

Nomes: Ricardo Moreira Pires, Felipe Izaguirre

## **Ambiente**

O módulo para o controlador SDN construído foi feito em java, utilizando o controlador FloodLight. Os testes foram executados sobre o ambiente da VM do Mininet.

Utilizando o controlador FloodLight criamos um novo pacote chamado `net.floodlight.policyforward`.

Definimos uma classe chamada `PolicyForward` que herda da classe `ForwardBase`, classe abstrata apropriada para gerenciar o encaminhamento de pacotes na rede. Implementamos também 3 interfaces sendo elas `IOFMessageListener`, `IFloodlightModule`, `ILinkDiscoveryListener`.

`IFloodlightModule` foi utilizado pois queríamos implementar um módulo adicional ao controlador, sendo então obrigatório o uso desta interface. É com ela que temos acesso aos outros serviços do controlador e inicializamos corretamente o nosso módulo.

A interface `IOFMessageListener` é utilizada para capturar pacotes sem flow que são enviados ao controlador. Sempre que um pacote chega para análise, cada módulo que implementa esta interface é notificado da chegada do pacote através da função `receive()`.

A `ILinkDiscoveryListener` seria utilizada para tratar mudanças na topologia, mas acabou sendo pouco utilizada durante o desenvolvimento.

Também fizemos uso dos serviços `TopologyManager` para a gerência da topologia e `StatisticService` para coletar as estatísticas de tráfego na rede.

## **Funcionamento**

Através do serviço `TopologyManager` temos acesso a topologia, esta topologia vai sendo formada através de notificações do módulo de LLDP tratadas pelo `TopologyManager`. Na inicialização do módulo também colocamos para executar uma thread que fica coletando informações sobre tráfego na rede e guarda e um histórico, escolhemos esta abordagem pois uma chamada ao serviço de estatística é lento, o que acabava causando problemas de tratamento de pacotes da rede, atrasando a adição de flows e causando dificuldades no uso.

A principal função do módulo é tratar os pacotes que chegam e não tem flow associados, definindo o que deve ser feito e qual o melhor caminho. Para isto ao receber um pacote verificamos se ele é do tipo ARP ou IPv4. Se for do tipo ARP executamos um broadcast do pacote na rede, caso contrário, sendo o pacote do tipo IPv4, buscamos os nodos de origem e destino e através do `TopologyManager`. O número máximo de caminhos foi pré definido em 5, dentro destes caminhos calculamos o melhor seguindo as métricas do algoritmo descrito na próxima sessão. Após um caminho ser escolhido é adicionado os Flows nos switches.

## Balanceamento de carga

O controlador Policyforward utiliza a seguinte lógica para realizar um uso eficiente da rede:

- Cálculo da topologia de rede e de até 5 rotas para cada par de hosts, se existente;
- Armazena a banda consumida em cada enlace, para ser utilizada na escolha do caminho;
- Ao chegar um pacote que não esteja associado um flow e não seja broadcast, é encontrado a melhor rota utilizando a função utilidade descrita mais adiante e criado o respectivo flow nos switches que compõem o caminho;
- Depois de criado o flow, o pacote é encaminhado para o próximo hop.

O cálculo dos melhores caminhos é feito utilizando o algoritmo do Yen's k-shortest paths, disponível no floodlight. A métrica para geração de caminhos escolhida foi a quantidade de hops.

Quanto a coleta de banda, a cada 3 segundos é coletado a quantidade de bits por segundo de cada enlace e guardado num histórico. Este histórico de cada enlace é utilizado como base para cálculo das rotas, de forma a tentar evitar congestionamentos. Atualmente é armazenado as 5 amostras de cada enlace mais atuais.

