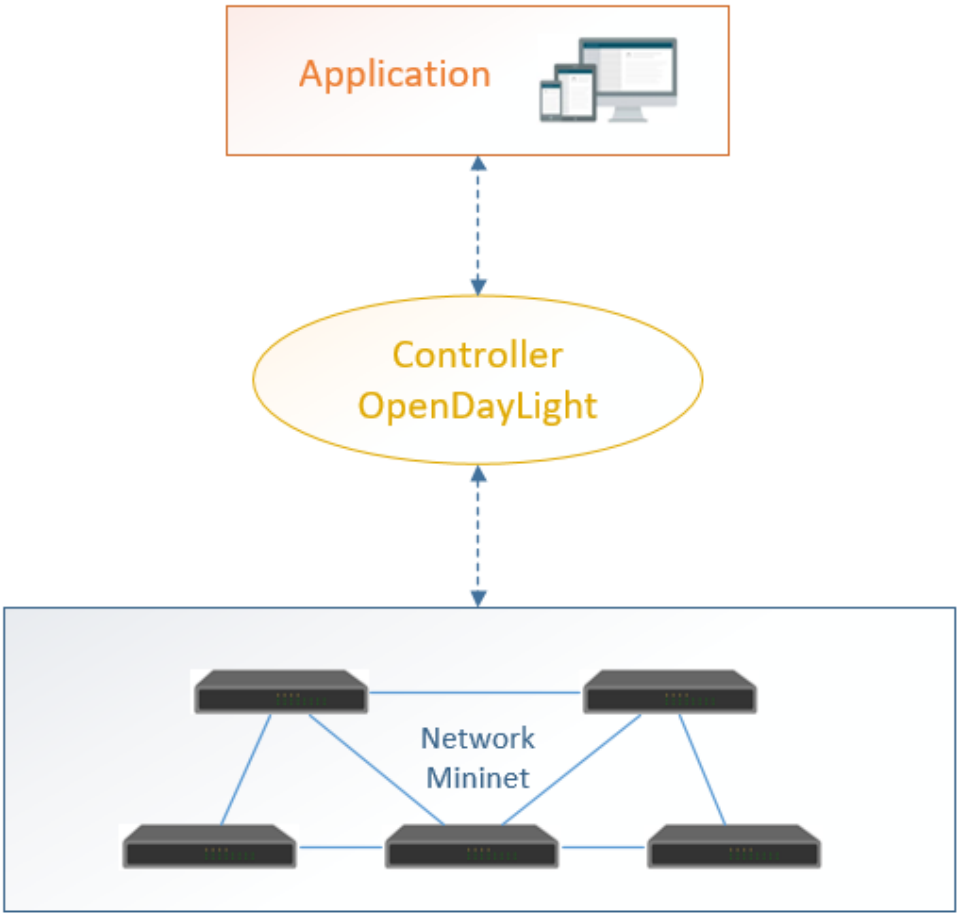
Engenharia Informática

Laboratório de Tecnologias de Informação

Cloud – Analytics – Service Assurance



Estudantes:
Marin Grabovschi nº 2151476
Tiago Cardoso nº 2141291
Docente:
Daniel Fuentes

Índice

Resumo

O presente relatório descreve todo o trabalho realizado no âmbito da unidade curricular de Laboratório de Tecnologias de Informação, onde desenvolvemos uma aplicação que é capaz de comunicar com um controlador **SDN** que por sua vez consegue gerir um conjunto de equipamentos numa rede virtual.

1. Introdução

Seguindo uma topologia ***Cloud*** – ***Analytics*** – ***Service Assurance*** o nosso trabalho divide-se nas seguintes camadas.

Camada aplicacional desenvolvida em C#;

Controlador **SDN** OpenDaylight;

Rede virtual Mininet.

2. Camada de Aplicação

Com a necessidade de existir uma interface capaz de interagir com o controlador **SDN** (OpenDaylight), desenvolvemos uma aplicação em C# capaz de fazer pedidos à **API** incorporada no OpenDaylight.
Partindo deste pressuposto a nossa aplicação é capaz:
- Obter informação através de pedidos **GET**;
- Enviar informação através de pedidos **POST**;
- Atualizar informação através de pedidos **PUT**;
- Apagar informação através de pedidos **DELETE**.

