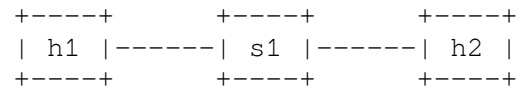
# Nowoczesne sieci komputerowe

Laboratorium – sieci programowalne

## 1. Zapoznanie z środowiskiem mininet

Środowisko **mininet** jest zainstalowane na udostępnionej maszynie wirtualnej, można je również pobrać ze strony http://mininet.org.




Aby uruchomić prostą topologię złożoną z jednego przełącznika Open vSwitch oraz dwóch hostów:

```
sudo mn --topo=single,2
```

Po uruchomieniu środowiska otworzony zostanie tekstowy interfejs, aby podejrzeć konfigurację można posłużyć się poleceniami:

```
mininet> net
mininet> links
mininet> nodes
```

Uruchamianie poleceń na konkretnym przełączniku lub hoście:

```
mininet> h1 ping h2        # można stosować nazwy hostów zamiast adresacji IP
mininet> h1 ip a
mininet> s1 ip link
```

Uruchamianie poleceń powłoki:

```
mininet> sh ovs-vsctl show
mininet> sh ovs-ofctl show s1
```

Zamknij mininet za pomocą polecenia **exit** lub znakiem EOF – Ctrl+D.

Wybór kontrolera OpenFlow:

```
sudo mn --topo=single,2 --controller=<default|none|ref|remote>
```

→ Wykonaj polecenie *pingall* po uruchomieniu sieci bez kontrolera (opcja *none*).

Przy pomocy opcji --*mac* adresy MAC hostów zostaną skonfigurowane z użyciem prostych id:

```
sudo mn --topo=single,2 --mac
mininet> pingall
mininet> h1 ip link
```

Przełącznik Open vSwitch posiada narzędzie **ovs-ofctl** do konfiguracji reguł OpenFlow na przełącznikach, dodatkowo możemy użyć polecenia **dpctl** w mininet, aby wykonać ovs-ofctl na wszystkich przełącznikach jednocześnie. Przykłady:

```
mininet> pingall
```

```
mininet> dpctl show
mininet> dpctl dump-flows
mininet> sh ovs-ofctl dump-flows s1
mininet> sh ovs-ofctl del-flows s1
mininet> sh ovs-ofctl add-flow s1 action=normal # tryb zwykłego przełącznika L2
```

Przełącznik --topo pozwala na wybranie jednej z dostępnych przykładowych topologii:

```
sudo mn --topo=single,<liczba_hostów>  # pojedynczy przełącznik z hostami
sudo mn --topo=linear,<k>,<n>          # linia k przełączników z liczbą n hostów
sudo mn --topo=tree,<głębokość>,<fanout> # topologia drzewa
sudo mn --topo=torus,<x>,<y>        # topologia torusa, uwaga zawiera pętle!
    # używaj opcji torus tylko bez kontrolera, z własnym kontrolerem lub STP
```

Jest możliwość stworzenia własnych topologii wraz z konfiguracją hostów, prosty przykład:

*mesh_topo.py*

```
from mininet.topo import Topo

class MyTopo( Topo ):

    def build( self ):
        # Add hosts and switches
        h1 = self.addHost( 'h1' )
        h2 = self.addHost( 'h2' )
        h3 = self.addHost( 'h3' )
        h4 = self.addHost( 'h4' )
        s1 = self.addSwitch( 's1' )
        s2 = self.addSwitch( 's2' )
        s3 = self.addSwitch( 's3' )
        s4 = self.addSwitch( 's4' )


        switches = (s1, s2, s3, s4)
        hosts = (h1, h2, h3, h4)
        # Add links
        for i, switch in enumerate(switches):
            for target in switches[i+1:]:
                self.addLink(switch, target)
        for i in range(len(switches)):
            self.addLink(hosts[i], switches[i])


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Uruchom poleceniem:

```
sudo mn --custom mesh_topo.py --topo=mytopo --controller=none
```

# 2. Manualna konfiguracja podstawowych reguł OpenFlow przełączników OpenVSwitch

Uruchom prostą topologię z czterema hostami, bez kontrolera.