Dispondo dessas informações, o cálculo do valor utilidade ( $V$ ) de um caminho é calculado da seguinte forma:

$$X = \sum M(Ei)$$

$$V = X + a * h$$

O caminho escolhido é aquele com menor  $V$ , onde:

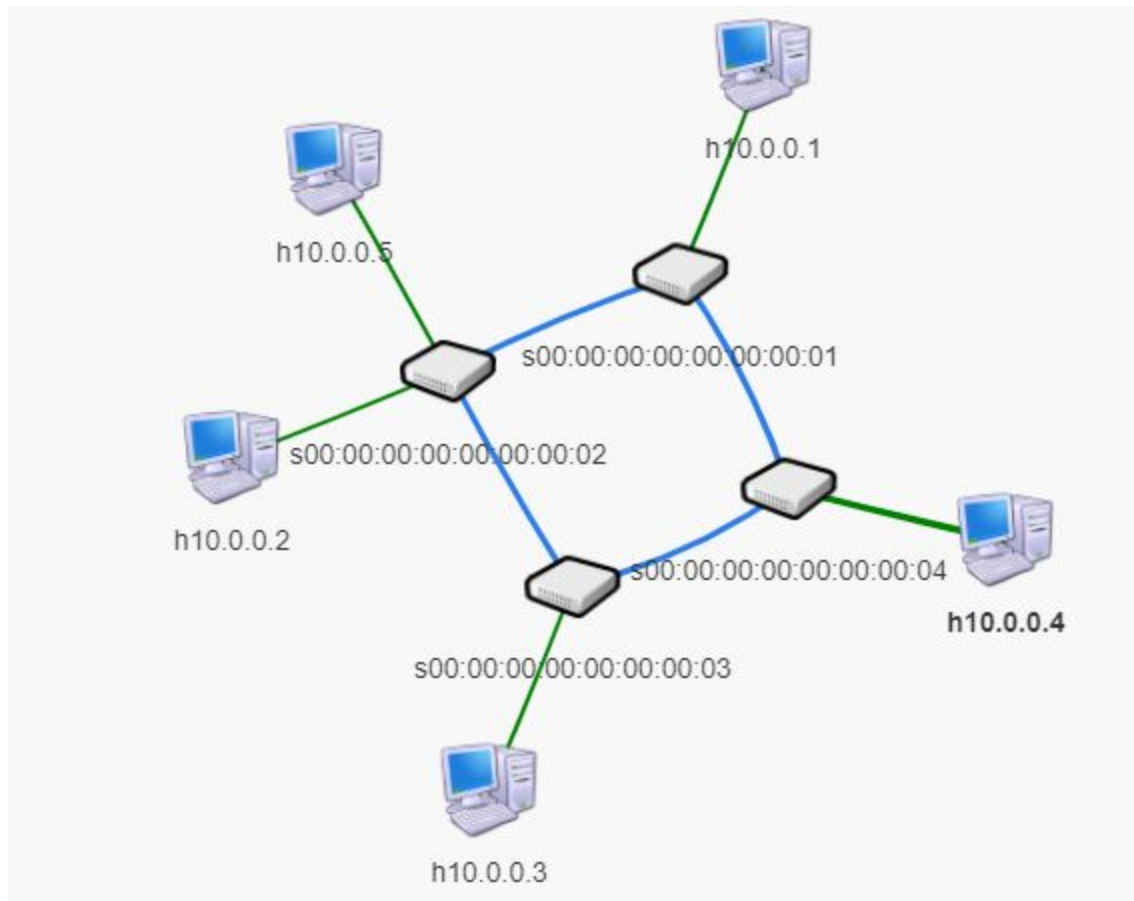
- $h$  é a quantidade de hops na rota;
- $a$  é uma constante definida pelo usuário (default = 10000);
- $M(Ei)$  é a média aritmética das amostras dos enlaces que pertencem a o caminho.

O fator  $a * h$  é utilizado para priorizar caminhos menores que possuam pouco tráfego ao invés de maiores que não estejam ainda sendo utilizados.