De seguida iremos descrever todo o processo de desenvolvimento nesta camada.
Numa primeira fase começamos por estudar o processo de comunicação através de **HttpWebRequest** em C#.
Uma vez garantido a ligação da nossa aplicação ao controlador **SDN**, tivemos de analisar/investigar toda a informação e documentação do OpenDaylight referente aos pedidos **GET**, **POST** e **PUT**.
As principais dificuldades neste ponto foi a escassa informação por parte do OpenDaylight relativa aos pedidos à **API** e no parse do **JSON** recebido através dos pedidos **GET**, ainda assim, conseguimos com mais ou menos esforço atingir os nossos objetivos.

A nossa aplicação consegue com sucesso, ligar-se ao controlador **SDN**, executar os pedidos **GET**, fazer parse do **JSON** recebido e mostrar numa caixa de texto informações como:
- NodeID;
- Node Termination Points;

Relativamente aos pedidos **POST**, a nossa aplicação dispõe de um formulário com campos obrigatórios que devem ser preenchidos pelo utilizador que quando submetidos através de um click no botão **Create** enviam a informação em **XML** para o controlador **SDN**. Este **POST** é responsável por criar um **Flow**.
Os pedidos **PUT** atualizam a informação dos **Flow** e os pedidos **DELETE** eliminam os **Flow** existentes.