```
sudo mn --topo=single,4 --controller=none --mac
mininet> pingall   # brak łączności
mininet> sh ovs-ofctl dump-flows s1   # wyświetl listę reguł
```

2.1 Przełączanie całego ruchu na interfejsach pomiędzy h1 a h3 oraz h2 i h4.

```
mininet> sh ovs-ofctl del-flows s1
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=1,actions=output:3
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=3,actions=output:1
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=2,actions=output:4
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=4,actions=output:2
mininet> h1 ping h3
```

2.2 Przełączanie ruchu w ramach warstwy 2: h1-h3, h2-h4.

```
mininet> sh ovs-ofctl del-flows s1      # usunięcie wszystkich reguł
```

- Ruch warstwy dostępowej:

```
mininet> sh ovs-ofctl add-flow s1
dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03,actions=output:3
mininet> sh ovs-ofctl add-flow s1
dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01,actions=output:1
mininet> sh ovs-ofctl add-flow s1
dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:02,actions=output:2
mininet> sh ovs-ofctl add-flow s1
dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:04,actions=output:4
```

- Reguła do broadcastu ARP:

```
mininet> sh ovs-ofctl add-flow s1 dl_type=0x0806,nw_proto=1,actions=flood
```

2.3 Przełączanie ruchu w ramach warstwy 3 (routing): h1-h3, h2-h4.

Wyjdź z mininet i uruchom ponownie, by opróżnić tablice arp hostów.

Ruch IP:

```
mininet> sh ovs-ofctl add-flow s1
dl_type=0x0800,nw_src=10.0.0.1/32,nw_dst=10.0.0.3/32,actions=output:3
mininet> sh ovs-ofctl add-flow s1
dl_type=0x0800,nw_src=10.0.0.3/32,nw_dst=10.0.0.1/32,actions=output:1
mininet> sh ovs-ofctl add-flow s1
dl_type=0x0800,nw_src=10.0.0.2/32,nw_dst=10.0.0.4/32,actions=output:4
mininet> sh ovs-ofctl add-flow s1
dl_type=0x0800,nw_src=10.0.0.4/32,nw_dst=10.0.0.2/32,actions=output:2
```

Ruch ARP:

```
mininet> sh ovs-ofctl add-flow s1
dl_type=0x0806,nw_src=10.0.0.3,nw_dst=10.0.0.1,actions=output:1

mininet> sh ovs-ofctl add-flow s1
dl_type=0x0806,nw_src=10.0.0.4,nw_dst=10.0.0.2,actions=output:2

mininet> sh ovs-ofctl add-flow s1
dl_type=0x0806,nw_src=10.0.0.1,nw_dst=10.0.0.3,actions=output:3

mininet> sh ovs-ofctl add-flow s1
dl_type=0x0806,nw_src=10.0.0.2,nw_dst=10.0.0.4,actions=output:4
```

2.4 Przełączanie z modyfikacją portu docelowego i źródłowego.

```
mininet> sh ovs-ofctl del-flows s1      # usunięcie wszystkich reguł

mininet> sh ovs-ofctl add-flow s1 priority=100,actions=normal

mininet> sh ovs-ofctl add-flow s1
priority=200,ip,nw_proto=6,tp_dst=80,actions=mod_tp_dst=8080,normal

mininet> sh ovs-ofctl add-flow s1
priority=200,ip,nw_proto=6,tp_src=8080,actions=mod_tp_src=80,normal

mininet> h2 python3 -m http.server 8080 &

mininet> h1 curl h2
```

Uruchom program tshark przed zapytaniem na portach s1-eth1 i s1-eth2 aby upewnić się, że następuje modyfikacja portu.