## Testes

### Caso 1

Considere a topologia abaixo:



Onde:

sw1 : 00:00:00:00:00:00:01

sw2 : 00:00:00:00:00:00:02

sw3 : 00:00:00:00:00:00:03

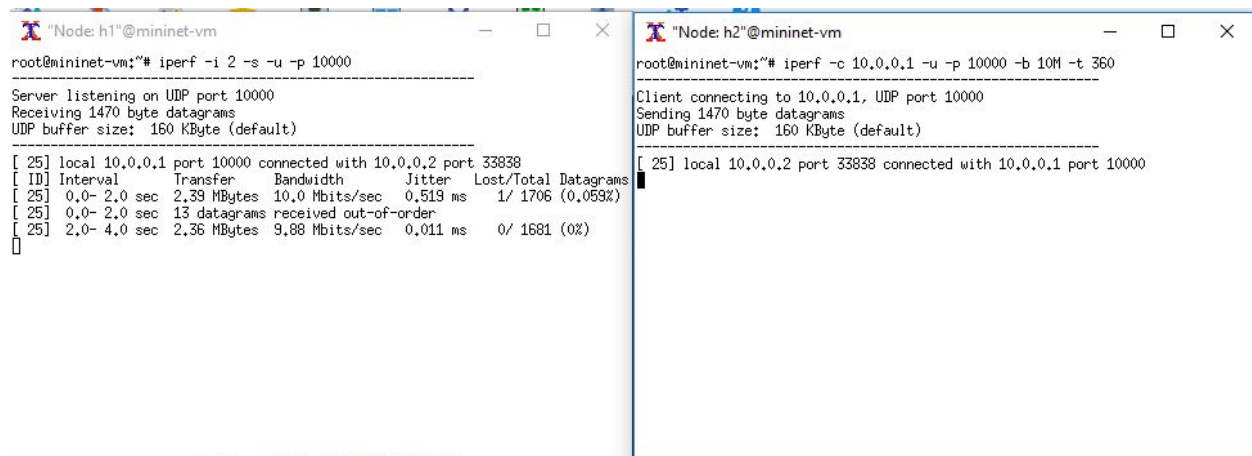
sw4 : 00:00:00:00:00:00:04

h1 : 00:00:00:00:00:01

h2 : 00:00:00:00:00:02

h5 : 00:00:00:00:00:05

Disparando iperf do host h2 -> h1:



The image shows two terminal windows side-by-side. The left window is titled '"Node: h1"@mininet-vm' and shows the output of 'iperf -i 2 -s -u -p 10000'. It indicates the server is listening on UDP port 10000 and has received 1470 byte datagrams. The right window is titled '"Node: h2"@mininet-vm' and shows the output of 'iperf -c 10.0.0.1 -u -p 10000 -b 10M -t 360'. It indicates the client is connecting to 10.0.0.1, UDP port 10000, and is sending 1470 byte datagrams. Both windows show a table of performance metrics over three intervals (0.0-2.0 sec, 2.0-4.0 sec).

```
"Node: h1"@mininet-vm
root@mininet-vm:~# iperf -i 2 -s -u -p 10000
-----
Server listening on UDP port 10000
Receiving 1470 byte datagrams
UDP buffer size: 160 KByte (default)
-----
[ 25] local 10.0.0.1 port 10000 connected with 10.0.0.2 port 33838
[ ID] Interval      Transfer    Bandwidth    Jitter    Lost/Totl  Datagrams
[ 25] 0.0- 2.0 sec  2.39 MBytes 10.0 Mbits/sec  0.519 ms   1/ 1706 (0.059%)
[ 25] 2.0- 4.0 sec  13 datagrams received out-of-order
[ 25] 2.0- 4.0 sec  2.36 MBytes  9.88 Mbits/sec  0.011 ms   0/ 1681 (0%)

"Node: h2"@mininet-vm
root@mininet-vm:~# iperf -c 10.0.0.1 -u -p 10000 -b 10M -t 360
-----
Client connecting to 10.0.0.1, UDP port 10000
Sending 1470 byte datagrams
UDP buffer size: 160 KByte (default)
-----
[ 25] local 10.0.0.2 port 33838 connected with 10.0.0.1 port 10000
```

O controlador encontrou a seguinte rota:

```
2017-07-02 22:57:07.429 INFO [n.f.p.PolicyForward] Path Candidate 1 Utilization 60698
2017-07-02 22:57:07.429 INFO [n.f.p.PolicyForward] === Path 0 chosed ===
2017-07-02 22:57:07.429 INFO [n.f.p.PolicyForward] Path:  -> sw 2 -> sw 2 -> sw 1 -> sw 1
2017-07-02 22:57:07.429 INFO [n.f.p.PolicyForward]
```