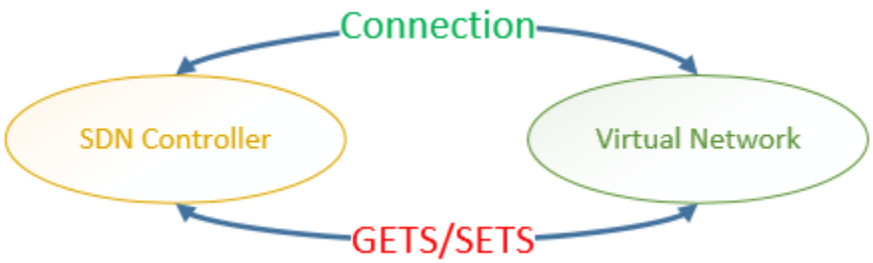


Figura 1 - Interação App c/ SDN Controller

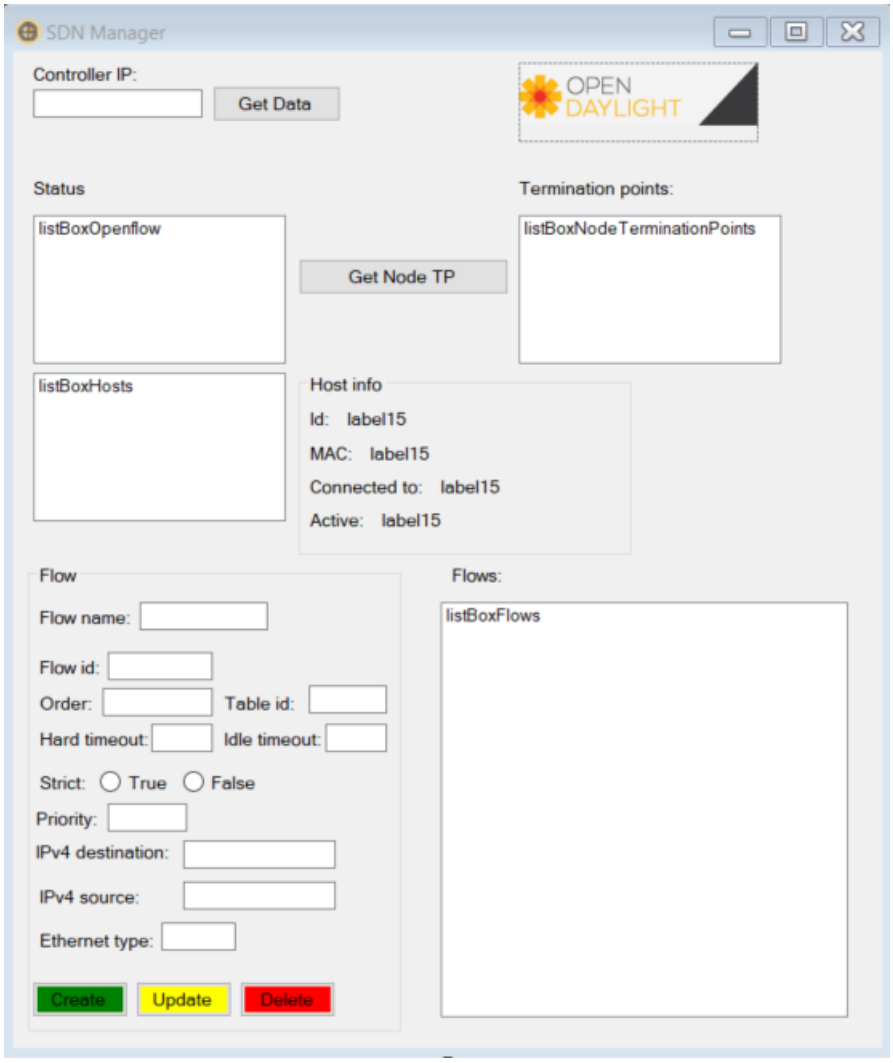


Figura 2 - SDN Manager

A aplicação que interage com o controlador **SDN** foi desenvolvida na linguagem **C#.** Dada a possibilidade de escolher a linguagem de desenvolvimento, foi optada a linguagem **C#** pela facilidade de fazer **PARSE** da informação recebida pelos pedidos **GET**. Toda a informação recebida através dos pedidos **GET** estão em formato **JSON**. O conteúdo do **Body** quando se fazem pedidos **POST** e **PUT** está em formato **XML**. Outra vantagem da aplicação desenvolvida é o facto de conseguir comunicar com qualquer controlador, mesmo que esteja instalado numa rede privada ou cloud pública, fazendo com que seja o mais desacopulada possível e fácil de integrar num sistema já existente.

A figura acima representa a aplicação desenvolvida, **SDN Manager**. Ao introduzir o **IP** do controlador e carregando no botão **Get Data**, são preenchidas as listas **listBoxOpenflow** (para os equipamentos **OVS**) e **listBoxHosts** (para os hosts). Ao selecionar um equipamento, é possível clicar no botão **Get Node TP** onde é preenhida a lista **listBoxNodeTerminationPoints** com as ligações existentes a outros equipamentos.

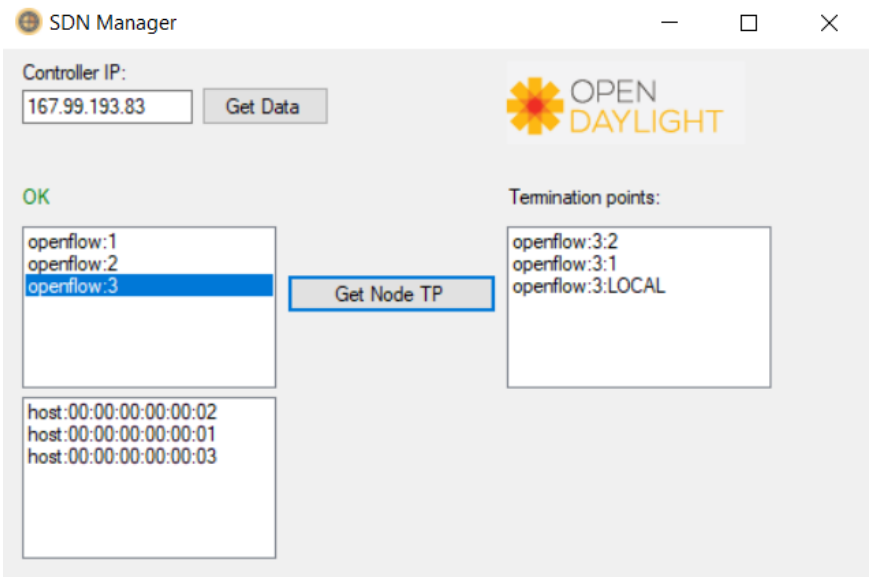


Figura 3 - Node Termination Points

Ao selecionar um host para executar o mesmo passo, a **listBoxNodeTerminationPoints** é preenhida com o **IP** da máquina e surge também umas informações adicionais acerca do equipamento, tais como **MAC**, o **Id** e a que **OVS** está ligado.

