Relatório do EP3
MAC0352 – Redes de Computadores e Sistemas Distribuídos – 2/2019

Matheus Barcellos de Castro Cunha (11208238), Breno Helfstein Moura (9790972)

1 Passo 0

Dentro do contexto de funcionamento do protocolo OpenFlow, o *switch* não tem o poder de decisão do caminho o qual os pacotes irão seguir na rede. Então, sempre que um *switch* recebe um pacote novo há uma troca de mensagens por meio do protocolo OpenFlow entre o *switch* e o controlador. Ao receber as informações do pacote recebido pelo *switch*, o controlador tem total autonomia para decidir qual caminho na rede o pacote tomará, e assim essa decisão será enviada para o *switch* para que o pacote possa ser direcionado na rede.

2 Passo 2

```
<traceroute to www.inria.fr (128.93.162.84), 30 hops max,
60 byte packets
 1  10.0.2.2 (10.0.2.2)  0.251 ms  0.245 ms  0.257 ms

 2  www.instaladorvivofibra.br (192.168.15.1)  2.994 ms
 3.003 ms  2.999 ms

 3  * * *

 4  187-100-179-44.dsl.telesp.net.br (187.100.179.44)  6.551 ms
6.554 ms  6.555 ms

 5  187-100-197-252.dsl.telesp.net.br (187.100.197.252)
21.557 ms  21.565 ms  21.578 ms

 6  152-255-158-26.user.vivozap.com.br (152.255.158.26)  6.311 ms
 4.745 ms  4.385 ms

 7  84.16.9.109 (84.16.9.109)  13.686 ms  13.736 ms  13.213 ms

 8  94.142.98.177 (94.142.98.177)  118.356 ms  118.324 ms  118.304 ms
```

```
 9  84.16.15.129 (84.16.15.129)  119.157 ms  119.157 ms  119.145 ms

10  213.140.36.89 (213.140.36.89)  124.501 ms  122.227 ms  122.223 ms

11  ip4.gtt.net (208.116.240.149)  119.864 ms  118.424 ms  117.965 ms

12  et-3-3-0.cr4-par7.ip4.gtt.net (213.200.119.214)  212.829 ms  244.825 ms
238.006 ms

13  renater-gw-ix1.gtt.net (77.67.123.206)  233.874 ms  232.856 ms
232.101 ms

14  te1-1-inria-rtr-021.noc.renater.fr (193.51.177.107)  232.139 ms
232.258 ms 232.304 ms
15  inria-rocquencourt-te1-4-inria-rtr-021.noc.renater.fr (193.51.184.177)
228.374 ms  226.059 ms  225.240 ms

16  unit240-reth1-vfw-ext-dc1.inria.fr (192.93.122.19)  224.212 ms
213.279 ms  406.979 ms

17  www.inria.fr (128.93.162.84)  406.441 ms  405.644 ms  405.269 ms>
```

Pela saída do comando “traceroute -I www.inria.fr”o primeiro acesso a um roteador europeu foi no 7 salto no IP 84.16.9.109, o qual localiza-se na Espanha. Foi possível descobrir essa informação por meio do uso do comando “whois”no endereços de IP mostrados pelo “traceroute”.

3 Passo 3 - Parte 1

Valor médio da taxa de transferência de dados sem a opção “–switch user”: 2,95 Gbits/sec. Nível de confiança: 95%. Intervalo de confiança: $2,95 \pm 0,308$.

4 Passo 3 - Parte 2

Valor médio da taxa de transferência de dados com a opção “–switch user”: 29,48 Mbits/sec. Nível de confiança: 95%. Intervalo de confiança: $29,48 \pm 0,608$.

Essa diferença notável de mais de 121 vezes menor no valor da taxa de transferência media de dados deve-se ao fato de ao usar a opção “–switch user”os pacotes precisam passar do *user-space* pro *kernel-space* e voltar a cada pulo, o que sem a opção “–switch user”não precisava já que os pacotes ficavam pelo *kernel-space*.

5 Passo 4 - Parte 1

Valor médio da taxa de transferência de dados com a opção “–switch user”: 5,56 Mbits/sec. Nível de confiança: 95%. Intervalo de confiança: $5,56 \pm 0,292$.

