12 de julho de 2021

Disciplina de Redes de Computadores
Profª Cristina Dominicini
Prof. Gilmar Vassoler

Trabalho 2 – Topologias de Datacenter com Mininet

1 Orientações gerais

O trabalho deverá ser realizado em grupos de dois alunos, ou individualmente caso deseje.

2 Objetivos

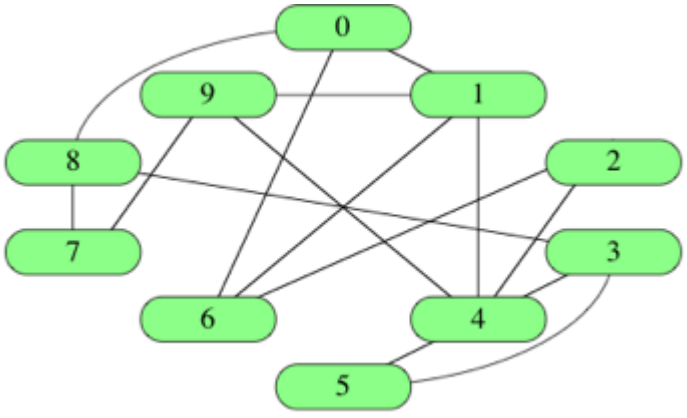
- Familiarizar-se a emulação de cenários de redes utilizando a API do Mininet.
- Familiarizar-se com a biblioteca NetworkX análise de topologias de rede em Python, versão 3.6, ou superior.

3 Conhecimentos necessários para desenvolvimento do trabalho

3.1 Representando grafos

Um modo comum de representar um gráfico é simplesmente como uma lista de nós adjacentes, que chamamos de lista de arestas. Por exemplo, a lista [[0,1], [0,6], [0,8], [1,4], [1,6], [1,9], [2,4], [2,6], [3,4], [3,5], [3,8], [4,5], [4,9], [7,8], [7,9]] representa do grafo da figura abaixo.

Mais informações podem ser obtidas em https://pt.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/representing-graphs



3.2 NetworkX

O NetworkX é uma biblioteca para criação, manipulação e estudos sobre redes que possui uma extensa lista de funções. Veja mais detalhes em: https://networkx.github.io/

3.3 Mininet

Conforme apresentado na disciplina, o Mininet é um emulador de rede que cria uma rede de hosts virtuais, comutadores, controladores e enlaces.

Veja mais detalhes em: http://mininet.org

4 Descrição do trabalho

Neste trabalho você será desafiado a emular uma de rede de computadores no Mininet, montada a partir de um arquivo com a lista de arestas da topologia para executar o teste detalhado abaixo:

4.1 Teste de Ping all da topologia

Executar o script de teste que aceita os seguintes parâmetros e executa as ações abaixo:

```
# Exemplo de chamada do script

# Executa o teste para topologia Fat-Tree com switches de 4 portas ou k=4
sudo python teste_topologia.py --topology fattree --k 4

# Executa o teste para uma topologia genérica a partir de uma lista de arestas
sudo python teste_topologia.py --topology generic --file topo1.txt
```

Fazer o *parsing* dos parâmetros de entrada.
Levantar a topologia de rede no Mininet com os parâmetros de configuração adequados.
Enviar pacotes de *ping* de todas as máquinas para todas as máquinas com 100% de sucesso (*0% dropped 100% received*).

Observação: Neste trabalho, ainda não será necessário desenvolver uma aplicação SDN. O objetivo é que você seja capaz de executar um teste de *ping all*. Você pode escolher usar um controlador padrão, mas preste atenção no que acontece quando a topologia possui loops:
https://github.com/mininet/mininet/wiki/FAQ#ethernet-loops
A solução para estes casos pode ser configurar os switches com o protocolo **Spanning Tree Protocol** (**STP**), como vocês podem fazer neste trabalho, ou desenvolver uma aplicação SDN com um controlador que consiga explorar múltiplos caminhos, como iremos fazer no próximo trabalho.