```
sudo tshark -i s1-eth1
```

**Zadanie.** Wykorzystaj topologię mesh_topo z poprzedniego ćwiczenia i skonfiguruj następujące połączenia:

- wszystkie węzły osiągają siebie nawzajem (pingall)
- komunikacja TCP pomiędzy hostami h1 i h2, przebiega poprzez kolejne 4 przełączniki s1,s2,s3 i s4 (najdłuższą drogą)
- zablokuj docieranie pakietów UDP z hosta h3 do hosta h2

Dodatkowe materiały:

- David Mahler: OpenFlow flow entries on Open vSwitch (OVS) https://www.youtube.com/watch?v=FyV4MoQ3T0I

Dostępne pola protokołów dla wpisów OpenFlow:

- in_port, dl_src, dl_dst, dl_vlan, dl_type, nw_tos, nw_proto, nw_src, nw_dst, tp_src, tp_dst, mpls_label, mpls_tc

Dostępne akcje dla wpisów OpenFlow:

- output, flood, drop, mod_dl_src, mod_dl_dst, mod_dl_vlan, mod_nw_tos, mod_nw_src, mod_nw_dst, mod_tp_src, mod_tp_dst, mod_mpls_label, mod_mpls_tc

# 3. Obserwacja komunikacji kontrolera OpenFlow z przełącznikiem

Uruchom **tshark** nasłuchujący na interfejsie loopback *lo*. Pomocny w analizie pakietów może się okazać przełącznik verbose -V oraz przekierowanie wyjścia do pliku. Komunikacja pomiędzy przełącznikiem a kontrolerem odbywa się poprzez interfejs loopback z powodu komunikacji wewnątrz jednego hosta.

```
sudo tshark –Vi lo > lo_packets.txt
```

Jeśli używasz systemu Linux jako hosta maszyny wirtualnej, możesz wykorzystać poniższą linijkę do przekierowania wyjścia programu tshark w formacie pcapng do lokalnego wiresharka (jeśli nie skonfigurowaliśmy klucza ssh po uruchomieniu polecenia musimy podać hasło).

```
ssh –p 2222 vagrant@localhost sudo tshark –i lo –w – | wireshark –k –i –
```

Następnie w osobnym terminalu uruchom mininet:

```
sudo mn --controller=ref --topo=single,4 --mac
```

**Zadanie.** Przeanalizuj pakiety protokołu OpenFlow:

A) Podaj schemat wymiany komunikatów po nawiązaniu połączenia przełącznika z kontrolerem.

B) Do czego służą pakiety OFPT_PACKET_IN, OFPT_FLOW_MOD i OFPT_PACKET_OUT?

Pomocne materiały:

- David Mahler: Introduction to OpenFlow https://www.youtube.com/watch?v=l25Ukkmk6Sk
- Specyfikacja OpenFlow 1.0

# 4. Własna implementacja kontrolera OpenFlow w języku Python [dla chętnych]

Framework Ryu służy do tworzenia kontrolerów OpenFlow i zawiera wiele gotowych aplikacji dla przełącznika OpenFlow.  Ryu znajduje się w udostępnionej do laboratoriów maszynie wirtualnej.

Uruchamianie przykładowej aplikacji SDN na każdym podłączonych przełączniku OpenFlow (tj.: uruchamianie przełącznika warstwy 2 na przełączniku OpenFlow):

```
ryu-manager ryu/ryu/app/simple_switch_13.py
```

W osobnym terminalu uruchamiamy infrastrukturę kontrolowaną przez kontroler OpenFlow:

```
sudo mn --topo=single,3 --controller=remote --mac
```

**Zadanie.** Zapoznaj się z frameworkiem Ryu i stwórz własny kontroler OpenFlow.

Materiały:

- https://osrg.github.io/ryu/
- https://osrg.github.io/ryu-book/en/html/  - podręcznik
- https://github.com/osrg/ryu – kod źródłowy
- https://github.com/osrg/ryu/tree/master/ryu/app  - przykładowe programy