Agora, disparando h5 -> h1 :

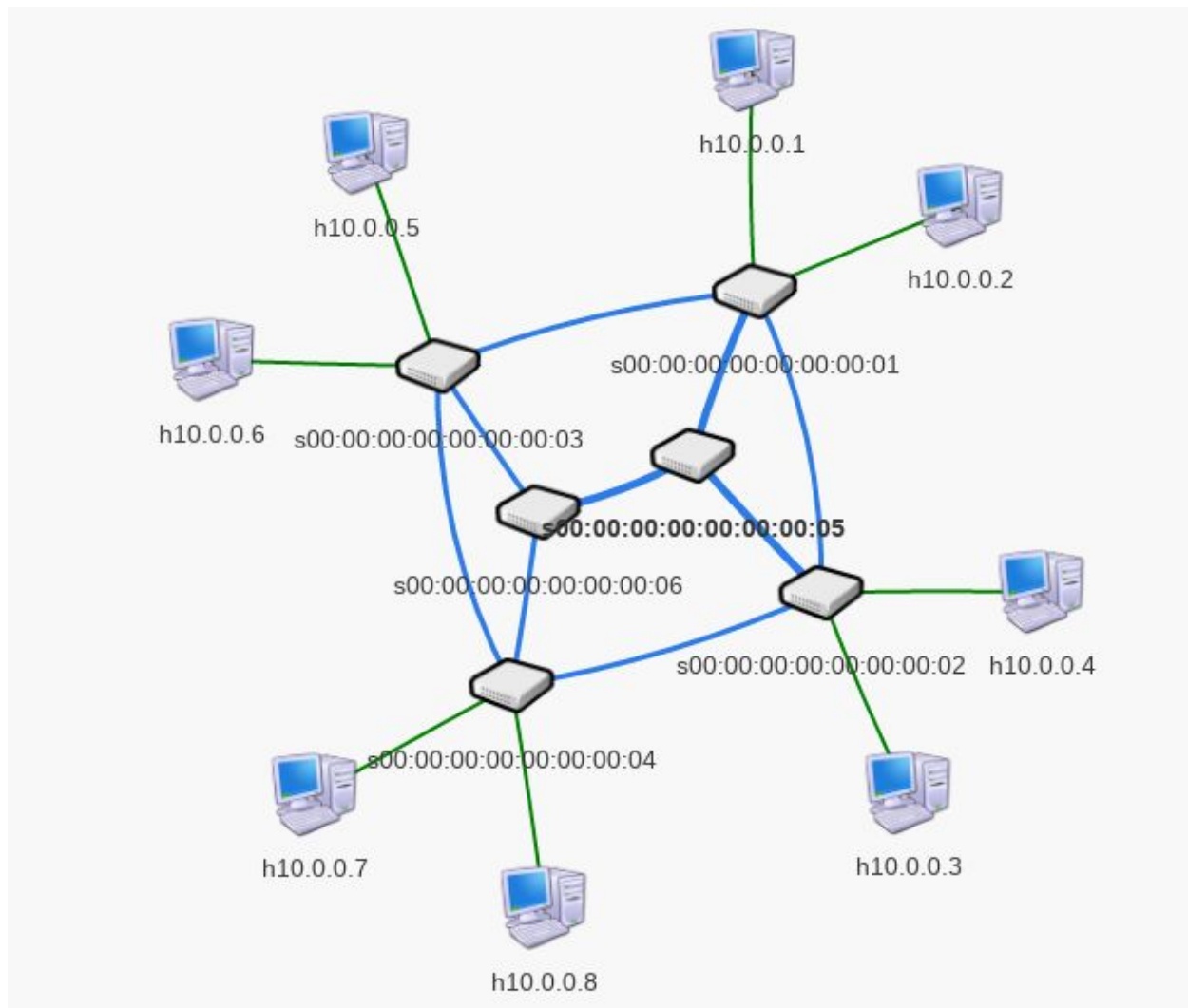
```
"Node: h5"@mininet-vm
root@mininet-vm:~# iperf -c 10.0.0.1 -u -p 10000 -b 10M -t 360
-----
Client connecting to 10.0.0.1, UDP port 10000
Sending 1470 byte datagrams
UDP buffer size: 160 KByte (default)
-----
[ 25] local 10.0.0.5 port 46003 connected with 10.0.0.1 port 10000
█
```