O valor médio da taxa de transferência de dados é mais de 5 vezes menor nesse caso, isso porque agora todos os pacotes estão passando pelo controlador e todos os *hosts*.

```
<Host 1:

12:32:58.485591 IP 10.0.0.1.59280 > 10.0.0.3.5001: Flags [S], seq 1948985874,
w$
        0x0000:  0000 0000 0003 0000 0000 0001 0800 4510  ..............E.
        0x0010:  003c 5c47 4000 4006 ca61 0a00 0001 0a00  .<\G@.@..a......
        0x0020:  0003 e790 1389 742b 2a12 0000 0000 a002  ......t+*.......
        0x0030:  7210 1432 0000 0204 05b4 0402 080a 0005  r..2............
        0x0040:  7655 0000 0000 0103 0309
                                          .
                                          .
                                          .                                >

<Host 2:

12:32:58.533854 IP 10.0.0.1.59280 > 10.0.0.3.5001: Flags [S], seq 1948985874,
w$
        0x0000:  0000 0000 0003 0000 0000 0001 0800 4510  ..............E.
        0x0010:  003c 5c47 4000 4006 ca61 0a00 0001 0a00  .<\G@.@..a......
        0x0020:  0003 e790 1389 742b 2a12 0000 0000 a002  ......t+*.......
        0x0030:  7210 b238 0000 0204 05b4 0402 080a 0005  r..8............
        0x0040:  7655 0000 0000 0103 0309
                                          .
                                          .
                                          .                                >

<Host 3:
112:32:58.533820 IP 10.0.0.1.59280 > 10.0.0.3.5001: Flags [S], seq 1948985874,
w$
        0x0000:  0000 0000 0003 0000 0000 0001 0800 4510  ..............E.
        0x0010:  003c 5c47 4000 4006 ca61 0a00 0001 0a00  .<\G@.@..a......
        0x0020:  0003 e790 1389 742b 2a12 0000 0000 a002  ......t+*.......
        0x0030:  7210 b238 0000 0204 05b4 0402 080a 0005  r..8............
        0x0040:  7655 0000 0000 0103 0309
                                          .
                                          .
                                          .                                >
```

6 Passo 4 - Parte 2

Valor médio da taxa de transferência de dados: 9,59 Mbits/sec. Nível de confiança: 95%. Intervalo de confiança: $9,59 \pm 0,366$.

O resultado corresponde 1,72 vezes o resultado da seção anterior. Isso é efeito do fato de os pacotes não passarem mais por todos os *hosts* conectados ao *swtich*.

Host 1:

```
<02:54:36.340947 IP 10.0.0.1.46496 > 10.0.0.3.5001: Flags [S], seq 3923196604,
win 29200, options [mss 1460,sackOK,TS val 10991016 ecr 0,nop,wscale 9],
length 0
    0x0000:  0000 0000 0003 0000 0000 0001 0800 4510  ..............E.
    0x0010:  003c 8f14 4000 4006 9794 0a00 0001 0a00  .<..@.@.........
    0x0020:  0003 b5a0 1389 e9d7 3abc 0000 0000 a002  ........:.......
    0x0030:  7210 1432 0000 0204 05b4 0402 080a 00a7  r..2............
    0x0040:  b5a8 0000 0000 0103 0309
                                      .
                                      .
                                      .

3567 packets captured
3567 packets received by filter
0 packets dropped by kernel>
```

Host 2:

```
<0 packets captured
0 packets received by filter
0 packets dropped by kernel>
```

Host 3:

```
<02:54:36.345111 IP 10.0.0.1.46496 > 10.0.0.3.5001: Flags [S], seq 3923196604,
win 29200, options [mss 1460,sackOK,TS val 10991016 ecr 0,nop,wscale 9],
length 0
    0x0000:  0000 0000 0003 0000 0000 0001 0800 4510  ..............E.
    0x0010:  003c 8f14 4000 4006 9794 0a00 0001 0a00  .<..@.@.........
    0x0020:  0003 b5a0 1389 e9d7 3abc 0000 0000 a002  ........:.......
    0x0030:  7210 1ddd 0000 0204 05b4 0402 080a 00a7  r...............
    0x0040:  b5a8 0000 0000 0103 0309
                                      .
                                      .
                                      .

5275 packets captured
5275 packets received by filter
0 packets dropped by kernel>
```

7 Passo 4 - Parte 3

Valor médio da taxa de transferência de dados: 3,04 Gbits/sec. Nível de confiança: 95%. Intervalo de confiança: $3,04 \pm 0,37$.

