UNIVERSIDADE ESTADUAL DE CAMPINAS

INF-0556 :: REDES DEFINIDAS POR SOFTWARE.
CURSO DE ESPECIALIZAÇÃO EM REDES DE COMPUTADORES
MODALIDADE EXTENSÃO UNIVERSITÁRIA

Lab 3 : Openflow e Controlador RYU

Prof.: Christian Esteve Rothenberg

Monitor: Fabricio Rodríguez

Monitor: Wilson Borba

Campinas, SP
2019

Máquinas do Laboratório

As máquinas (computadores) do laboratório devem ser acessadas com as contas fornecidas para cada aluno. Caso a máquina não inicie corretamente, peça auxílio ao responsáveis pelo laboratório. Para iniciar a VM utilizada nessa prática, abra um terminal e execute o comando a seguir.

```
$ vm-inf556-lab-01
```

Para acessar a VM, utilize as seguintes credenciais:

```
Login:  wifi
Password:  wifi
```

Além disso, para esta prática é necessário instalar o software RYU. RYU é um controlador SDN desenvolvido em Python famoso pela proposta de simplicidade no desenvolvimento de sua arquitetura, permitindo ao usuário o controle do comportamento em baixo nível do controlador. Isso faz do Ryu um controlador vantajoso para cenários simples que não necessitam de um controlador de alta performance, sendo, também, uma ótima escolha para o primeiro contato com controladores SDN.

Para instalar o Ryu e todas as dependências necessárias execute os seguintes passos.

```
$ wget https://raw.githubusercontent.com/intrig-unicamp/ea080
/master/software/install_ryu.sh

$ sudo chmod u+x install_ryu.sh

$ sudo ./install_ryu.sh
```

Atividades

A Figura 1é uma ilustração da topologia simples que será usada para este tutorial. A topologia possui um switch OpenFlow S1, conectando dois hosts H1 e H2.

⚠**Nota:** O OpenFlow 1.3 é o protocolo padrão para o mininet. Para versões mais antigas, adicione o parâmetro `protocols = OpenFlow10` após ovsk para o OpenFlow 1.0 conforme indicado no comando abaixo.

```
$ sudo mn --mac --switch ovsk, protocolos = OpenFlow10 --controller remote
--arp
```

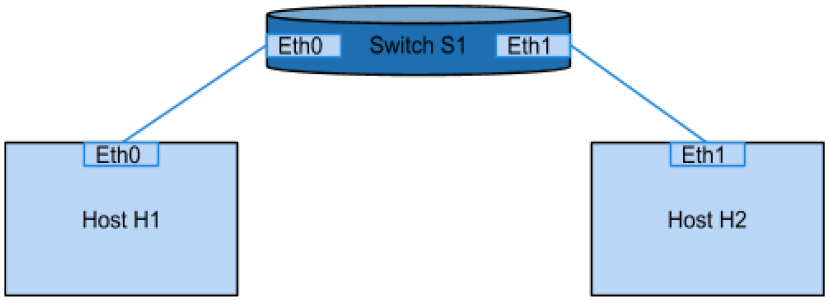

Figura 1: Topologia básica

Atividade 1: Introdução ao OpenFlow

a) Para iniciar essa topologia com o OpenFlow 1.3, inicie o mininet digitando o seguinte comando no terminal:

```
$ sudo mn --mac --switch ovsk --controller remote
```

b) Inicie o Wireshark para monitorar os pacotes no switch S1. Selecione a interface loopback (`lo`) e aplique o filtro `openflow_v4` para ver as mensagens OpenFlow trocadas entre Switch e Controlador.

```
$ s1 wireshark -i lo -k &
```

c) O controlador Ryu possui um framework de gerenciamento responsável por dar suporte e comunicar aplicativos do plano de controle com os equipamentos no plano de dados. Assim, para utilizar o Ryu cria-se um script de um aplicativo que é instanciado pelo gerenciador do ryu (`ryu-manager`). Por enquanto, vamos iniciar apenas o framework do Ryu sem nenhum aplicativo. Para isso, abra um terminal externo ao mininet (`ctrl + alt + T`) e digite o comando a seguir.

$ ryu-manager

d) Observe os pacotes capturados no Wireshark

Q1: Quais mensagens do protocolo OpenFlow você vê e qual a função delas?

e) Agora vá ao mininet e tente executar um ping entre h1 e h2.

```
mininet> h1 ping h2 -c5
```

f) Observe os pacotes do Wireshark novamente.