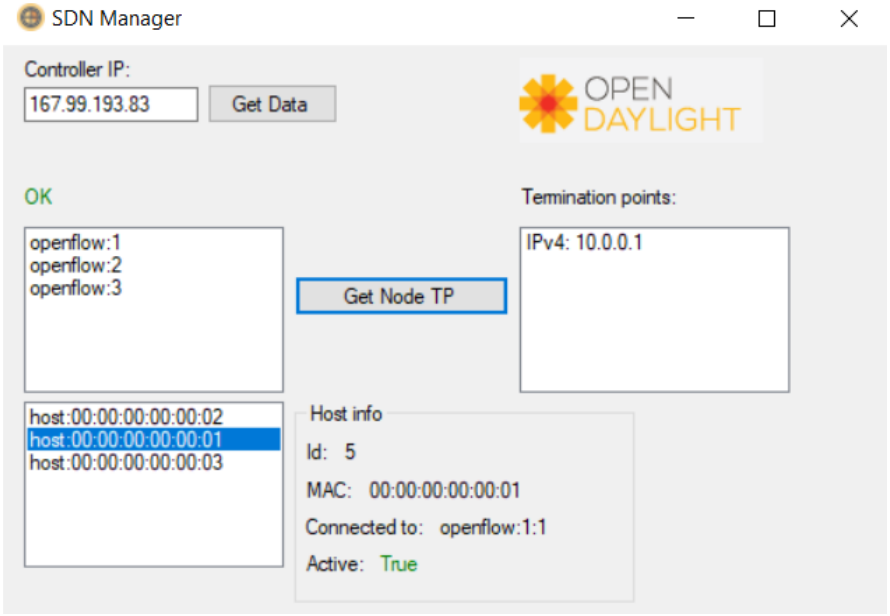


Figura 4 - Host Info

Na segunda metade do **SDN Manager**, é dada a possibilidade de criar flows para os equipamentos que funcionam através do protocolo **Openflow**. Para isso basta selecionar um **OVS** e preencher o formulário de criação do flow, clicando no botão **Create** e o pedido POST é enviado.

