplnok o dátový tok
   if (!string.IsNullOrWhiteSpace(flow_name)) {
      url = "http://" + address + ":8181/restconf/config/opendaylight-
inventory:nodes/node/openflow:" + device_number + "/flow-node-inventory:table/" + table_number +
"/flow/" + flow_name;
   }
   //ak nebol zadaný názov dátového toku, tak vygenerovaná URL identifikuje všetky dátové toky
   else {
      url = "http://" + address + ":8181/restconf/config/opendaylight-
inventory:nodes/node/openflow:" + device_number + "/flow-node-inventory:table/" + table_number;
   }
   //v prípade zle vyplnených údajov pre URL môže dôjsť k chybe pri vytváraní HTTP komunikácie
   try {
      //vytvorenie HTTP požiadavky
      var client = new RestClient(url);
      client.Timeout = -1;
      var request = new RestRequest(Method.GET);
      //nastavenie hlavičiek pre nastavenie formátu prenášaných dát a autentifikáciu (Base64 pre
meno: admin, heslo: admin)
      request.AddHeader("Content-Type", "application/json");
      request.AddHeader("Authorization", "Basic YWRtaW46YWRtaW4=");

      //odoslanie HTTP požiadavky
      IRestResponse response = client.Execute(request);
```

```
        //podmienka overujúca úspešnosť prijatej odpovede - status kód 2xx
        if (response.IsSuccessful) {
           //deserializácia prijatých dát do podoby JSON objektu
           JObject obj = Newtonsoft.Json.JsonConvert.DeserializeObject<JObject>(response.Content);
          //vyplnenie obsahu elementu Label prijatou odpoveďou v okne aplikácie a zvýraznenie modrou
          Label OF_label = new();
          OF_label.Content = obj.ToString();
          OF_label.Background=(SolidColorBrush)new BrushConverter().ConvertFromString("#4EBCFF");
          OF_Get_show.Children.Add(OF_label);
        }
        else {
        //vyplnenie obsahu Label chybovou hláškou v okne aplikácie a zvýraznenie chyby červenou
           Label OF_label = new();
           OF_label.Content = "Bad response";
           OF_label.Background=(SolidColorBrush)new BrushConverter().ConvertFromString("#f99b9b");
           OF_Get_show.Children.Add(OF_label);
        }
     }
     catch {
        //vyplnenie obsahu Label chybovou hláškou v okne aplikácie a zvýraznenie chyby červenou
        Label OF_label = new();
        OF_label.Content = "HTTP communication FAILED";
        OF_label.Background=(SolidColorBrush)new BrushConverter().ConvertFromString("#f99b9b");
        OF_Get_show.Children.Add(OF_label);
     }
  }
```

V prípade operácie SET je potrebné, aby mal používateľ možnosť zadať viacero parametrov, nie len tie pre nadviazanie spojenia, ale aj tie, špecifikujúce identifikačný atribút a akciu, ktorá sa vykoná. Vzorové rozhranie by mohlo vyzerať nasledovne (obrázok 5.35):



Obrázok 6.35 Vzorové používateľské rozhranie po vykonaní PUT operácie

Ľavá časť používateľského rozhrania obsahuje sekciu, pre vloženie údajov. Pravá časť obsahuje sekciu pre výpis notifikácie o úspešnosti vykonania operácie.

Zdrojový kód používateľského rozhrania je zobrazený v nasledujúcich výpisoch:

```
<TabItem Header="OpenFlow SET">
   <Grid>
      <Grid.RowDefinitions>
         <RowDefinition Height="1*"/>
         <RowDefinition Height="8*"/>
      </Grid.RowDefinitions>
```

```
      <Grid.ColumnDefinitions>
         <ColumnDefinition Width="1*"/>
         <ColumnDefinition Width="1*"/>
      </Grid.ColumnDefinitions>
      <Label Content="OpenFlow SET" Grid.Row="0" Grid.ColumnSpan="2" HorizontalAlignment="Center"
VerticalAlignment="Center" FontWeight="Bold" FontSize="22"/>

      <Grid Grid.Row="1" Grid.Column="0">
         <Grid.RowDefinitions>
            <RowDefinition Height="2*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="2*"/>
         </Grid.RowDefinitions>
         <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
         </Grid.ColumnDefinitions>
         <Label Content="Set request parameters" Grid.Row="0" Grid.ColumnSpan="2"
HorizontalAlignment="Center" VerticalAlignment="Center" FontWeight="Bold" FontSize="18"/>

         <Label Content="Controller address: " Grid.Row="1" Grid.Column="0"
HorizontalAlignment="Right" VerticalAlignment="Center" FontWeight="Bold"/>

         <TextBox Name="of_set_controller_address" Grid.Row="1" Grid.Column="1" Height="20" Width
="200" HorizontalAlignment="Center" VerticalAlignment="Center" BorderBrush="#4ebcff"
BorderThickness="2">
            <TextBox.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </TextBox.Effect>
         </TextBox>


         <Label Content="OF switch number: " Grid.Row="2" Grid.Column="0"
HorizontalAlignment="Right" VerticalAlignment="Center" FontWeight="Bold"/>
         <TextBox Name="of_set_node_number" Grid.Row="2" Grid.Column="1" Height="20" Width ="200"
HorizontalAlignment="Center" VerticalAlignment="Center" BorderBrush="#4ebcff" BorderThickness="2">
            <TextBox.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </TextBox.Effect>
         </TextBox>


         <Label Content="Table number: " Grid.Row="3" Grid.Column="0" HorizontalAlignment="Right"
VerticalAlignment="Center" FontWeight="Bold"/>
         <TextBox Name="of_set_table_number" Grid.Row="3" Grid.Column="1" Height="20" Width ="200"
HorizontalAlignment="Center" VerticalAlignment="Center" BorderBrush="#4ebcff" BorderThickness="2">
            <TextBox.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </TextBox.Effect>
         </TextBox>

         <Label Content="Flow name: " Grid.Row="4" Grid.Column="0" HorizontalAlignment="Right"
VerticalAlignment="Center" FontWeight="Bold"/>
         <TextBox Name="of_set_flow_name" Grid.Row="4" Grid.Column="1" Height="20" Width ="200"
HorizontalAlignment="Center" VerticalAlignment="Center" BorderBrush="#4ebcff" BorderThickness="2">
            <TextBox.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </TextBox.Effect>
         </TextBox>

