LAB 1/3 - OpenFlow e gerenciamento de Fluxos

Objetivo

O objetivo desse laboratório é introduzir o aluno aos conceitos do OpenFlow na prática, uma vez que esse é um dos protocolos mais utilizados para a implementação de SDN. Entendê-lo se faz essencial para a programação e depuração das aplicações desenvolvidas no paradigma de SDN.

Introdução

Nesta prática será descrita uma experiência simples para emular dispositivos OpenFlow em um computador rodando o sistema operacional Linux, através do aplicativo Mininet, além de testes usando essa rede emulada.

Iremos gerenciar os fluxos manualmente através dos comandos providos pela interface do OpenFlow e consultar estatisticas dos fluxos criados.

O objetivo dessa prática é familiarizar o aluno com algumas ferramentas básicas do OpenFlow. Apesar de ser um acesso de baixo nível, algumas dessas funcionalidades podem ser úteis para a depuração do comportamento do ambiente SDN.

Comandos básicos para gerência de *switches* Open vSwitch:

- **ovs-dpctl** (*datapath management utility*): Utilitário que envia mensagens OpenFlow ao *switch*. É útil tanto para visualizar o status do *switch* quanto para inserir manualmente entradas da tabela de fluxo;
- **ovs-ofctl** (*management utility for openflow*): Utilitário que provê visibilidade e controle sobre a tabela de fluxo de um único *switch* OVS. É especialmente útil para depuração, exibindo contadores de estado dos fluxos e de vazão. A maioria dos *switches* OpenFlow inicia-se com uma porta de escuta (por padrão 6634), a partir da qual o **ofctl** pode consultar o *switch*, evitando a instalaçao de código de depuração no controlador;
- **ovs-vsctl**: Utilitário para gerenciar o *switch* através da comunicação com o ovsdb-server (servidor de database do OpenFlow).

A maioria dos comandos acima sobrepõem diversas funções, como a possibilidade de criar e remover fluxos manualmente. Iremos apenas demonstrar algumas das funcionalidades, uma vez que na maior parte dos casos, o controlador é responsável por gerenciar as regras OpenFlow.

As principais funções do **ovs-dpctl** e **ovs-ofctl** são exibidas abaixo, como referência futura se necessário. As funções sobrepostas pelas duas funções estão realçadas em **laranja**.

```
#> ovs-dpctl --help
usage: ovs-dpctl [OPTIONS] COMMAND [ARG...]
  add-dp DP [IFACE...]     add new datapath DP (with IFACEs)
  del-dp DP                delete local datapath DP
  add-if DP IFACE...       add each IFACE as a port on DP
  set-if DP IFACE...       reconfigure each IFACE within DP
  del-if DP IFACE...       delete each IFACE from DP
  dump-dps                 display names of all datapaths
  show                     show basic info on all datapaths
  show DP...               show basic info on each DP
  dump-flows DP            display flows in DP
  add-flow DP FLOW ACTIONS add FLOW with ACTIONS to DP
  mod-flow DP FLOW ACTIONS change FLOW actions to ACTIONS in DP
  del-flow DP FLOW         delete FLOW from DP
  del-flows DP             delete all flows from DP
```

```
#> ovs-ofctl --help
usage: ovs-ofctl [OPTIONS] COMMAND [ARG...]
  show SWITCH                  show OpenFlow information
  dump-desc SWITCH             print switch description
  dump-tables SWITCH           print table stats
  mod-port SWITCH IFACE ACT    modify port behavior
  get-frags SWITCH             print fragment handling behavior
  set-frags SWITCH FRAG_MODE   set fragment handling behavior
  dump-ports SWITCH [PORT]     print port statistics
  dump-ports-desc SWITCH       print port descriptions
  dump-flows SWITCH            print all flow entries
  dump-flows SWITCH FLOW       print matching FLOWs
  dump-aggregate SWITCH        print aggregate flow statistics
  dump-aggregate SWITCH FLOW   print aggregate stats for FLOWs
  queue-stats SWITCH [PORT [QUEUE]]  dump queue stats
  add-flow SWITCH FLOW         add flow described by FLOW
  add-flows SWITCH FILE        add flows from FILE
  mod-flows SWITCH FLOW        modify actions of matching FLOWs
  del-flows SWITCH [FLOW]      delete matching FLOWs
  replace-flows SWITCH FILE    replace flows with those in FILE
  diff-flows SOURCE1 SOURCE2   compare flows from two sources
  packet-out SWITCH IN_PORT ACTIONS PACKET...
```