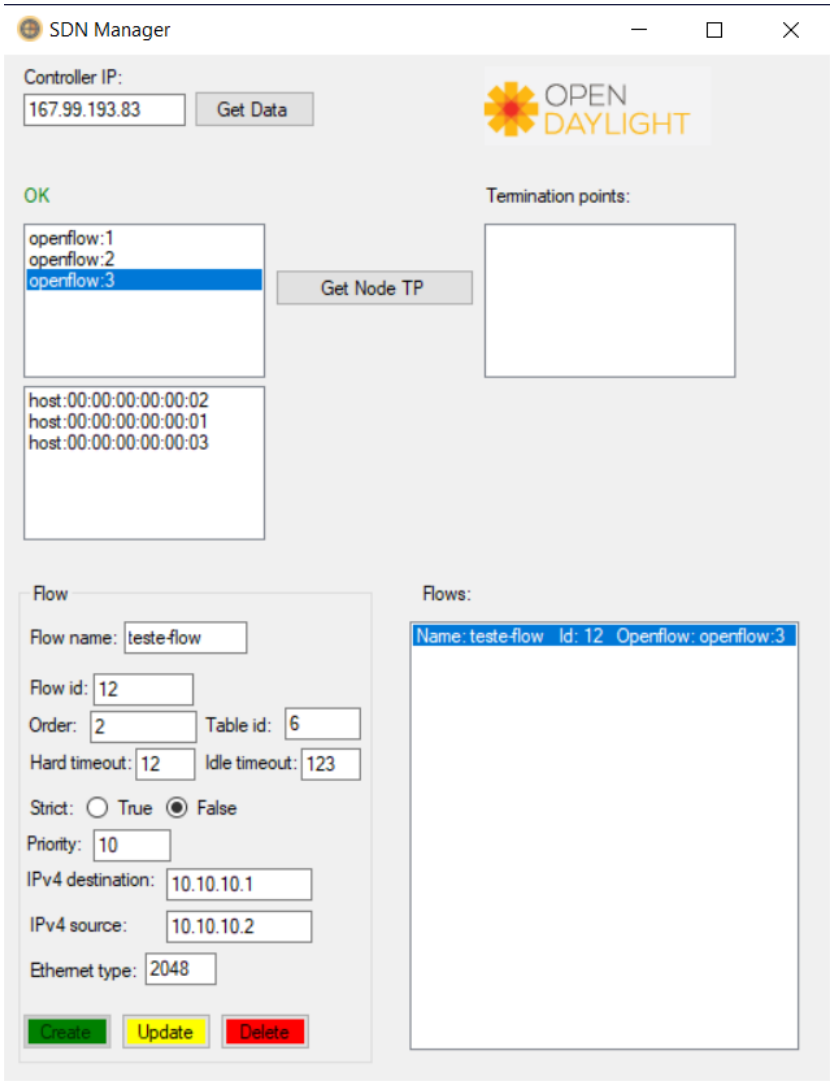


Figura 5 - Criação de flows

Depois de criado o flow, será guardada uma indicação de que foi criado um flow numa lista de flows ao lado do formulário. Ao selecionar um flow dessa lista, é preenchido o formulário com os dados do flow fazendo um pedido **GET** ao controlador. De seguida o utilizador pode alterar os campos necessários e clicar no botão **Update**, onde será feito um pedido **PUT**, enviando a informação em formato **XML**, ou então apagar o flow, clicando o botão **Delete**.

Na figura em baixo foi usada a ferramenta **Postman** para verificar o flow que foi criado acima, verificando-se que o flow foi criado na aplicação, e feito o **POST** e enviado a informação **XML** no **Body** do pedido.