O resultado corresponde a aproximadamente 324 vezes o resultado da seção anterior. Esse aumento no valor médio deve-se ao fato de que na versão melhorada do *switch* o controlador "impulsiona fluxo"para enviar os pacotes ao invés de reenvia-los.

Host 1:

```
<02:45:39.395597 IP 10.0.0.1.46478 > 10.0.0.3.5001: Flags [S], seq 3683688427,
win 29200, options [mss 1460,sackOK,TS val 10856780 ecr 0,nop,wscale 9],
length 0
    0x0000:  0000 0000 0003 0000 0000 0001 0800 4510  ..............E.
    0x0010:  003c 36a7 4000 4006 f001 0a00 0001 0a00  .<6.@.@.........
    0x0020:  0003 b58e 1389 db90 9feb 0000 0000 a002  ................
    0x0030:  7210 1432 0000 0204 05b4 0402 080a 00a5  r..2............
    0x0040:  a94c 0000 0000 0103 0309                 .L........
                                    .
                                    .
                                    .
672 packets captured
18462 packets received by filter
17790 packets dropped by kernel>
```

Host 2:

```
<0 packets captured
0 packets received by filter
0 packets dropped by kernel>
```

Host 3:

```
<02:45:39.410819 IP 10.0.0.1.46478 > 10.0.0.3.5001: Flags [S], seq 3683688427,
win 29200, options [mss 1460,sackOK,TS val 10856780 ecr 0,nop,wscale 9],
length 0
    0x0000:  0000 0000 0003 0000 0000 0001 0800 4510  ..............E.
    0x0010:  003c 36a7 4000 4006 f001 0a00 0001 0a00  .<6.@.@.........
    0x0020:  0003 b58e 1389 db90 9feb 0000 0000 a002  ................
    0x0030:  7210 d364 0000 0204 05b4 0402 080a 00a5  r..d............
    0x0040:  a94c 0000 0000 0103 0309                 .L........
                                    .
                                    .
                                    .
655 packets captured
18462 packets received by filter
17807 packets dropped by kernel>
```

ovs-ofctl dump-flows s1:

```
<NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=1.385s, table=0, n_packets=2454, n_bytes=161972,
 idle_timeout=5, idle_age=1, tcp,vlan_tci=0x0000,dl_src=00:00:00:00:00:03,
```

```
dl_dst=00:00:00:00:00:01,nw_src=10.0.0.3,nw_dst=10.0.0.1,nw_tos=0,
tp_src=5001,tp_dst=46378 actions=output:1

cookie=0x0, duration=1.421s, table=0, n_packets=3, n_bytes=206,
idle_timeout=5, idle_age=1, tcp,vlan_tci=0x0000,dl_src=00:00:00:00:00:03,
dl_dst=00:00:00:00:00:01, nw_src=10.0.0.3,nw_dst=10.0.0.1, nw_tos=0,
tp_src=5001, tp_dst=46376 actions=output:1

cookie=0x0, duration=1.387s, table=0, n_packets=8430, n_bytes=468531956,
idle_timeout=5, idle_age=0, tcp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,
dl_dst=00:00:00:00:00:03, nw_src=10.0.0.1,nw_dst=10.0.0.3,nw_tos=0,
tp_src=46378, tp_dst=5001 actions=output:3

cookie=0x0, duration=1.423s, table=0, n_packets=4, n_bytes=272,
idle_timeout=5, idle_age=1, tcp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,
dl_dst=00:00:00:00:00:03, nw_src=10.0.0.1,nw_dst=10.0.0.3,nw_tos=16,
tp_src=46376, tp_dst=5001 actions=output:3>
```

8 Passo 5

8.1 Lógica do controlador:

Sempre que o controlador recebe a informação de um pacote, o firewall faz a checagem para tomar conhecimento dos campos de IP origem, IP destino, protocolo e porta a fim de saber se o pacote deve ou não seguir na rede. Caso o pacote deva seguir na rede, é feita uma checagem para saber se a porta de destino do switch já é conhecida (por meio do endereço MAC), caso não seja, o pacote é enviado para todos os *hosts* conectados ao *switch*, caso contrario o pacote é enviado somente a porta atrelada ao MAC de destino.