         <Label Content="Priority: " Grid.Row="5" Grid.Column="0" HorizontalAlignment="Right"
VerticalAlignment="Center" FontWeight="Bold"/>
         <TextBox Name="of_set_priority" Grid.Row="5" Grid.Column="1" Height="20" Width ="200"
HorizontalAlignment="Center" VerticalAlignment="Center" BorderBrush="#4ebcff" BorderThickness="2">
            <TextBox.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </TextBox.Effect>
         </TextBox>
```

```
         <Label Content="Destination address: " Grid.Row="6" Grid.Column="0"
HorizontalAlignment="Right" VerticalAlignment="Center" FontWeight="Bold"/>
         <TextBox Name="of_set_d_add" Grid.Row="6" Grid.Column="1" Height="20" Width ="200"
HorizontalAlignment="Center" VerticalAlignment="Center" BorderBrush="#4ebcff" BorderThickness="2">
            <TextBox.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </TextBox.Effect>
         </TextBox>

         <Label Content="Output-port: " Grid.Row="7" Grid.Column="0" HorizontalAlignment="Right"
VerticalAlignment="Center" FontWeight="Bold"/>
         <TextBox Name="of_set_output_port" Grid.Row="7" Grid.Column="1" Height="20" Width ="200"
HorizontalAlignment="Center" VerticalAlignment="Center" BorderBrush="#4ebcff" BorderThickness="2">
            <TextBox.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </TextBox.Effect>
         </TextBox>

         <Button Content="OF SET" Grid.Row="8" Grid.ColumnSpan="2" Height="20" Width="100"
BorderThickness="0" HorizontalAlignment="Center" VerticalAlignment="Top" Click="OpenFlow_Set"
Margin="0,20,0,0">
            <Button.Effect>
              <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
            </Button.Effect>
            <Button.Resources>
               <Style TargetType="Border">
                  <Setter Property="CornerRadius" Value="8"/>
               </Style>
            </Button.Resources>
         </Button>
      </Grid>

      <Grid Grid.Row="1" Grid.Column="1">
         <Grid.RowDefinitions>
            <RowDefinition Height="2*"/>
            <RowDefinition Height="3*"/>
            <RowDefinition Height="4*"/>
         </Grid.RowDefinitions>
         <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
         </Grid.ColumnDefinitions>
         <Label Content="OpenFlow notification" Grid.Row="0" Grid.ColumnSpan="2"
HorizontalAlignment="Center" VerticalAlignment="Center" FontWeight="Bold" FontSize="18"/>

         <ScrollViewer Grid.Row="1" Grid.ColumnSpan="2" HorizontalScrollBarVisibility="Auto"
VerticalScrollBarVisibility="Auto">
            <Label Name="OF_Set_show" Content="" Width="400" Height="100" Background="#ccebff"
FontWeight="Bold" HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
FontSize="15">
               <Label.Resources>
                  <Style TargetType="Border">
                     <Setter Property="CornerRadius" Value="10"/>
                  </Style>
               </Label.Resources>
               <Label.Effect>
                 <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4"
BlurRadius="10"/>
               </Label.Effect>
            </Label>
         </ScrollViewer>
      </Grid>
   </Grid>
</TabItem>
```

Kód pre odoslanie PUT požiadavky je uvedený v nasledujúcich výpisoch:

```
private void OpenFlow_Set(object sender, RoutedEventArgs e) {
   //definovanie premenných pre uloženie vstupu od používateľa
   string address = of_set_controller_address.Text;
   string device = of_set_node_number.Text;
   string table = of_set_table_number.Text;
   string flow_name = of_set_flow_name.Text;
   string priority = of_set_priority.Text;
   string dst_add = of_set_d_add.Text;
   string out_port = of_set_output_port.Text;

   //sekcia spravujúca vznik chybových udalostí - zlé vyplnenie údajov, chyba pri HTTP komunikácii
   try {
      //vytvorenie HTTP požiadavky
      var client = new RestClient("http://" + address + ":8181/restconf/config/opendaylight-
inventory:nodes/node/openflow:" + device + "/flow-node-inventory:table/" + table + "/flow/" +
flow_name);
      client.Timeout = -1;
      var request = new RestRequest(Method.PUT);
      //nastavenie hlavičiek pre definovanie formátu transportovaných dát a pre autentifikáciu
(Base64 na základe mena: admin, hesla: admin)
      request.AddHeader("Content-Type", "application/json");
      request.AddHeader("Authorization", "Basic YWRtaW46YWRtaW4=");

      //vytvorenie nového JObject objektu pre uloženie opisu záznamu o dátovom toku, ktorého
štruktúru definuje YANG model
      JObject flow = new(
         new JProperty("flow-node-inventory:flow",
            new JArray(
               new JObject(
                  new JProperty("id", flow_name),
                  new JProperty("instructions",
                     new JObject(
                        new JProperty("instruction",
                           new JArray(
                              new JObject(
                                 new JProperty("order", 0),
                                 new JProperty("apply-actions",
                                    new JObject(
                                       new JProperty("action",
                                          new JArray(
                                             new JObject(
                                                new JProperty("order", 0),
                                                new JProperty("output-action",
                                                   new JObject(
                                                      new JProperty("output-node-connector",
out_port)))))))))))),
                  new JProperty("match",
                     new JObject(
                        new JProperty("ipv4-destination", dst_add),
                        new JProperty("ethernet-match",
                           new JObject(
                              new JProperty("ethernet-type",
                                 new JObject(
                                    new JProperty("type", 2048)))))),
                  new JProperty("hard-timeout", 0),
                  new JProperty("priority", Int32.Parse(priority)),
                  new JProperty("table_id", Int32.Parse(table)),
                  new JProperty("idle-timeout", 0)))));

      //serializácia JObject dátového toku do podoby JSON string-u, ktorý je možné odoslať
protokolom RESTCONF
      string body = Newtonsoft.Json.JsonConvert.SerializeObject(flow);
      //pridanie vytvoreného tela správy do HTTP požiadavky
      request.AddParameter("application/json", body, ParameterType.RequestBody);
      //odoslanie HTTP požiadavky
      IRestResponse response = client.Execute(request);
      //overenie úspešnosti vykonania metódy PUT
      if (response.IsSuccessful) {
         //výpis úspešnosti vykonania požiadavky do okna aplikácie a zvýraznenie výpisu modrou
         OF_Set_show.Content = "PUT operation was SUCCESSFUL: " + response.StatusCode.ToString();
         OF_Set_show.Background=(SolidColorBrush)new BrushConverter().ConvertFromString("#89ffaa");
      }
```