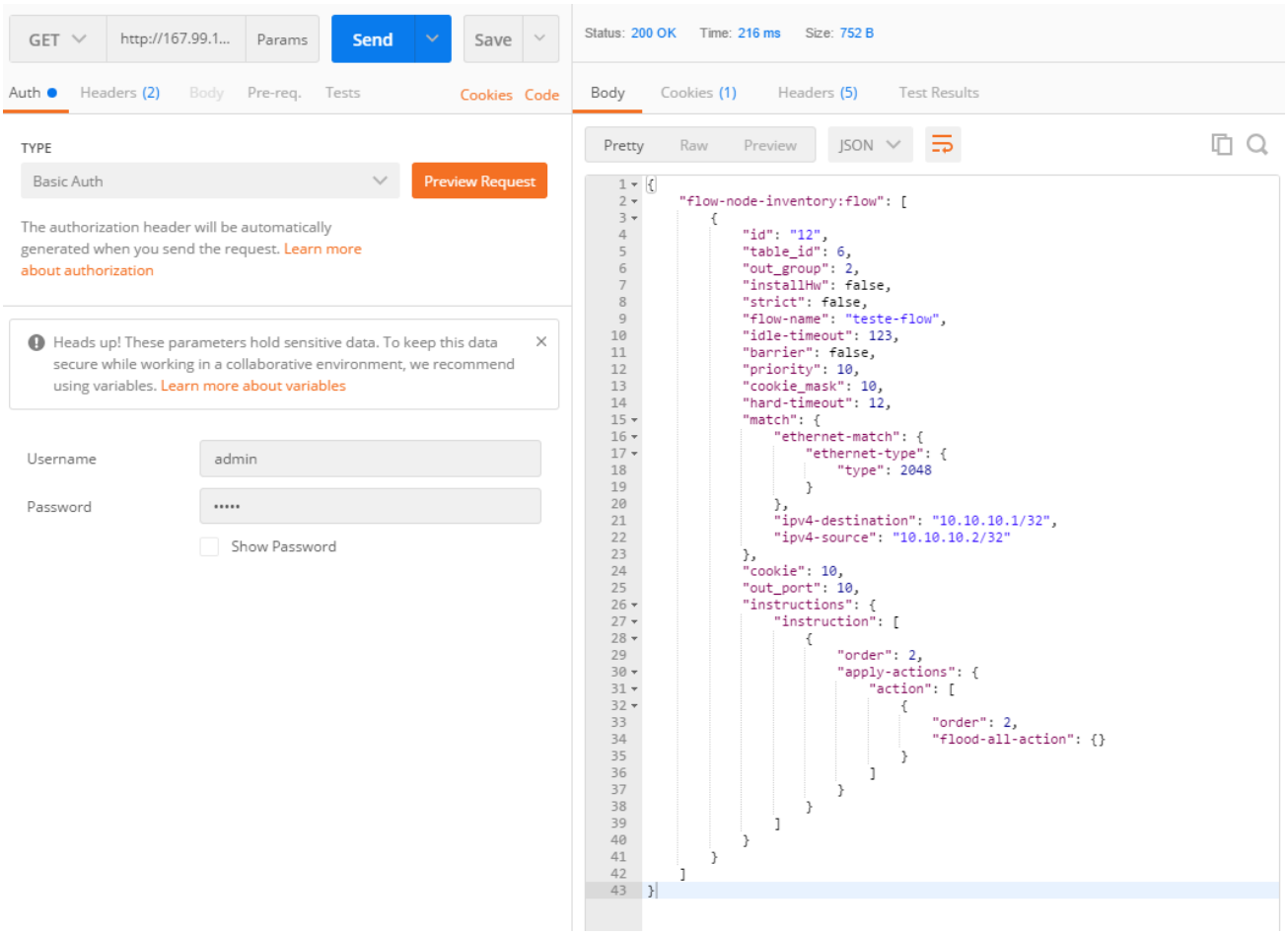


Figura 6 - GET do flow c/ Postman

3. Controlador SDN

O controlador **SDN** que estamos a utilizar no projeto é o OpenDaylight e foi instalado numa máquina ubuntu server 17.10.

No processo de instalação as maiores dificuldades estiveram relacionadas com a instalação do Java e com a necessidade de descobrir quais os pacotes relacionados com o acesso à **API** (odl-restconf-all).

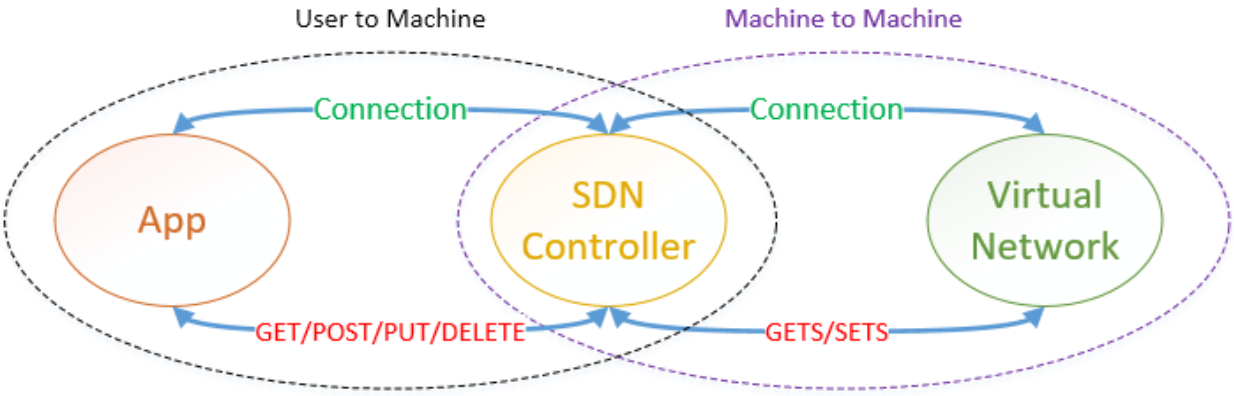


Figura 7 - Interação SDN & Virtual Network

Pontos de instalação:

Obter pacote OpenDaylight para o Ubuntu Server:

```
Wget https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integrati
on/karaf/0.8.0/karaf-0.8.0.tar.gz
```

Descompactação:

```
tar -xvf karaf-0.8.0.tar.gz
```

Iniciar OpenDaylight:

```
cd karaf-0.8.0
./bin/karaf
```

Instalar features no OpenDaylight

```
feature:install odl-restconf odl-l2switch-switch odl-mdsal-apidocs odl-dlux-core odl-dluxa
ppsapplications odl-dluxapps-topology odl-dluxapps-yangutils
```

O OpenDaylight possui uma interface web onde podemos consultar a topologia de rede criada no Mininet.

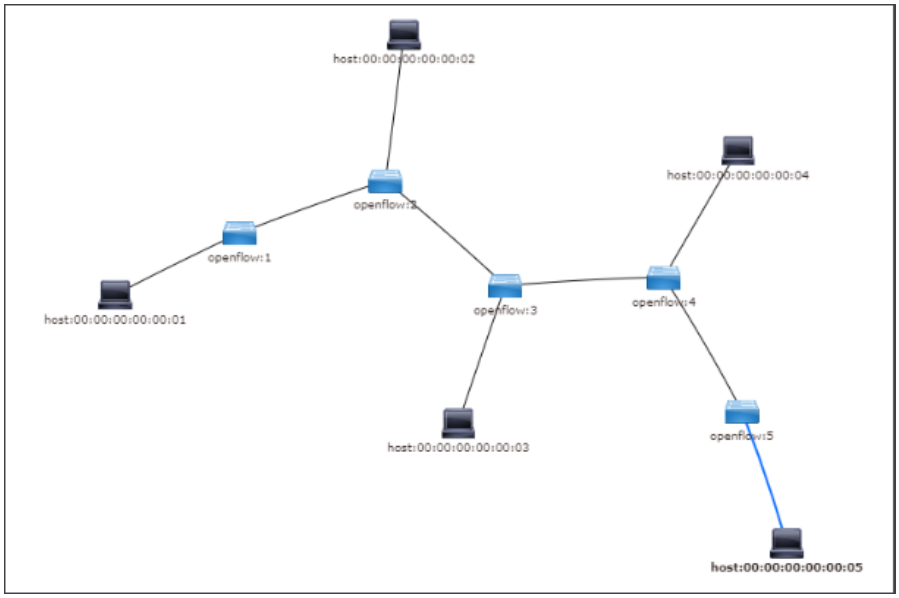


Figura 8 - OpenDaylight Network Topology

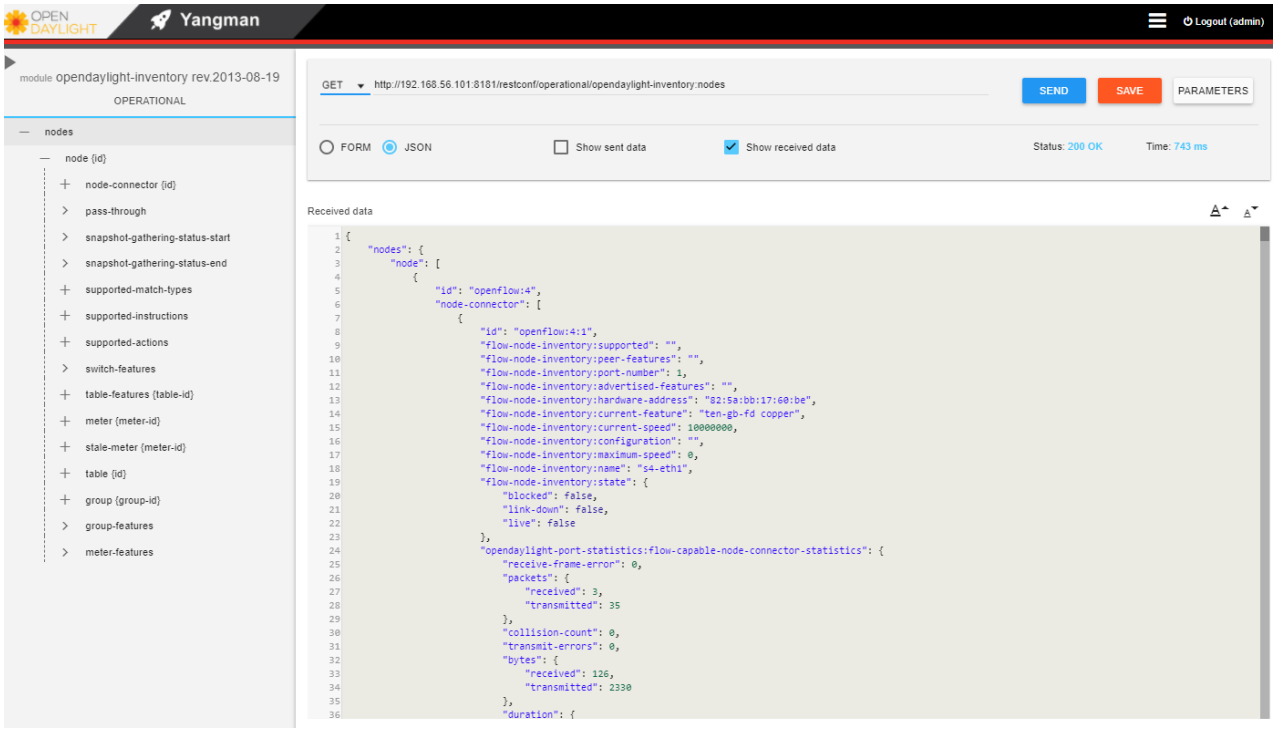


Figura 9 - OpenDaylight Yangman

4. Mininet

Para criar redes virtuais que possam ser controladas por um controlador **SDN**, foi usado o **Mininet**. Não teve qualquer processo de instalação, visto que na página oficial do **Mininet** (http://mininet.org/download/) está disponível uma máquina virtual Ubuntu 14.04 LTS com o Mininet pré-instalado, de modo a ser uma solução plug & play.

Depois de iniciar a máquina e autenticar, está pronta a receber comandos para criar redes virtuais.
Abaixo segue-se o comando que foi usado para gerar uma rede virtual:

```
sudo mn --topo linear,3 --mac --controller=remote,ip=167.99.193.83,port=6633 --switch ovs,
protocols=OpenFlow13
```