8.2 Regras de descarte de pacotes:

8.2.1 Bloquear protocolo:

Para bloquear pacotes com base no protocolo, é feita uma checagem por meio de "packet.find(PROTOCOL_TYPE)". Caso o retorno seja diferente de "None"então o pacote é do protocolo que deve ser bloqueado, portanto ele é descartado.

Nesse exemplo tem-se o funcionamento do bloqueio de pacotes por tipo de protocolo, neste caso bloqueio de pacotes TCP. Após usar o comando "iperf", pode ser visto no retorno do "tcpdump"que o único pacote que o *host 2* recebe é um pacote ARP, isso porque todos os pacotes TCP que tentaram ser enviados pelo *host 1* foram bloqueados.

Host 1:

```
<06:45:24.899026 IP 10.0.0.1.49398 > 10.0.0.3.5001: Flags [S], seq 2317308505,
win 29200, options [mss 1460,sackOK,TS val 14453156 ecr 0,nop,wscale 9],
length 0
```

```
     0x0000:  0000 0000 0003 0000 0000 0001 0800 4510  ..............E.
     0x0010:  003c dbfd 4000 4006 4aab 0a00 0001 0a00  .<..@.@.J.......
     0x0020:  0003 c0f6 1389 8a1f 5259 0000 0000 a002  ........RY......
     0x0030:  7210 1432 0000 0204 05b4 0402 080a 00dc  r..2............
     0x0040:  89a4 0000 0000 0103 0309                 ..........
06:45:25.897530 IP 10.0.0.1.49398 > 10.0.0.3.5001: Flags [S], seq 2317308505,
win 29200, options [mss 1460,sackOK,TS val 14453406 ecr 0,nop,wscale 9],
length 0
     0x0000:  0000 0000 0003 0000 0000 0001 0800 4510  ..............E.
     0x0010:  003c dbfe 4000 4006 4aaa 0a00 0001 0a00  .<..@.@.J.......
     0x0020:  0003 c0f6 1389 8a1f 5259 0000 0000 a002  ........RY......
     0x0030:  7210 1432 0000 0204 05b4 0402 080a 00dc  r..2............
     0x0040:  8a9e 0000 0000 0103 0309                 ..........
06:45:27.905975 IP 10.0.0.1.49398 > 10.0.0.3.5001: Flags [S], seq 2317308505,
win 29200, options [mss 1460,sackOK,TS val 14453907 ecr 0,nop,wscale 9],
length 0
     0x0000:  0000 0000 0003 0000 0000 0001 0800 4510  ..............E.
     0x0010:  003c dbff 4000 4006 4aa9 0a00 0001 0a00  .<..@.@.J.......
     0x0020:  0003 c0f6 1389 8a1f 5259 0000 0000 a002  ........RY......
     0x0030:  7210 1432 0000 0204 05b4 0402 080a 00dc  r..2............
     0x0040:  8c93 0000 0000 0103 0309                 ..........
```

.
.
.

```
7 packets captured
8 packets received by filter
0 packets dropped by kernel>
```

Host 3:

```
<06:45:29.956783 ARP, Request who-has 10.0.0.3 tell 10.0.0.1, length 28
     0x0000:  0000 0000 0003 0000 0000 0001 0806 0001  ................
     0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
     0x0020:  0000 0000 0000 0a00 0003                 ..........
06:45:29.959046 ARP, Reply 10.0.0.3 is-at 00:00:00:00:00:03, length 28
     0x0000:  0000 0000 0001 0000 0000 0003 0806 0001  ................
     0x0010:  0800 0604 0002 0000 0000 0003 0a00 0003  ................
     0x0020:  0000 0000 0001 0a00 0001                 ..........

1 packets captured
1 packets received by filter
0 packets dropped by kernel>
```

8.2.2 Bloquear IP:

Para fazer a checagem dos endereços de IP dos pacotes, primeiro é feita a checagem se o pacote é um pacote IPV4 por meio do retorno de "packet.find("ipv4")". Caso o valor retornado seja diferente de "None"então o pacote é IPV4 e são checados os campos ".dstip"e ".srcip"para saber se os endereços de destino e fonte são iguais aos endereços definidos para bloqueio, assim bloqueado todos os pacotes que trafegam entre os IPs definidos para descarte.