```
        else {
            //výpis notifikácie o neúspešnom vykonaní PUT operácie a zvýraznenie výpisu červenou
            OF_Set_show.Content = "PUT operation FAILED: " + response.StatusCode.ToString();
            OF_Set_show.Background=(SolidColorBrush)new BrushConverter().ConvertFromString("#f99b9b");
        }
    }
    catch {
        //výpis notifikácie o chybe HTTP komunikácie a zvýraznenie výpisu červenou farbou
        OF_Set_show.Content = "HTTP communication FAILED";
        OF_Set_show.Background=(SolidColorBrush)new BrushConverter().ConvertFromString("#f99b9b");
    }
}
```

Protokol OpenFlow – ODL kontrolér

Úvod

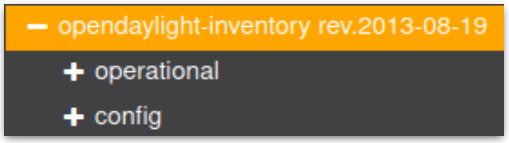
Táto kapitola predstavuje doplnok k predchádzajúcej, pričom cieľom je názorne ukázať tvorbu dátových tokov pre OpenFlow prepínače, využitím webového grafického rozhrania kontroléra OpenDayLight. Motiváciou je hlavne poukázať na reprezentáciu a formu zápisu jednotlivých atribútov použitých pre špecifikáciu dátového toku, ktoré sa jemne líšia od manuálneho vloženia cez terminál.

Požiadavka GET

Na čítanie informácií o vytvorených dátových tokoch stačí zvoliť správny YANG model a vnoriť sa až do úrovne, ktorá reprezentuje príslušnú tabuľku tokov. YANG modely je možné nájsť v záložke **Yang UI**. Model, reprezentujúci konfiguráciu a stav zariadenia, ktorý bude použitý sa nazýva:

```
opendaylight-inventory rev.2013-08-19
```

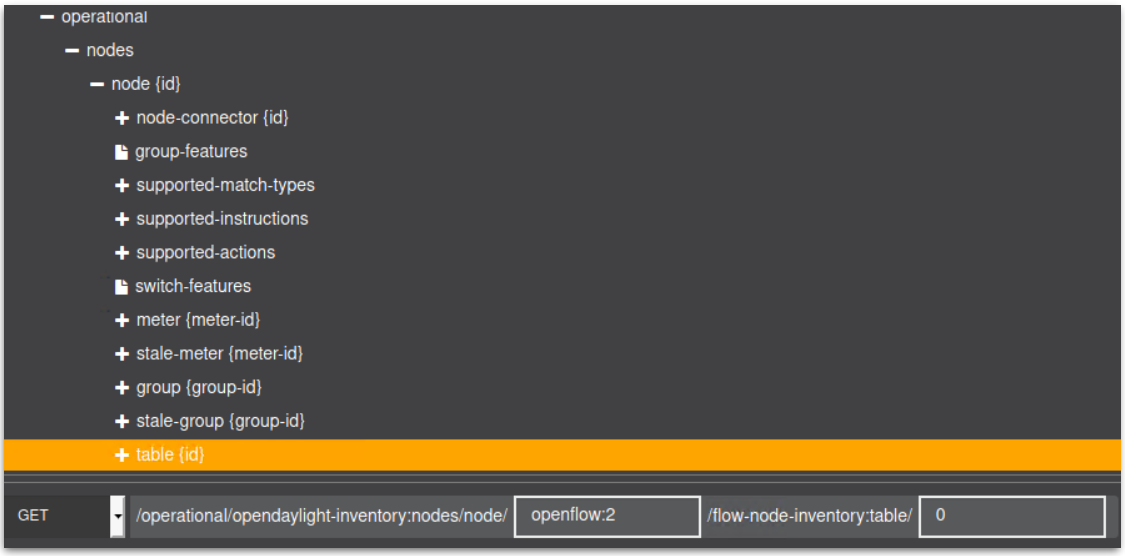
Po zobrazení modelu je možné vidieť (obrázok 7.1), že obsahuje časť pre operačné a časť pre konfiguračné dáta:



Obrázok 7.1 Základná štruktúra modulu inventory

V tejto časti je najskôr ukázané ako pracovať s operačnými dátami, cez ktoré je možné čítať informácie o danom zariadení a o dátových tokoch, ktoré sú na ňom vytvorené. Záložka **operational** teda umožňuje vykonanie len jedinej metódy a to metódy **GET**.

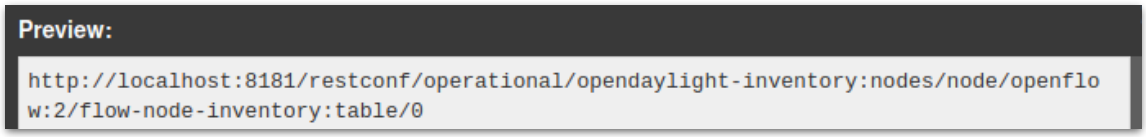
Záložku **operational** je možné postupne rozbaľovať a vnárať sa k čo najšpecifickejším častiam YANG modelu. Príklad pre zobrazenie všetkých tokov definovaných na prepínači **S2**, ktoré sa nachádzajú v tabuľke **0** znázorňuje obrázok 7.2:



Obrázok 7.2 Definovanie URL adresy pre prístup k tabuľke 0

Dôležité si je všimnúť, že názov zariadenia je potrebné špecifikovať v tvare: **openflow:x**, kde **x** predstavuje číslo zariadenia. Po stlačení tlačidla **Send** dôjde k odoslaniu **GET** požiadavky využitím protokolu **RESTCONF** smerom ku kontroléru, ktorý následne danú požiadavku pretransformuje do protokolu **OpenFlow** a odošle smerom k spravovaným zariadeniam.

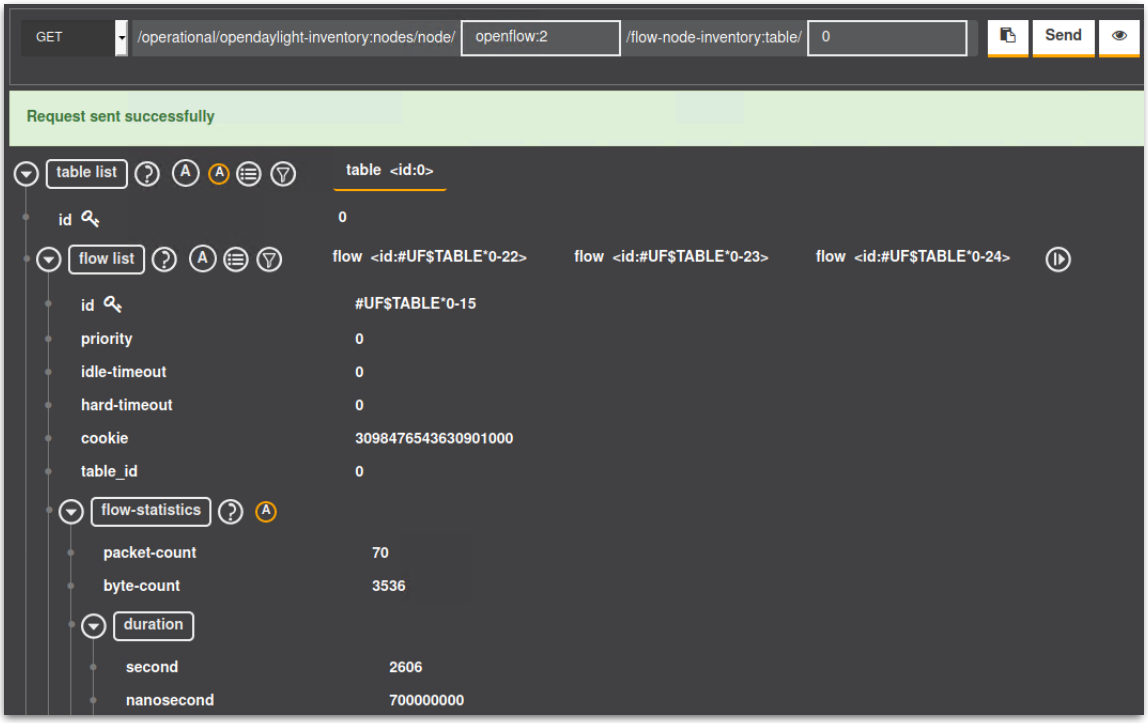
Užitočným tlačidlom je tiež **Show Preview**, ktoré zobrazí vygenerovanú URL (obrázok 7.3), na ktorú sa daná **GET** požiadavka odošle. Táto URL špecifikuje zdroj, ktorý sme identifikovali v štruktúre YANG modelu:



Obrázok 7.3 Zobrazenie vytvorenej URL adresy

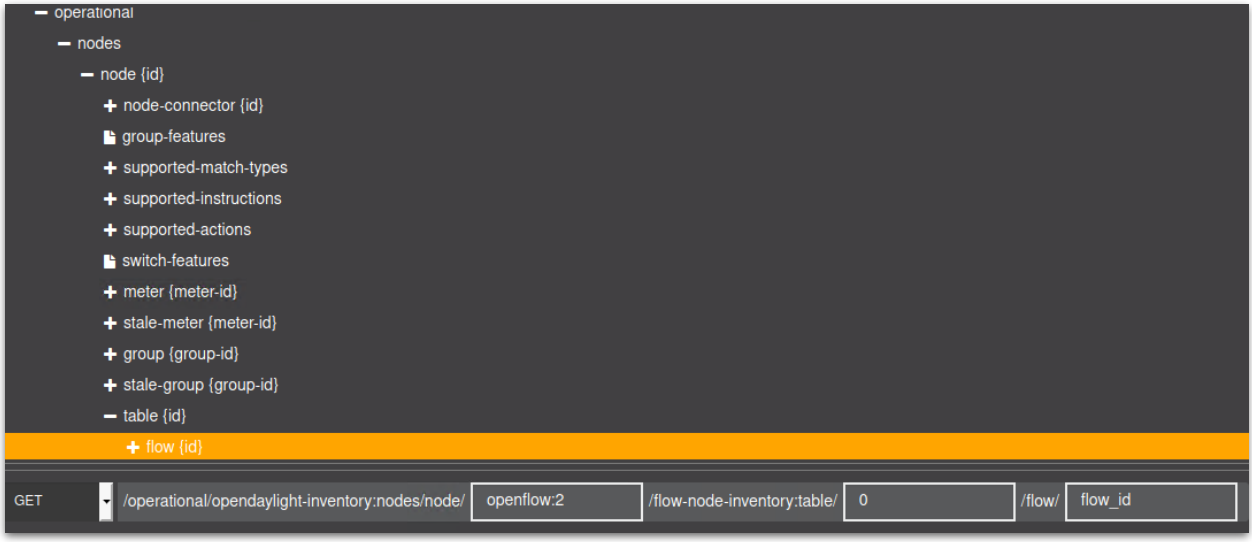
Po prijatí odpovede, sú prijaté dáta, v stromovej štruktúre, zobrazené v spodnej časti okna, kde sa zobrazia požadované zdroje (v našom prípade definované dátové toky), a k nim ich základné údaje ako napr. ID, priorita, počet prenesených správ, celkové trvanie prenosu atď.

Obrázok 7.4 znázorňuje vzorovú ukážku prijatej odpovede:



Obrázok 7.4 Odoslanie GET požiadavky