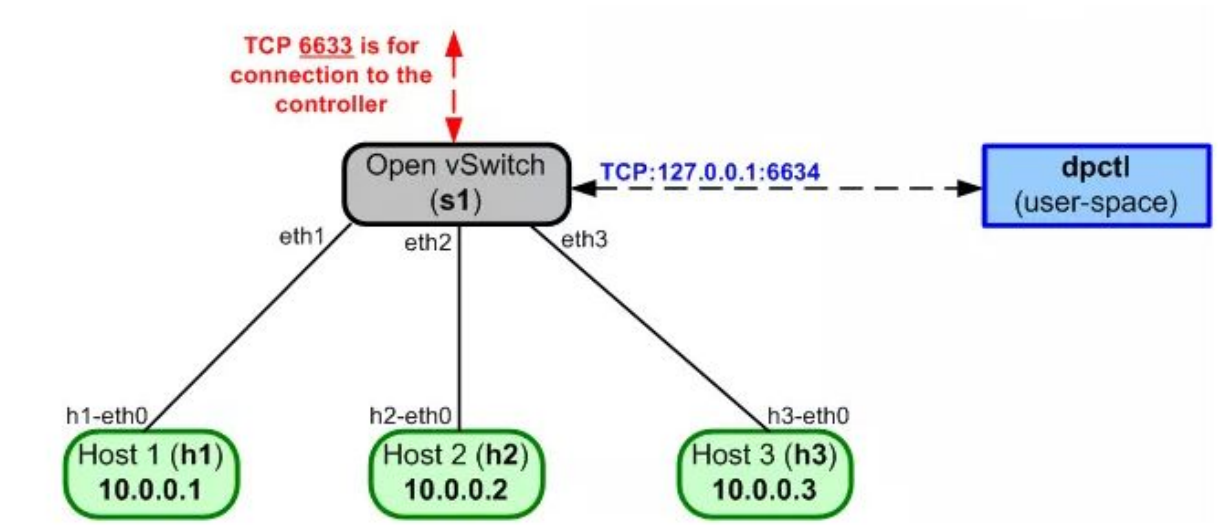
No entanto, os comandos acima administram apenas *switches* Open vSwitch (virtuais) e podemos utilizar um utilitário para interação com *switches* OpenFlow em geral, chamado **dpctl:**

- **dpctl**: É um utilitário incluído na distribuição de referência OpenFlow que permite a visualização e o controle sobre a tabela de fluxos de um *switch*. Ele é especialmente útil para identificar erros, pois permite exibir o estado e os contadores dos fluxos. A maioria dos *switches* OpenFlow inicia com uma porta de escuta (por default a porta 6634), na qual é possível interagir com o *switch* - como obter suas regras de fluxo - sem ter que passar pelo controlador.

Apesar da sintaxe dos comandos do Open vSwitch serem mais simples, iremos utilizar o **dpctl** devido a ser mais genérico e compatível com qualquer tipo de *switch* OpenFlow.

Exercício

Nesse exercício iremos criar uma rede com um *switch* e 3 *hosts* e utilizar o comando **dpctl** para configurar as regras de fluxos no *switch* e realizar um ping de h1 para h2



Utilize o seguinte comando para subir uma rede Mininet com a configuração acima

```
#> sudo mn --topo=single,3 --mac --switch=ovsk --controller=remote
```

O parâmetro **'show'** conecta ao *switch* e exibe o estado e capacidade das portas:

```
#> dpctl show tcp:127.0.0.1:6634
features_reply (xid=0xe1a4c560): ver:0x1, dpid:1
n_tables:254, n_buffers:256
features: capabilities:0xc7, actions:0xfff
 1(s1-eth1): addr:5a:a6:3a:ce:b3:da, config: 0, state:0
     current:    10GB-FD COPPER
 2(s1-eth2): addr:f6:92:c6:b4:af:4b, config: 0, state:0
     current:    10GB-FD COPPER
 3(s1-eth3): addr:22:52:79:4c:34:0b, config: 0, state:0
     current:    10GB-FD COPPER
 LOCAL(s1): addr:e6:b1:92:cb:1c:45, config: 0, state:0
get_config_reply (xid=0x41097735): miss_send_len=0
```

Após conectar ao *switch,* podemos utilizar o comando **dpctl flows** para listar a tabela de fluxos em execução no momento.