Q2: O ping foi concluído com sucesso? Alguma mensagem OpenFlow foi trocada entre switch e controlador? Qual era o comportamento esperado (em relação à troca de pacotes OpenFlow) nesta situação?

Atividade 2: Learning Switch

Agora, vamos implementar nosso primeiro aplicativo OpenFlow: um Learning Switch. A seguir discutimos o funcionamento do switch e introduziremos alguma intuição em relação ao aplicativo OpenFLow.

1. Funcionalidade básica: o *learning switch* encaminha os pacotes com base no aprendizado dos endereços MAC; Ao receber um pacote, se o switch não souber em qual porta está o seu destino, ele inundará o pacote recebido em todas as interfaces; caso contrário, se souber ond está o MAC de destino, o switch encaminhará o pacote diretamente para a interface correta.

2. OpenFlow / SDN: O aplicativo OpenFlow receberá um pacote com destino desconhecido através de uma mensagem Packet-In e aprenderá o MAC e porta associado à fonte do pacote. Logo após realizará um broadcast do pacote, assim como no caso anterior. Assim que receber um novo Packet-In que tenha como destino um dos MACs aprendidos, o aplicativo encaminhará o pacote recebido para a porta relacionada ao MAC aprendido.

Controlador SDN Ryu

Vamos dar uma olhada em alguns elementos Ryu que serão necessários para concluir o exercício.

Para mais informações: https://github.com/osrg/ryu http://ryu.readthedocs.io/en/latest/

1. **ofp_event**: esta classe armazena possíveis eventos que o plano de dados pode disparar no plano de controle. O controlador Ryu é orientado a estes eventos, então podemos definir ações que devem ser realizadas quando um determinado evento acontece. No exemplo abaixo utilizamos um artifício em python (chamado de *decorator*) para executar uma função quando o evento PacketIn acontecer.

```
@set_ev_cls(ofp_event.EventOFPPacketIn)
def tratar_packet_in():
    instalar_flows()
```

2. **ofproto_parser**: é o objeto responsável pela criação e serialização das mensagens do OpenFlow. É com esse objeto que sabemos como enviar e como interpretar mensagens codificadas no protocolo OpenFlow.

```
parser = datapath.ofproto_parser
```

3. **datapath**: é o objeto que representa os switches que estabelecem comunicações com o controlador. Ele possui um ofproto_parser como membro da classe, já que cada switch "sabe"com qual versão do OpenFlow a comunicação com o controlador foi estabelecida. O datapath pode ser obtido das mensagens recebidas pelo controlador, já que o controlador "sabe"qual switch enviou a mensagem.

```
@set_ev_cls(ofp_event.EventOFPPacketIn)
def tratar_packet_in(self, ev):
   # ev : evento recebido
```

```
        # msg: mensagem do evento (Packet In)
        msg = ev.msg
        datapath = msg.datapath
        out = criar_packet_out()
        datapath.send_msg(out)
        return 1
```

4. **ofproto**: Contém definições especificação OpenFlow.

Openflow 1.3

a) Inicie o mininet novamente com o Open vSwitch

```
$ sudo mn --mac --switch ovsk --controller remote --arp
```

b) Inicie o Wireshark para monitorar os pacotes no switch S1. Selecione a interface loopback (`lo`) e aplique o filtro `openflow_v4` para ver as mensagens OpenFlow trocadas entre Switch e Controlador.

```
$ s1 wireshark -i lo -k &
```

c) Abra outro terminal e observe se há flows instaladas no switch S1 com o comando `ovs-ofctl`.

```
$ sudo ovs-ofctl dump-flows s1
```

d) Execute ryu em um terminal externo ao mininet com o aplicativo *simple_switch_13*

```
$ ryu-manager ryu.app.simple_switch_13
```

e) Observe,novamente, as flows instaladas no switch S1 com o comando `ovs-ofctl`.

Q3: Qual flow está instalada no switch S1? Qual a função desempenhada por esta flow?

f) Acesse o arquivo da aplicação *Learning switch* em `/ryu/ryu/app/simple_switch_13.py` e abra o arquivo com seu editor predileto. Identifique no arquivo o trecho (função) responsável por tratar o evento de conexão de um novo switch ao controlador, assim como foi exemplificado no início dessa atividade. Copie esse trecho no relatório.