Ako je možné vidieť, tak uvedený výpis znázorňuje definíciu všetkých dátových tokov. Ak by bolo potrebné prijaté dáta ďalej filtrovať a zobraziť len jeden dátový tok, tak stačí rozbaliť ponuku **Table** a v nej zvoliť **Flow**, kde následne po špecifikovaní ID daného toku dôjde k bližšiemu špecifikovaniu URL a tým k získaniu konkrétnych dát o danom hľadanom toku. Filtrovanie je vhodné hlavne neskôr pri interakcii s modelmi využitím programovacích jazykov, kedy je jednoduchšie pracovať so špecifickým modelom ako s rozsiahlou štruktúrou opisujúcou celý model. Obrázok 7.5 zobrazuje možnosť tvorby spomínaného filtra pre toky:

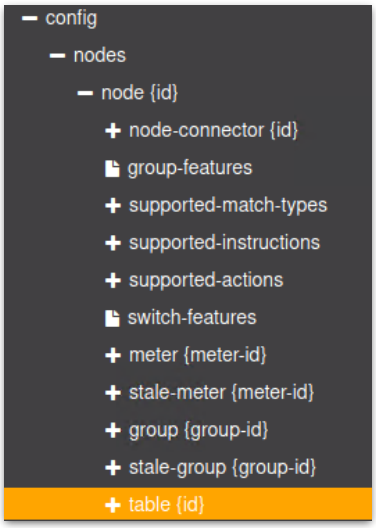


Obrázok 7.5 Definovanie URL adresy pre prístup k tabuľke 0 a dátovému toku flow_id

Ako je možné vidieť, grafické rozhranie umožňuje jednoduchým spôsobom nájsť tú správnu URL adresu, ktorá špecifikuje objekt nášho záujmu. Nasledujúca sekcia ukáže opačný prístup a to spôsob, ako vykonať zmenu v dátovom toku využitím grafického rozhrania.

Požiadavka PUT

YANG modelom opísané časti OpenFlow prepínačov je tiež možné upravovať využitím operácie **PUT**. Táto operácia je v rámci používaného YANG modelu dostupná pre sekciu **Config**. Podobne ako pri sekcii **Operational**, aj v tomto prípade sa je možné vnárať k špecifickejším častiam konfigurácie, vďaka čomu je následne možné pracovať s menšími dátovými štruktúrami (štandardne opísané v JSON formáte). Obrázok 7.6 znázorňuje hierarchiu konfigurácie OpenFlow prepínača:



Obrázok 7.6 Zobrazenie stromovej štruktúry záložky config

Konfigurovať je možné naraz všetky dátové toky, ktoré chceme vytvárať, prípadne je možné sa vnárať ďalej a konfigurovať len 1 konkrétny dátový tok. Grafické rozhranie umožňuje po zvolení konkrétneho objektu (dátový tok) nastaviť všetky požadované atribúty.

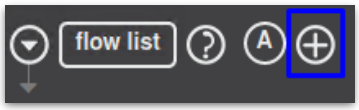
Príklad 1: Vytvorte dátový tok, ktorý na základe cieľovej MAC adresy (00:00:00:00:00:22) bude na prepínači **S22** preposielať dáta cez rozhranie **eth2** (príklad je vzhľadom k uvedenej topológii). Uistite sa, že dátový tok bude mať v tabuľke tokov **0** prioritu **22**, neexspiruje nikdy a bude mať názov **Poklad**. Vzorovú URL adresu pre identifikáciu daného objektu znázorňuje obrázok 7.7:

Obrázok 7.7 Vytvorenie URL adresy pre tabuľku 0 a dátového toku Poklad