Rota encontrada(utilidade 61043):

```
.142 INFO [n.f.p.PolicyForward] Path Candidate 1 Utilization 61043
.143 INFO [n.f.p.PolicyForward] === Path 1 chosed ===
.145 INFO [n.f.p.PolicyForward] Path: -> sw 2 -> sw 2 -> sw 3 -> sw 3 -> sw 4 -> sw 4 -> sw 1 -> sw 1
.145 INFO [n.f.p.PolicyForward] =====
```

## Caso 2

Considerando a topologia abaixo:



Onde :

sw1 : 00:00:00:00:00:00:01

sw2 : 00:00:00:00:00:00:02

sw3 : 00:00:00:00:00:00:03

sw4 : 00:00:00:00:00:00:04

sw5 : 00:00:00:00:00:00:05

sw6 : 00:00:00:00:00:00:06

h1 : 00:00:00:00:00:00:01

h7 : 00:00:00:00:00:00:07

h8 : 00:00:00:00:00:00:08

Gerando tráfego UDP com iperf de h7 para h1, o seguinte caminho foi escolhido:

```

"Node: h1"
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)

[ 35] local 10.0.0.1 port 5001 connected with 10.0.0.7 port 35709
[ ID] Interval      Transfer    Bandwidth    Jitter    Lost/Total Datagrams
[ 35] 0.0- 1.0 sec  1.23 MBytes 10.3 Mbits/sec 0.014 ms  1/ 878 (0.11%)
[ 35] 1.0- 2.0 sec  26 datagrams received out-of-order
[ 35] 2.0- 3.0 sec  1.19 MBytes 10.0 Mbits/sec 0.013 ms  0/ 851 (0%)
[ 35] 3.0- 4.0 sec  1.19 MBytes 10.0 Mbits/sec 0.011 ms  0/ 850 (0%)
[ 35] 4.0- 5.0 sec  1.19 MBytes 10.0 Mbits/sec 0.020 ms  0/ 850 (0%)
[ 35] 5.0- 6.0 sec  1.19 MBytes 10.0 Mbits/sec 0.016 ms  0/ 851 (0%)
[ 35] 6.0- 7.0 sec  1.19 MBytes 10.0 Mbits/sec 0.009 ms  0/ 850 (0%)
[ 35] 7.0- 8.0 sec  1.19 MBytes 10.0 Mbits/sec 0.164 ms  0/ 850 (0%)
[ 35] 8.0- 9.0 sec  1.19 MBytes 10.0 Mbits/sec 0.005 ms  0/ 851 (0%)
[ 35] 9.0-10.0 sec  1.19 MBytes 10.0 Mbits/sec 0.012 ms  0/ 850 (0%)
[ 35] 10.0-11.0 sec 1.19 MBytes 10.0 Mbits/sec 0.051 ms  0/ 850 (0%)
[ 35] 11.0-12.0 sec 1.19 MBytes 10.0 Mbits/sec 0.012 ms  0/ 851 (0%)
[ 35] 12.0-13.0 sec 1.19 MBytes 10.0 Mbits/sec 0.011 ms  0/ 850 (0%)
[ 35] 13.0-14.0 sec 1.19 MBytes 10.0 Mbits/sec 0.015 ms  0/ 850 (0%)
[ 35] 14.0-15.0 sec 1.19 MBytes 10.0 Mbits/sec 0.004 ms  0/ 851 (0%)
[ 35] 15.0-16.0 sec 1.19 MBytes 10.0 Mbits/sec 0.047 ms  0/ 850 (0%)

"Node: h7"
root@mininet-vml:~# ./iperfclient10M.sh 10.0.0.1 5001
Client connecting to 10.0.0.1, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 208 KByte (default)

[ 35] local 10.0.0.7 port 35709 connected with 10.0.0.1 port 5001