Repare que não há nenhum fluxo criado ainda:

```
#> dpctl dump-flows tcp:127.0.0.1:6634
stats_reply (xid=0xb92b48c7): flags=none type=1(flow)
```

Como a tabela de fluxos está vazia, os *hosts* não conseguirão comunicar entre si. Iremos popular a tabela com duas regras para estabelecer a comunicação entre os *hosts* h1 e h2.

```
#> dpctl add-flow tcp:127.0.0.1:6634 in_port=1,idle_timeout=150,actions=output:2

#> dpctl add-flow tcp:127.0.0.1:6634 in_port=2,idle_timeout=150,actions=output:1
```

A primeira regra define que todos os fluxos recebidos na porta 1 do switch devem ser encaminhados para a porta 2. Por padrão, uma regra irá expirar se não ocorrer nenhum *match* (*idle*) por 60 segundos, sendo o parâmetro *idle_timeout* utilizado para alterar esse tempo para 150 segundos.

Verifique as regras novamente com o comando **dump-flows** e veja que elas foram criadas com sucesso.

```
#> dpctl dump-flows tcp:127.0.0.1:6634
stats_reply (xid=0x331e663): flags=none type=1(flow)
  cookie=0, duration_sec=19s, duration_nsec=655000000s, table_id=0,
priority=32768, n_packets=0, n_bytes=0,
idle_timeout=150,hard_timeout=0,in_port=1,actions=output:2
  cookie=0, duration_sec=11s, duration_nsec=884000000s, table_id=0,
priority=32768, n_packets=0, n_bytes=0,
idle_timeout=150,hard_timeout=0,in_port=2,actions=output:1
```

Podemos verificar que as regras funcionaram, realizando um ping entre h1 e h2

```
mininet> h1 ping h2 -c 3
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.159 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.047 ms
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
```

Para finalizar, iremos consultar o número de *bytes* encaminhados pelas regras. Para isso, primeiramente enviaremos 20MB de dados com o auxílio do comando **iperf** e, em seguida, verificaremos esse valor através do comando **dump-flows**.

Pela CLI do Mininet, abra um terminal em h1 para rodar o servidor de *iperf* e execute o *iperf* cliente em h2 através da própria CLI do Mininet, conforme mostrado a seguir:

```
mininet> xterm h1
h1> iperf -s -u
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[ 16] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 44635
[ 17] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 39384
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[ 17]  0.0-483.7 sec  23.8 MBytes   413 Kbits/sec   0.005 ms    0/17007 (0%)
[ 17]  0.0-483.7 sec  1 datagrams received out-of-order
```

```
mininet> h2 iperf -c 10.0.0.1 -u -b 20m
------------------------------------------------------------
Client connecting to 10.0.0.1, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.2 port 39384 connected with 10.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  23.8 MBytes  20.0 Mbits/sec
[  3] Sent 17008 datagrams
[  3] Server Report:
[  3]  0.0-483.7 sec  23.8 MBytes   413 Kbits/sec   0.005 ms    0/17007 (0%)
[  3]  0.0-483.7 sec  1 datagrams received out-of-order
```

Verifique, pelo contador de bytes das regras, que aproximadamente 20Mb foram encaminhados (**n_bytes**=25718224):

```
mininet@mininet-vm:~$ dpctl dump-flows tcp:127.0.0.1:6634
stats_reply (xid=0x29b09d67): flags=none type=1(flow)
  cookie=0, duration_sec=84s, duration_nsec=331000000s, table_id=0,
priority=32768, n_packets=9, n_bytes=2128,
idle_timeout=150,hard_timeout=180,in_port=1,actions=output:2

  cookie=0, duration_sec=78s, duration_nsec=919000000s, table_id=0,
priority=32768, n_packets=17017, n_bytes=25718224,
idle_timeout=150,hard_timeout=180,in_port=2,actions=output:1
```

Há muitos outros recursos disponíveis com os comandos do OpenFlow, para maiores informações, consulte o endereço a seguir:

http://ranosgrant.cocolog-nifty.com/openflow/dpctl.8.html

Finalizando a prática

Encerre a execução do Mininet para encerrar a prática:

```
mininet> exit
```