Dica: O evento de interesse nesse caso é o EventOFPSwitchFeatures

g) Sobre o trecho do item anterior, procure a função *self.add_flow()*. Ela é utilizada para a instalação de flows.

Q4: Quais são os argumentos da função `self.add_flow()`? Qual é a flow instalada plela função neste caso?

Dica: Argumentos são os parâmetros passados para função localizados dentro dos parênteses logo após o nome da função.

h) Realize o ping entre H1 e H2 e observe os pacotes no Wireshark. Observe também como o primeiro ping é mais demorado do que os demais.

```
mininet> h1 ping h2 -c5
```

Q5: Quais são os novos pacotes OpenFlow trocados entre switch e controladore em relação aqueles da Atividade 1? Qual a função desses pacotes?

Atividade 3: API REST

Nesta atividade iremos instanciar um aplicativo no controlador Ryu capaz de realizar algumas requisições no plano de controle utilizando uma API REST[1]. A API REST pode ser utilizada por aplicações externas ao controlador para gerenciar o plano de dados.

Na atividade anterior observamos que os hosts H1 e H2 possuem conectividade, mas que esta foi garantida de forma *reativa*, isto é, o controlador reagiu ao pacotes recebidos por ele. Agora, vamos instalar regras em S1 de forma *proativa*, simulando uma aplicação que saberia com antecedência que H1 e H2 necessitam de comunicação.

a) Para iniciar essa topologia com o OpenFlow 1.3, inicie o mininet digitando o seguinte comando no terminal:

```
$ sudo mn --mac --switch ovsk --controller remote --arp
```

b) Inicie o Wireshark para monitorar os pacotes no switch S1. Selecione a interface loopback (`lo`) e aplique o filtro `openflow_v4` para ver as mensagens OpenFlow trocadas entre Switch e Controlador.

```
$ s1 wireshark -i lo -k &
```

c) Agora, inicie, em um outro terminal, o aplicativo que implementa a API REST além do nosso aplicativo de controle simples da atividade passada.

```
$ ryu-manager ryu.app.ofctl_rest ryu.app.simple_switch_rest
```

d) O protocódigo a seguir será utilizado para fazer uma requisição à API REST do controlador. Modifique os campos **a**,**b**,**c** e **d** para obter as duas flows que serão utilizadas para comunicar H1 e H2 (coloque o resultado no relatório).

Dica: Utilize os comandos do mininet (`py h1.params`, links, intfs etc.) para obter informação dos hosts e topologia.

```
'{
    "dpid": 1,
    "cookie": 1,
    "hard_timeout": 0,
    "priority": 123,
    "match":{
        "in_port": a,
        "eth_dst": "b",
        "eth_src": "c"
        },
    "actions":[
        {
        "type":"OUTPUT",
        "port": d
        }
    ]
}'
```

[1] *https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html*

(a) in_port: porta de entrada do pacote no switch S1 (ex. 22)

(b) eth_dst (): endereço MAC de destino no pacote (ex. "00:00:33:00:44:55")

(c) eth_src: endereço MAC da fonte do pacote (ex. "00:00:33:00:44:55")

(d) port: porta de saída Switch S1 para qual o pacote deve ser redirecionado (ex. 5901)

e) Abra mais um terminal (prometemos ser o último) e execute uma requisição POST para o plano de controle com o comando `curl` utilizando como argumento as flows criadas no item anterior. *OBS: Não se esqueça de colocar após o argumento o endereço da requisição, conforme exemplificado a seguir.*

```
$ curl -X POST -d '{
    "dpid": 1,
    "cookie": 1,
    "cookie_mask": 1,
    "table_id": 0,
    "idle_timeout": 30,
    "hard_timeout": 30,
    "priority": 11111,
    "flags": 1,
    "match":{
        "in_port":22,
        "eth_src": "aa:bb:cc:dd:ee:ff",
        "eth_src": "11:22:33:44:55:66"
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 2
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

f) Verifique com o comando `ovs-ofctl` se as flows foram instaladas corretamente.

```
sudo ovs-ofctl dump-flows s1
```

g) Realize o ping entre H1 e H2 e observe os pacotes no Wireshark.

```
mininet> h1 ping h2 -c5
```

Q6: Comente as diferenças entre os resultados da Atividade 2 e Atividade 3. Quais os benefícios que trazidos pela API Northbound à aplicações de rede?