Através deste comando conseguimos indicar que tipo de topologia e quantas máquinas desejamos, através da opção `--topo`. Com a opção `--controller` indicamos o controlador remoto, nomeadamente o seu IP e porto e por fim indicamos que protocolo irão usar os switches **OVS** (**OpenFlow13**).

```
ubuntu@ip-172-31-22-247:~$ sudo mn --topo linear,3 --mac --controller=remote,ip=167.99.193.83,port=6633 --switch ovs,protocols=OpenFlow13
*** Creating network
*** Adding controller
Unable to contact the remote controller at 167.99.193.83:6633
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

Figura 10 - Criação da rede Mininet

Após a criação da rede Mininet, somos redirecionados para uma nova prompt (**mininet >**) onde é possível executar comandos sobre a rede criada. O comando `pingall` foi usado para testar as ligações dos equipamentos da rede.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

Figura 11 - Teste às ligações

5. Conclusão

Com o desenvolvimento desta solução deu-nos possibilidade de explorar capacidades de criar e gerir redes através de um único ponto, o controlador **SDN**, resultando numa completa virtualização da rede e do software que a gere.

Ao longo do desenvolvimento sentimos mais dificuldades a perceber o funcionamento e utilização da **API** que o controlador disponibiliza para criar a nossa aplicação devido à sua falta de documentação.

O que nos motivou mais foi a luta pelo conhecimento e pela informação que apesar de escassa, foi uma ajuda para a elaboração do trabalho, luta que teremos que demonstrar de igual maneira no mercado de trabalho.