Nesse exemplo tem-se o funcionamento do bloqueio de pacotes por endereço de IP, neste caso bloqueio de pacotes que tem como destino e fonte e vice-versa os endereços "10.0.0.1"e "10.0.0.2". O retorno do comando "pingall"tem como X o resultado da tentativa da comunicação entre os *hosts* 1 e 2, isso porque eles são endereçados com os IPs bloqueados.

```
<h1 -> X h3
h2 -> X h3
h3 -> h1 h2
*** Results: 33% dropped (4/6 received)>
```

8.2.3 Bloquear Porta:

Para fazer a checagem da porta da camada de transporte, o algoritmo utiliza a classe "ofp_match", mais especificamente, armazena o retorno de "of.ofp_match.from_packet(packet)"e checa os campos "tp_dst"e "tp_src"do que foi retornado. Caso alguma das portas corresponda ao valor bloqueado, o pacote não segue em frente na rede.

Usando "tcpdump -s -p 20202 -i 1"para iniciar um server TCP no *host 1* escutando na porta 20202 e "tcpdump 10.0.0.1 -p 20202 -t 15"para conectar o *host 2* ao *host 1* na porta 20202, vemos que ocorre a troca de pacotes de maneira normal.

Host 1(Server):

```
<-----------------------------------------------------------
Server listening on TCP port 20202
TCP window size: 85.3 KByte (default)
-----------------------------------------------------------
[ 16] local 10.0.0.1 port 20202 connected with 10.0.0.2 port 54362
[ ID] Interval       Transfer     Bandwidth
[ 16]  0.0- 1.0 sec   279 MBytes  2.34 Gbits/sec
[ 16]  1.0- 2.0 sec   298 MBytes  2.50 Gbits/sec
[ 16]  2.0- 3.0 sec   201 MBytes  1.69 Gbits/sec
                        .
                        .
                        .                          >
```

Host 2(Client):

```
<-----------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 20202
TCP window size: 85.3 KByte (default)
-----------------------------------------------------------
[ 15] local 10.0.0.2 port 54362 connected with 10.0.0.1 port 20202
```

```
[ ID] Interval       Transfer     Bandwidth
[ 15]  0.0-11.7 sec  3.30 GBytes  2.43 Gbits/sec
                          .
                          .
                          .                                          >
```

Nesse exemplo tem-se o funcionamento do bloqueio de pacotes por porta da camada de transporte, neste caso bloqueio de pacotes que tem como porta de destino ou fonte “20202”. O retorno dos comandos “iperf”:

Host 1(Server):

```
<-----------------------------------------------------------
Server listening on TCP port 20202
TCP window size: 85.3 KByte (default)
-----------------------------------------------------------
                          .
                          .
                          .                                      >
```

Host 2(Client): Não foi obtido qualquer resultado pois não conseguiu estabelecer uma conexão.

9 Configuração dos computadores virtual e real usados nas medições (se foi usado mais de um, especifique qual passo foi feito com cada um)

9.1 Real

- Intel® Core™ i5-7200U (2.5 GHz, up to 3.1 GHz, 3 MB cache, 2 cores)
- 8 GB DDR4-2133 SDRAM (1 x 8 GB)
- SSD 240 GB SA400S37
- Ubuntu 18.04.3 LTS

9.2 Virtual

- Ubuntu 14.04.4 LTS
- Base Memory: 1024 MB
- Virtual Size: 8,00 GB

10 Referências

- https://openflow.stanford.edu/display/ONL/POX+Wiki.htmlPOXWiki-SetEthernetsourceordestinationaddress
- https://openflow.stanford.edu/display/ONL/POX+Wiki.html
- http://sdnhub.org/tutorials/pox/