```

```

"Node: h1"
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)

[ 35] local 10.0.0.1 port 5002 connected with 10.0.0.8 port 53149
[ ID] Interval      Transfer    Bandwidth    Jitter    Lost/Total Datagrams
[ 35] 0.0- 1.0 sec  1.20 MBytes 10.1 Mbits/sec 0.007 ms  0/ 855 (0%)
[ 35] 1.0- 2.0 sec  1.19 MBytes 10.0 Mbits/sec 0.026 ms  0/ 851 (0%)
[ 35] 2.0- 3.0 sec  1.19 MBytes 10.0 Mbits/sec 0.039 ms  0/ 850 (0%)
[ 35] 3.0- 4.0 sec  1.19 MBytes 10.0 Mbits/sec 0.023 ms  0/ 850 (0%)
[ 35] 4.0- 5.0 sec  1.19 MBytes 10.0 Mbits/sec 0.003 ms  0/ 851 (0%)
[ 35] 5.0- 6.0 sec  1.19 MBytes 10.0 Mbits/sec 0.004 ms  0/ 851 (0%)
[ 35] 6.0- 7.0 sec  1.19 MBytes 10.0 Mbits/sec 0.007 ms  0/ 850 (0%)
[ 35] 7.0- 8.0 sec  1.19 MBytes 10.0 Mbits/sec 0.005 ms  0/ 850 (0%)
[ 35] 8.0- 9.0 sec  1.19 MBytes 10.0 Mbits/sec 0.004 ms  0/ 850 (0%)
[ 35] 9.0-10.0 sec  1.19 MBytes 10.0 Mbits/sec 0.005 ms  0/ 851 (0%)
[ 35] 10.0-11.0 sec 1.19 MBytes 10.0 Mbits/sec 0.017 ms  0/ 850 (0%)
[ 35] 11.0-12.0 sec 1.19 MBytes 10.0 Mbits/sec 0.012 ms  0/ 851 (0%)
[ 35] 12.0-13.0 sec 1.19 MBytes 10.0 Mbits/sec 0.009 ms  0/ 850 (0%)
[ 35] 13.0-14.0 sec 1.19 MBytes 10.0 Mbits/sec 0.004 ms  0/ 850 (0%)
[ 35] 14.0-15.0 sec 1.19 MBytes 10.0 Mbits/sec 0.004 ms  0/ 850 (0%)
[ 35] 15.0-16.0 sec 1.19 MBytes 10.0 Mbits/sec 0.005 ms  0/ 851 (0%)
[ 35] 16.0-17.0 sec 1.19 MBytes 10.0 Mbits/sec 0.005 ms  0/ 850 (0%)
[ 35] 17.0-18.0 sec 1.19 MBytes 10.0 Mbits/sec 0.014 ms  0/ 850 (0%)

"Node: h7"
root@mininet-vml:~# ./iperfclient10M.sh 10.0.0.1 5002
Client connecting to 10.0.0.1, UDP port 5002
Sending 1470 byte datagrams
UDP buffer size: 208 KByte (default)

[ 35] local 10.0.0.8 port 53149 connected with 10.0.0.1 port 5002