V prípade manuálnej konfigurácie by vytvorenie požadovaného dátového toku vyzeralo nasledovne:

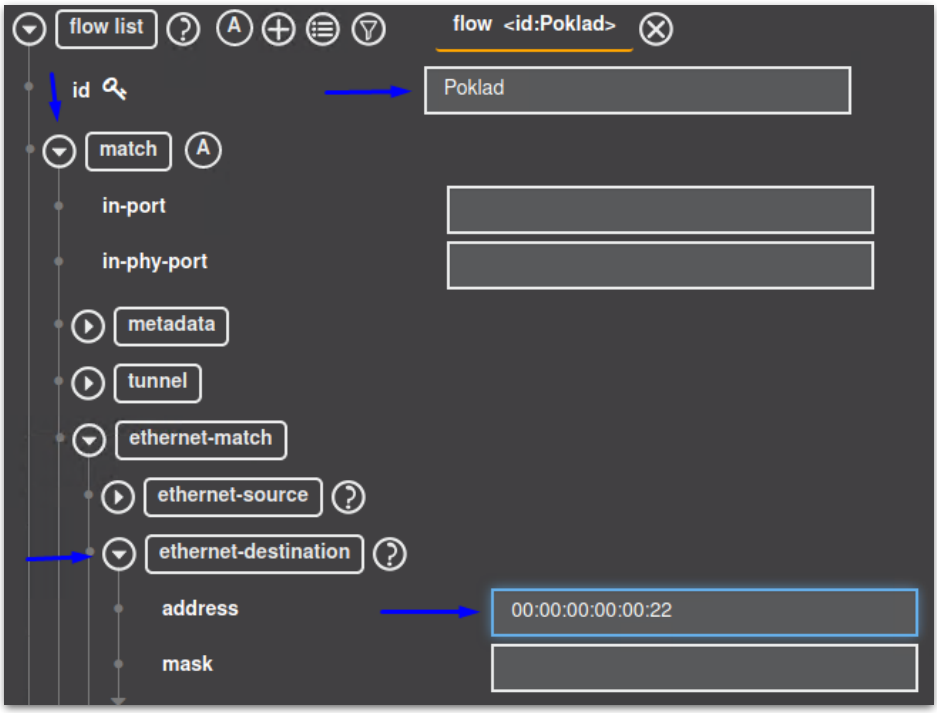
```
sh ovs-ofctl add-flow s22
priority=22,dl_dst=00:00:00:00:00:22,actions=output:2 -O OpenFlow13
```

Tieto údaje je potrebné vyhľadať a nastaviť využitím grafického rozhrania mapujúceho jednotlivé atribúty k YANG modelu. Niektoré časti vyžadujú definovanie názvu pre ID daného vkladaného objektu. Našou úlohou je vytvárať štruktúru YANG modelu, ktorá bude obsahovať tie parametre, ktoré chceme aby obsahoval. Na začiatku si môžeme všimnúť, že náš model je prázdny. Vytvárať ho je možné stláčaním tlačidla plus, čo znázorňuje obrázok 7.8:



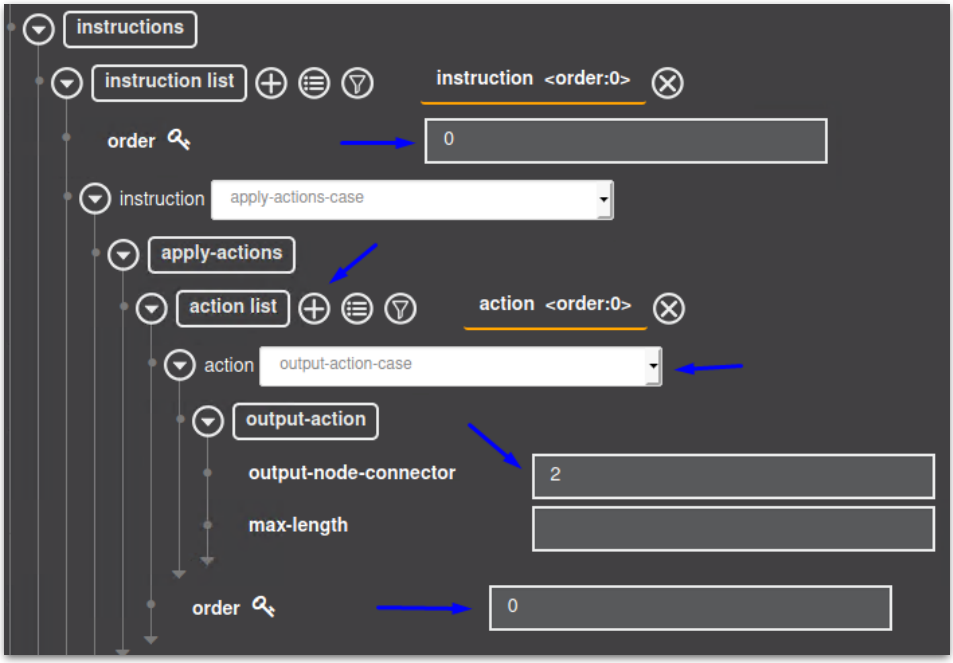
Obrázok 7.8 Tlačidlo pre vytvorenie nového toku

Obrázok 7.9 znázorňuje nastavenie názvu toku, rozbalenie ponuky pre identifikačný atribút (v našom prípade MAC adresa) a nastavenie samotnej MAC adresy:



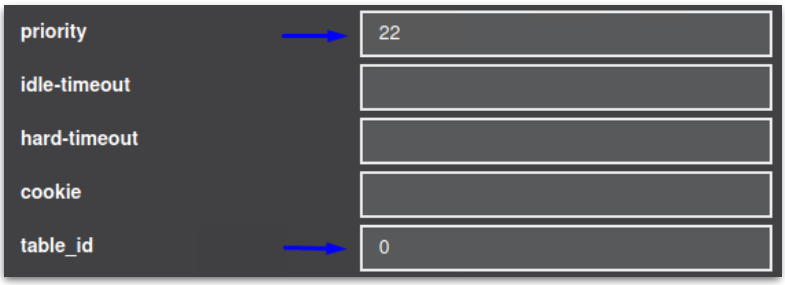
Obrázok 7.9 Definovanie základných parametrov pre vytváraný dátový tok

Po definovaní **Match** pravidla je potrebné ešte nastaviť inštrukciu, ktorá sa má vykonať. V našom prípade je potrebné zabezpečiť, aby boli dáta preposlané cez rozhranie **eth2**. Potrebné nastavenie znázorňuje obrázok 7.10:



Obrázok 7.10 Definovanie inštrukcií pre vytváraný dátový tok

Hodnoty, pri ktorých sa nachádza kľúč je potrebné vyplniť z pohľadu validity vytváraného YANG modelu. Na ich hodnotách ale aktuálne, pri definovaní jednej akcie a inštrukcie, nezáleží. Dôležité je zvoliť správny typ inštrukcie na **apply-actions-case** a akcie na **output-action-case**, čo umožní nastaviť výstupné rozhranie, na ktoré bude identifikovaný rámec odoslaný. Poslednou časťou, ktorú je potrebné z nášho pohľadu nastaviť, je priorita daného dátového toku, čo znázorňuje obrázok 7.11:



Obrázok 7.11 Definovanie priority pre vytváraný dátový tok

Po stlačení tlačidla **Send** bude naša požiadavka, využitím protokolu **RESTCONF**, odoslaná na cieľový OpenFlow prepínač a vloží nový záznam do tabuľky tokov. Obrázok 7.12 znázorňuje vytvorený záznam:

```
cookie=0x0, duration=12.158s, table=0, n_packets=0, n_bytes=0, priority=22,dl_dst=00:00:00:00:00:22 actions=output:"s22-eth2"
```

Obrázok 7.12 Zobrazenie vytvoreného dátového toku na OpenFlow prepínači

Grafické rozhranie tiež umožňuje, využitím tlačidla na zobrazenie **Preview**, zobraziť celý tvar URL adresy objektu spolu s vytvoreným YANG modelom zapísaným v JSON formáte. Obrázok 7.13 znázorňuje spomínané dáta:

```
http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:2
2/flow-node-inventory:table/0/flow/Poklad
{
    "flow": [
        {
            "id": "Poklad",
            "match": {
                "ethernet-match": {
                    "ethernet-destination": {
                        "address": "00:00:00:00:00:22"
                    }
                }
            },
            "instructions": {
                "instruction": [
                    {
                        "order": "0",
                        "apply-actions": {
                            "action": [
                                {
                                    "output-action": {
                                        "output-node-connector": "2"
                                    },
                                    "order": "0"
                                }
                            ]
                        }
                    }
                ]
            },
            "priority": "22",
            "table_id": "0"
        }
    ]
}
```

Obrázok 7.13 Zobrazenie dátového toku v JSON formáte

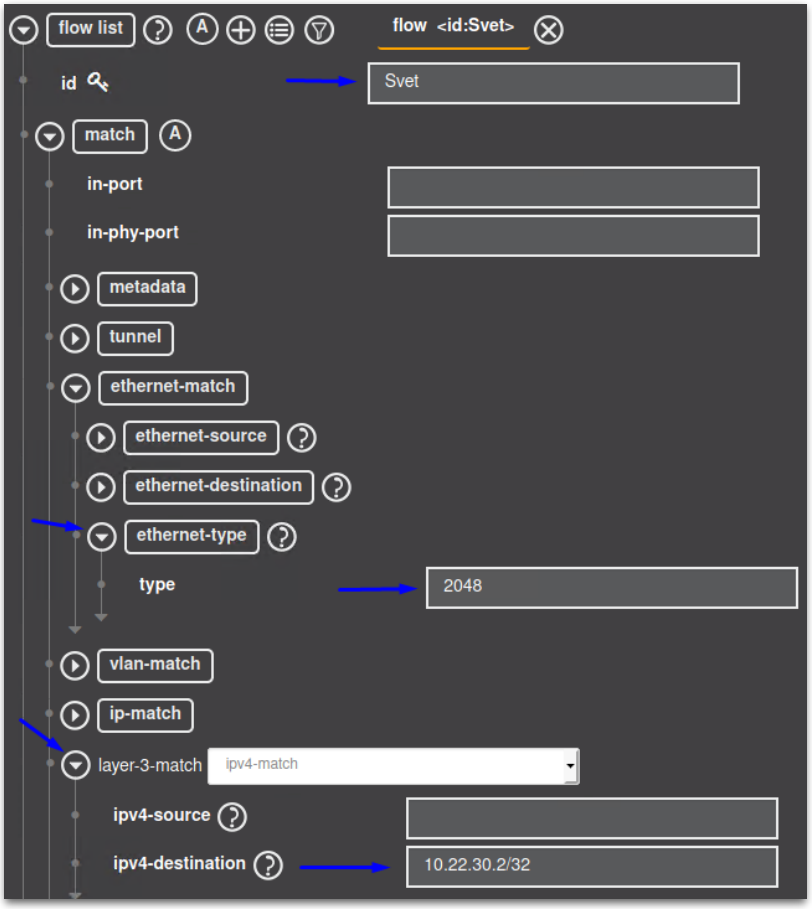
Tieto údaje je neskôr možné použiť, pri práci vo zvolenom programovacom jazyku, pre vytváranie rôznych obslužných funkcií.

Scenár môže byť nasledovný: Používateľ s MAC adresou 00:00:00:00:00:22 sa pripojí k rozhraniu eth2. Pred tým sa ale musí autentifikovať vzhľadom k autentifikačnému serveru (ISE, ...). Na túto udalosť môže následne reagovať náš kontrolér, ktorý vygeneruje vyššie uvedený JSON objekt a automaticky vloží potrebný záznam na OpenFlow prepínač, ktorý umožní doručovať dáta k používateľovi.

Príklad 2: Vytvorte dátový tok, ktorý na základe cieľovej IP adresy (10.22.30.2) bude na prepínači **S2** preposielať dáta cez rozhranie **eth1** (príklad je vzhľadom k uvedenej topológii). Uistite sa, že dátový tok bude mať v tabuľke tokov **0** prioritu **1030**, neexspiruje nikdy a bude mať názov **Svet**. Manuálna konfigurácia pre uvedenú úlohu by vyzerala nasledovne:

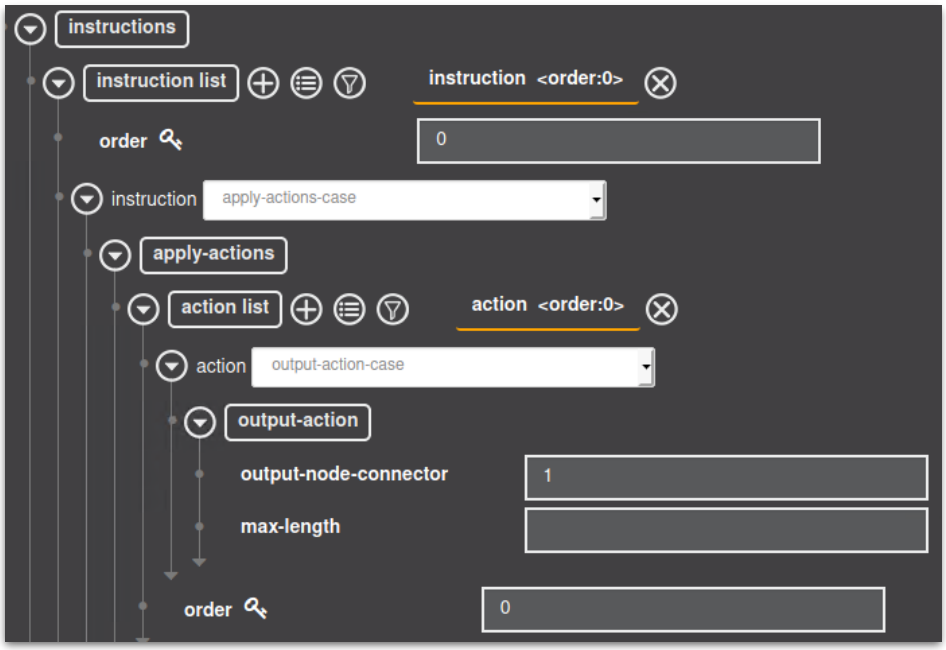
```
sh ovs-ofctl add-flow s2
priority=1030,ip,nw_dst=10.22.30.2,actions=output:1 -O OpenFlow13
```

V tomto prípade je potrebné myslieť na dvojicu faktov, ktoré možno nie sú úplne zjavné. Prvý **Match** záznam musí identifikovať L3 protokol (tento údaj sa nachádza v L2 hlavičke ako 0x800) a následne bude ešte potrebné pridať **Match** záznam pre identifikáciu poľa v L3 hlavičke. Údaj 0x800 je však potrebné zapísať v desiatkovej sústave a teda použiť hodnotu 2048. Pri IPv4 adrese je zase potrebné zapísať aj údaj o použitej maske, čo je v prípade jednej IP adresy /32. Obrázok 7.14 znázorňuje **Match** nastavenie konfiguračného YANG modelu:



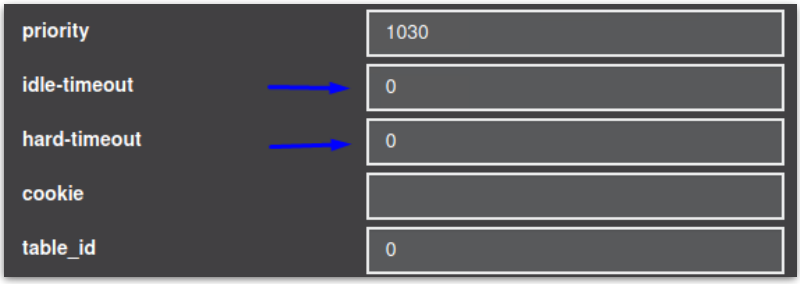
Obrázok 7.14 Definovanie základných parametrov pre vytváraný dátový tok

Z pohľadu definovania akcie sa postupuje rovnako, pričom znova stačí definovať typ inštrukcie, akciu a kľúčové identifikátory. Obrázok 7.15 znázorňuje potrebné nastavenie:



Obrázok 7.15 Definovanie inštrukcií pre vytváraný dátový tok

Nakoniec už len stačí definovať prioritu. Čo sa týka exspirácie záznamov z tabuliek, tú je možné nastaviť cez atribúty **idle/hard timeout**. Hodnota **0** znamená, že záznam neexspiruje. Obrázok 7.16 znázorňuje uvedené nastavenie:



Obrázok 7.16 Definovanie priority pre vytváraný dátový tok

Obrázok 7.17 znázorňuje vytvorený záznam uložený do tabuľky tokov na prepínači **S2**:

```
cookie=0x0, duration=25.150s, table=0, n_packets=0, n_bytes=0, priority=1030,ip,nw_dst=10.22.30.2 actions=output:"s22-eth1"
```

Obrázok 7.17 Zobrazenie vytvoreného dátového toku na OpenFlow prepínači

Obrázok 7.18 znázorňuje vytvorenú URL a JSON objekt, reprezentujúci zmenu konfigurácie:

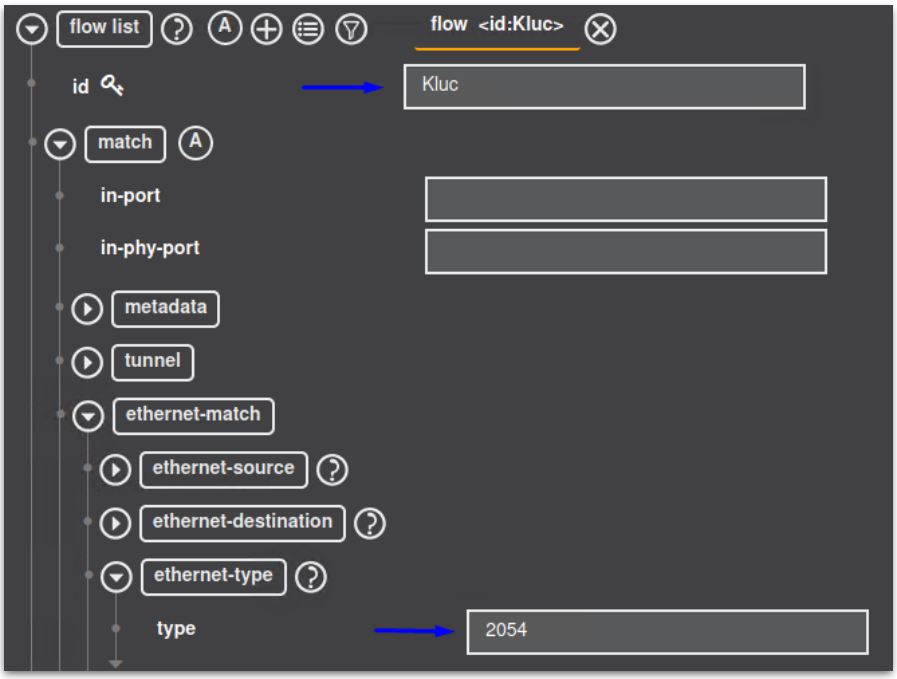
```
http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:2/flow
-node-inventory:table/0/flow/Svet
{
    "flow": [
        {
            "id": "Svet",
            "match": {
                "ethernet-match": {
                    "ethernet-type": {
                        "type": "2048"
                    }
                },
                "ipv4-destination": "10.22.30.2/32"
            },
            "instructions": {
                "instruction": [
                    {
                        "order": "0",
                        "apply-actions": {
                            "action": [
                                {
                                    "output-action": {
                                        "output-node-connector": "1"
                                    },
                                    "order": "0"
                                }
                            ]
                        }
                    }
                ]
            },
            "priority": "1030",
            "idle-timeout": "0",
            "hard-timeout": "0",
            "table_id": "0"
        }
    ]
}
```

Obrázok 7.18 Zobrazenie dátového toku v JSON formáte

Príklad 3: Vytvorte záznam pre definovanie funkcionality protokolu ARP. Dátový tok nech má názov **Kluc**. Zvyšok nastavení nech ostane v štandardnej podobe. Nezabudnite, že protokol ARP funguje tak, že jeho správy sú Flood-ované cez všetky rozhrania prepínača. Nastavenie realizujte na zariadení **S22**. Manuálne definovanie toku využitím shell príkazu vyzerá nasledovne:

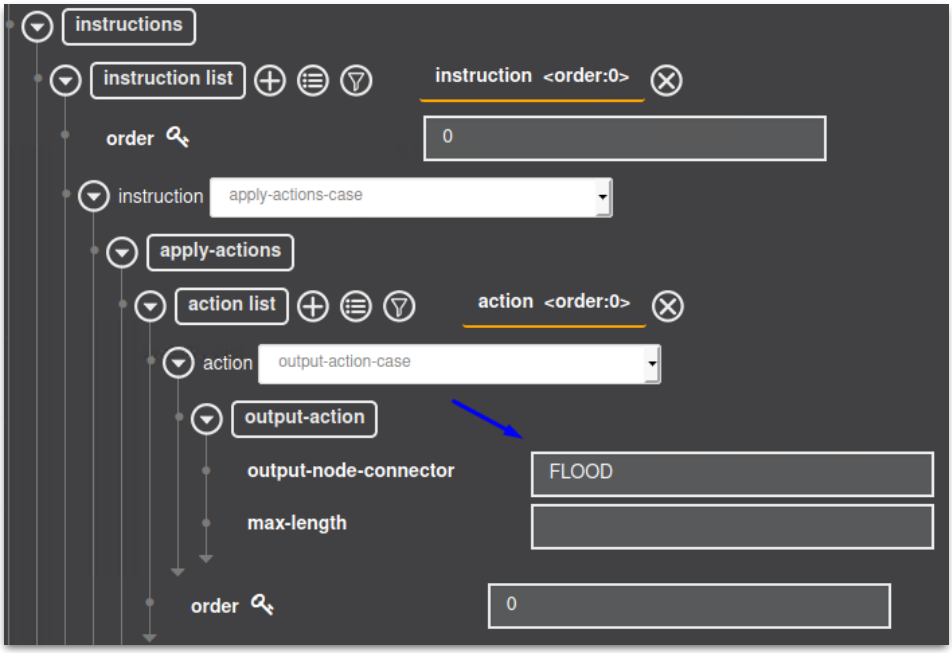
```
sh ovs-ofctl add-flow s22
dl_type=0x806,actions=flood -O OpenFlow13
```

Pri nastavení Ethernet type je potrebné vložiť číslo 2054, čo predstavuje hodnotu 0x806. Obrázok 7.19 znázorňuje **Match** sekciu pre definovanie správ ARP protokolu:



Obrázok 7.19 Definovanie základných parametrov pre vytváraný dátový tok

Definovanie typu inštrukcie ostáva rovnaké. Jedinou zmenou je definovanie akcie, ktorej hodnotu je potrebné nastaviť na **FLOOD**. Obrázok 7.20 znázorňuje uvedené nastavenie:



Obrázok 7.20 Definovanie inštrukcií pre vytváraný dátový tok

Výsledný záznam vytvorený v tabuľke tokov vyzerá nasledovne (obrázok 7.21):

```
cookie=0x0, duration=14.012s, table=0, n_packets=0, n_bytes=0, arp actions=FLOOD
```

Obrázok 7.21 Zobrazenie vytvoreného dátového toku na OpenFlow prepínači

Obrázok 7.22 znázorňuje výslednú URL adresu spolu s vytvoreným JSON objektom:

```
http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflo
w:22/flow-node-inventory:table/0/flow/Kluc
{
    "flow": [
        {
            "id": "Kluc",
            "match": {
                "ethernet-match": {
                    "ethernet-type": {
                        "type": "2054"
                    }
                }
            },
            "instructions": {
                "instruction": [
                    {
                        "order": "0",
                        "apply-actions": {
                            "action": [
                                {
                                    "output-action": {
                                        "output-node-connector": "FLOOD"
                                    },
                                    "order": "0"
                                }
                            ]
                        }
                    }
                ]
            },
            "table_id": "0"
        }
    ]
}
```

Obrázok 7.22 Zobrazenie dátového toku v JSON formáte