1 Laboratorní úloha SDN

Vaším úkolem navrhnout a implementovat jednoduchou aplikaci integrovanou do kontroleru **Ryu**. Tento kontroler bude komunikovat s virtuálním přepínačem pomocí protokolu **OpenFlow**. Každá student bude mít k dispozici virtuální stroj s předinstalovaným prostředím připraveným pro spuštění kontroleru a naprogramování aplikace.

Cílem cvičení je vyzkoušet rozbočovač (*hub*) a na jeho základě vytvořit přepínač (*switch*). Nejdříve vyzkoušíte spojení mezi kontrolerem a řízeným přepínačem a poté naprogramujete aplikaci pro kontroler tak, aby řízený virtuální přepínač pracoval v režimu rozbočovač.

Následně vytvořte klasický L2 učící se přepínač, který bude jednotlivé příchozí rámce přepínat dle cílových MAC adres připojených stanic. V případě zájmu můžete kontroler rozšířit i o další funkce, např. blokování přístupu, filtrování provozu nebo sestavení topologie sítě, případně propojení dvou virtuálních přepínačů pomocí GRE tunelu.

Základní struktura zdrojového kódu kontroleru je naznačena v příloze B a bude k dispozici v textové formě i v připraveném virtuálním prostředí.

1.1 Python

Jazyk Python je programovací jazyk vytvořený s cílem jednoduchého a přehledného zápisu programu tak, aby se ideálně blížil zápisu pseudokódu. Jedná se o objektový interpretovaný jazyk. Při cvičení budete potřebovat pouze základní operace, takže není potřeba podrobně procházet všechny níže zmíněné informační zdroje. Nicméně lepší znalost jazyka Vám usnadní práci, protože nebudete muset všechny příkazy hledat.

Jelikož se jedná o jazyk interpretovaný, tak lze zdrojový kód spustit například příkazem `python ./soubory.py` nebo přímo `./soubor.py` pokud je definována první řádka s běhovým prostředím. Zdrojový kód v příloze A ukazuje základní konstrukce, které budete při cvičení pravděpodobně používat. Pokud ještě nemáte jazyk nainstalovaný a chcete si základní příkazy vyzkoušet, doručujeme použít Virtualenv pro Linux a WinPython pro Windows. Místo základního běhového prostředí doporučujeme používat iPython, jež je uživatelsky mnohem přívětivější. Toto běhové prostředí je integrovanou součástí WinPython a v Linuxu jej lze doinstalovat u většiny distribucí z repozitáře.

- České stránky o pythonu - `http://python.cz/`
- Podrobný kurz na codeacademy - `http://www.codecademy.com/en/tracks/python`
- Virtualenv - `http://docs.python-guide.org/en/latest/dev/virtualenvs/`
- WinPython - `http://winpython.sourceforge.net/`
- iPython - `http://ipython.org/`

1.2 Ryu

Ve cvičení budete používat framework pro kontroler, který se jmenuje Ryu. Je napsaný v pythonu a dostupný jako OpenSource. Základní kód aplikace v kontroleru budete mít připravený na cvičení a také je v příloze B.

Vaším úkolem bude doplnit do této šablony vlastní logiku řídící aplikace. Můžete se inspirovat níže uvedenými zdroji, zejména druhý zdroj by Vám mohl velmi pomoci. Kontroler spustíte příkazem `ryu-manager soubor.py`

- Zdrojové kódy Ryu kontroleru - `https://github.com/osrg/ryu`
- První aplikace v Ryu - `http://ryu.readthedocs.org/en/latest/writing_ryu_app.html`
- Dokumentace k OpenFlow třídám v Ryu - `http://ryu.readthedocs.org/en/latest/ofproto_v1_3_ref.html`

1.3 Mininet

V rámci demonstrační úlohy budete využívat virtální přepínač Open VSwitch skrze prostředí emulátor datových sítí Mininet. Tento software dokáže vytvořit virtuální síť včetně přepínačů a připojených stanic.

Připravený VM jehož adresu obdržíte na cvičení již obsahuje předdefinovanou Mininet síť s jedním přepínačem a 5-ti stanicemi připojenými k tomuto přepínači. Všechny potřebné zdrojové kódy jsou vám přístupné po zadání **sudo -i** v adresáři */root/pst/*. Ke spuštění jsou pro Vás připraveny wrappery, které spustíte v adresáři */root/pst* jako uživatel *root* následujícími příkazy.

```
./controller.sh [template.py] a ./mininet.sh
```

Jelikož je nutné provozovat současně Mininet i Ryu kontroler doporučujeme použít pro rozdělení terminálové obrazovky například *tmux*.

V případě nutnosti můžete síť simulovanou vytvořit příkazem

```
mn --controller=remote,ip=192.168.1.1,port=6633 --topo=single,5 --mac
```

a ještě je nutné zapnout podporu OpenFlow 1.3 na vSwitch příkazem

```
ovs-vsctl set bridge s1 protocols=OpenFlow10,OpenFlow13
```

- Tutorial Ryu a Mininet - `https://github.com/osrg/ryu/wiki/OpenFlow_Tutorial`

A Základy pythonu

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# import package
import time

# assign variable
a = 1 # integer
b = 'some string' #string
c = {'Kevin': 1, 'Clu': 34, 'Quorra': 90} # dict

# print variable
print a
print b
print 'a: %i, b: %s' % (a, b)
print c
print c['Clu']

# if
if a == 1:
    print 'a equals 1'
else:
    print 'a !equals 1'

# for
for k in c:
    print 'c[%s] is %s' % (k, c[k])

# test value in dict
if 'Clu' in c:
    print 'Clu is in c'

if 'Chris' in c:
    print 'Chris is in c'
else:
    print 'Chris is NOT in c'

# function definition
# optionally a @decorator can be added
def myFunc(input1,input2=True):
    if input2:
        return input1*2
    else:
        return 0
```

B Šablona kontroleru v Ryu

```
# imports
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls

# openflow version 1.3
from ryu.ofproto import ofproto_v1_3

# timestamp
from datetime import datetime


class Hub(app_manager.RyuApp):
    # set OF 1.3
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    # class initialization
    def __init__(self, *args, **kwargs):
        super(Hub, self).__init__(*args, **kwargs)

    # event definition
    # run on: ofp_event.EventOFPPacketIn
    # when switch state is: MAIN_DISPATCHER (connected to controller)
    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def packet_in_handler(self, ev):
        ## get data about incoming mgs
        print '--- new packet_in at %s' % datetime.now()

        # incomming msg - OFPPacketIn
        msg = ev.msg
        print msg

        # datapath - (= switch)
        datapath = msg.datapath
        print datapath

        # OF protocol
        ofp = datapath.ofproto
        ofp_parser = datapath.ofproto_parser
        print ofp
        print ofp_parser

        ## add your code here
```

C Nápověda

```
from ryu.lib.packet import packet, ethernet
pkt = packet.Packet(msg.data)
eth = pkt.get_protocol(ethernet.ethernet)
if eth != None:
...
ofp_parser.OFPInstructionActions
ofp.OFPIT_APPLY_ACTIONS
ofp_parser.OFPFlowMod
```