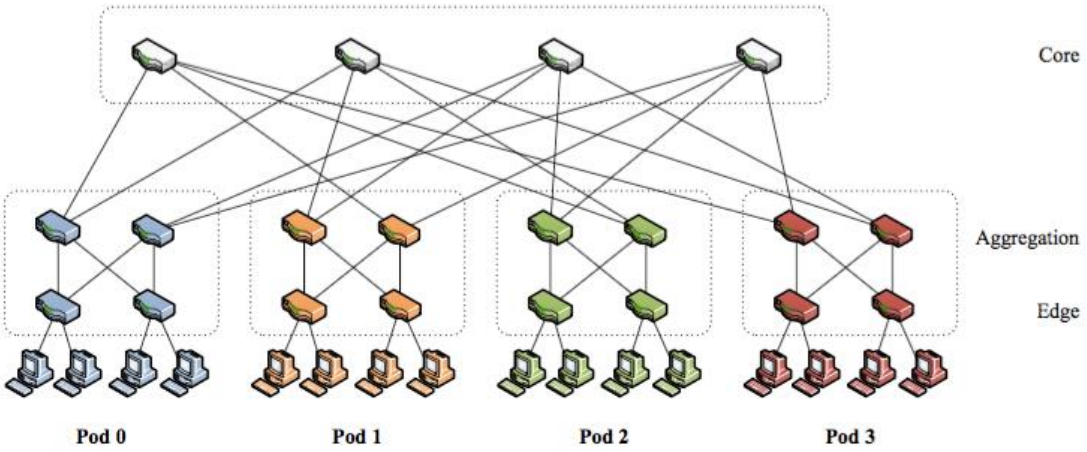
OBS: Incluir no relatório uma explicação sobre a solução utilizada para resolver o problema de loops e as desvantagens de usar o protocolo STP.

4.2 Topologias de teste

Seu script deve ser capaz de criar as seguintes **03 topologias** no Mininet:

4.2.1 Topologia Fat-Tree [3] com switches de 4 portas ou k=4 (20 switches, 16 hosts).

Para criar as topologias de data center você deve usar as regras de formação da topologia conforme descrito nos artigos. Os repositórios abaixo podem te auxiliar nesta tarefa:
https://github.com/brandonheller/ripl/blob/master/ripl/dctopo.py
https://fnss.readthedocs.io/en/latest/_modules/fnss/topologies/datacenter.html



4.2.2 Topologias aleatórias

Para gerar topologias aleatórias você deve usar o NetworkX para gerar 2 arquivos de entrada com a lista de arestas de topologias aleatórias com 16 switches e a mesma quantidade de enlaces que a Fat-Tree. A lista de arestas representa apenas os enlaces entre os switches e considera que cada switch está conectado a um host.
O exemplo abaixo pode te auxiliar a criar um gráfico aleatório G {n, m} com ***n*** nós e ***m*** arestas:
https://networkx.github.io/documentation/stable/auto_examples/graph/plot_erdos_renyi.html

OBS: Incluir no relatório representação gráfica de todos os grafos das topologias usadas.

5 Entrega do trabalho e pontuação

- Entrega: Relatório explicando em detalhes a implementação e código fonte comentado.
- Prazo: 27/07/2021 às 23:55
- Valor: 10 pontos.

6 Referências

[1] G. L. Vassoler, M. H. M. Paiva, M. R. N. Ribeiro and M. E. V. Segatto, "Twin Datacenter Interconnection Topology," in IEEE Micro, vol. 34, no. 5, pp. 8-17, Sept.-Oct. 2014, doi: 10.1109/MM.2014.63.

[2] N. Dukkipati and N. McKeown, “Why FlowCompletion Time is the Right Metric,” SIGCOMM Computer Comm. Rev., vol. 36, no. 1, 2006, pp. 59-62.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” ACM SIGCOMM Computer Communication Review, 2008.