```

```

04:17:44.414 INFO [n.f.p.PolicyForward] =====
04:17:44.414 INFO [n.f.p.PolicyForward] Packet from 00:00:00:00:00:00:00:04(1) 00:00:00:00:00:00:00:01(1)
04:17:44.415 INFO [n.f.p.PolicyForward] Path Candidate 0 Utilization 40750 on 81285
04:17:44.416 INFO [n.f.p.PolicyForward] Path Candidate 1 Utilization 61050 on 81348
04:17:44.416 INFO [n.f.p.PolicyForward] Path Candidate 2 Utilization 61050 on 60978
04:17:44.416 INFO [n.f.p.PolicyForward] Path Candidate 3 Utilization 81348 on 40624
04:17:44.416 INFO [n.f.p.PolicyForward] Path Candidate 4 Utilization 60978 on 101565
04:17:44.416 INFO [n.f.p.PolicyForward] Path Candidate 5 Utilization 40624 on 81285
04:17:44.416 INFO [n.f.p.PolicyForward] Path Candidate 6 Utilization 101565 on 101599
04:17:44.416 INFO [n.f.p.PolicyForward] Path Candidate 7 Utilization 81285
04:17:44.416 INFO [n.f.p.PolicyForward] Path Candidate 8 Utilization 101599
04:17:44.416 INFO [n.f.p.PolicyForward] === Path 5 chosen ===
04:17:44.416 INFO [n.f.p.PolicyForward] Path: -> sw 4 -> sw 4 -> sw 2 -> sw 2 -> sw 1 -> sw 1
04:17:44.416 INFO [n.f.p.PolicyForward] =====
04:17:47.110 INFO [n.f.p.PolicyForward] Collecting bandwidth status
04:17:49.297 INFO [n.f.p.PolicyForward] Packet from 00:00:00:00:00:00:00:01(1) 00:00:00:00:00:00:00:04(1)
04:17:49.298 INFO [n.f.p.PolicyForward] Path Candidate 0 Utilization 40600 on 60840
04:17:49.298 INFO [n.f.p.PolicyForward] Path Candidate 1 Utilization 60840 on 81080
04:17:49.298 INFO [n.f.p.PolicyForward] Path Candidate 2 Utilization 60840 on 81080
04:17:49.298 INFO [n.f.p.PolicyForward] Path Candidate 3 Utilization 81080 on 220288
04:17:49.299 INFO [n.f.p.PolicyForward] Path Candidate 4 Utilization 220288 on 989353
04:17:49.299 INFO [n.f.p.PolicyForward] Path Candidate 5 Utilization 989353 on 196988
04:17:49.299 INFO [n.f.p.PolicyForward] Path Candidate 6 Utilization 196988 on 908843
04:17:49.299 INFO [n.f.p.PolicyForward] Path Candidate 7 Utilization 908843 on 952734
04:17:49.299 INFO [n.f.p.PolicyForward] Path Candidate 8 Utilization 952734
04:17:49.299 INFO [n.f.p.PolicyForward] === Path 0 chosen ===
04:17:49.300 INFO [n.f.p.PolicyForward] Path: -> sw 1 -> sw 1 -> sw 3 -> sw 3 -> sw 4 -> sw 4
04:17:49.300 INFO [n.f.p.PolicyForward] =====

```

Caminho de h7 para h1: sw4 -> sw2 -> sw1



Caminho de h1 para h7: sw1 -> sw3 -> sw4

O caminho de ida e volta são diferentes pois quando h1 começou a responder para h7, já havia tráfego no primeiro caminho, o que fez com que o algoritmo escolhe-se um caminho alternativo com menor peso.

Após instalar este flow foi gerado um novo tráfego UDP, desta vez de h8 para h1 a fim de forçar um novo caminho.

```
04:20:24.726 INFO [n.f.p.PolicyForward] =====
04:20:24.727 INFO [n.f.p.PolicyForward] Packet from 00:00:00:00:00:00:04(2) 00:00:00:00:00:00:01(1)
04:20:24.727 INFO [n.f.p.PolicyForward] Path Candidate 0 Utilization 77174822
04:20:24.727 INFO [n.f.p.PolicyForward] Path Candidate 1 Utilization 40450
04:20:24.727 INFO [n.f.p.PolicyForward] Path Candidate 2 Utilization 60630
04:20:24.727 INFO [n.f.p.PolicyForward] Path Candidate 3 Utilization 60630
04:20:24.728 INFO [n.f.p.PolicyForward] Path Candidate 4 Utilization 80808
04:20:24.728 INFO [n.f.p.PolicyForward] Path Candidate 5 Utilization 51480720
04:20:24.728 INFO [n.f.p.PolicyForward] Path Candidate 6 Utilization 12937948
04:20:24.728 INFO [n.f.p.PolicyForward] Path Candidate 7 Utilization 10386702
04:20:24.728 INFO [n.f.p.PolicyForward] Path Candidate 8 Utilization 51521080
04:20:24.728 INFO [n.f.p.PolicyForward] === Path 1 chosed ===
04:20:24.729 INFO [n.f.p.PolicyForward] Path: -> sw 4 -> sw 4 -> sw 3 -> sw 3 -> sw 1 -> sw 1
04:20:24.729 INFO [n.f.p.PolicyForward] =====
04:20:26.110 INFO [n.f.p.PolicyForward] Collecting bandwidth status
04:20:29.110 INFO [n.f.p.PolicyForward] Collecting bandwidth status
04:20:29.731 INFO [n.f.p.PolicyForward] Packet from 00:00:00:00:00:00:00:01(1) 00:00:00:00:00:00:04(2)
04:20:29.732 INFO [n.f.p.PolicyForward] Path Candidate 0 Utilization 452086
04:20:29.732 INFO [n.f.p.PolicyForward] Path Candidate 1 Utilization 61071
04:20:29.732 INFO [n.f.p.PolicyForward] Path Candidate 2 Utilization 51462085
04:20:29.733 INFO [n.f.p.PolicyForward] Path Candidate 3 Utilization 403851
04:20:29.733 INFO [n.f.p.PolicyForward] Path Candidate 4 Utilization 132800
04:20:29.733 INFO [n.f.p.PolicyForward] Path Candidate 5 Utilization 8626428
04:20:29.733 INFO [n.f.p.PolicyForward] Path Candidate 6 Utilization 45083503
04:20:29.733 INFO [n.f.p.PolicyForward] Path Candidate 7 Utilization 46430723
04:20:29.733 INFO [n.f.p.PolicyForward] Path Candidate 8 Utilization 5611116
04:20:29.734 INFO [n.f.p.PolicyForward] === Path 1 chosed ===
04:20:29.734 INFO [n.f.p.PolicyForward] Path: -> sw 1 -> sw 1 -> sw 5 -> sw 5 -> sw 6 -> sw 6 -> sw 4 -> sw 4
04:20:29.734 INFO [n.f.p.PolicyForward] =====
```

Desta vez foram escolhidos os seguintes caminhos.

Caminho de h8 para h1: sw4 -> sw3 -> sw1

Caminho de h1 para h8: sw1 -> sw5 -> sw6 -> sw4

No caso escolhido o caminho poderia ter sido o mesmo para o dois tráfegos, visto que ambos eram tráfegos entre o switch 1 e 4, entretanto 4 caminhos diferentes foram usados, isto se deu pois a medida que novos tráfegos vão entrando na rede, o algoritmo de escolha busca distribuir da melhor forma buscando eficiência e equilíbrio no uso dos recursos.



## **Conclusão**

Apesar do controlador escolhido ter uma boa documentação, tivemos alguns problemas durante o desenvolvimento que custaram bastante tempo. O principal problema foi após escolher o melhor caminho, pois o método que busca os caminhos possíveis retorna o caminho entre os switches de origem e destino, desta forma tivemos que adicionar as portas onde os hosts de origem e destino estavam conectados para completar o caminho.

Pretendíamos também levar em conta a capacidade de transmissão do enlace, para isto era necessário utilizar o OpenFlow 1.2+ e obter estas informações através de uma consulta a porta do switch. Entretanto os switches ovsk do mininet configuram as portas para 10Gb, e o limite de banda é implementado com tc, que controla o tráfego artificialmente. Sendo assim, para o controlador a velocidade das portas é igual para todas sempre, pois a porta não assume a velocidade máxima do enlace como deveria ser no mundo real.

Algumas vezes tivemos problemas com pacotes de broadcast inundando a rede também.

Para identificar e tratar estes problemas fizemos uso do wireshark, debug do eclipse e impressão de informações de log.

Por fim, aprendemos bastante sobre a arquitetura do controlador, como implementar um módulo no floodlight e como programar redes definidas por software. Neste sentido o trabalho foi muito enriquecedor e ajudou a consolidar os exemplos teóricos vistos em aula.