AUTOMAÇÃO DE REDES UTILIZANDO PYTHON

Autor (Uênio Vicente Rocha); Orientador (Marcelo Portela Sousa);
Instituto Federal da Paraíba - Campus Campina Grande
uenio.v.rocha@ieee.org; marcelo.portela@ieee.org

INTRODUÇÃO

Com o passar dos anos, as redes de computadores começaram a consumir cada vez mais recursos e serviços. Este crescimento trouxe a necessidade de controle mais eficaz dos dados trafegados. Os diversos dispositivos de redes existentes, como Switches e Roteadores, contêm novos aplicativos e estes fazem com que as redes apresentem demandas mais complexas para gerenciar e controlar os dados. A formação em redes de comunicação de dados é muitas vezes projetada para administradores de redes e possui foco nas diversas especificações de protocolo de rede, arquitetura, projetos e ferramentas para administrar as redes [1].

Programas que reproduzem características e funcionalidades de dispositivos de rede tornam-se amplamente disponíveis. Um software emulador é uma ferramenta que reproduz uma plataforma virtualizada que permite que uma dada arquitetura de computador consiga executar sistemas que foram desenvolvidos para outra arquitetura específica [2].

No intuito de realizar a implantação ou modificação da rede, o administrador muitas vezes precisa acessar o equipamento físico, em que precisa dispor de um acesso console, ou acessá-lo pela rede, quando este, possui Telnet ou SSH previamente configurados, e assim realizar as alterações necessárias em todos os equipamentos [3]. Porém, isso torna um trabalho árduo e cansativo, aumentando a possibilidade de cometer erros, quando se tem uma rede escalável. Avaliando esse problema, é possível alterar esse cenário, utilizando scripts em Python, a fim de automatizar o processo de configuração dos equipamentos de redes [4].

Python é uma linguagem de programação de codificação flexível e simples, definida nos seus documentos da seguinte forma: “Python é uma linguagem de programação de grande propósito, dinâmica, orientada a objetos e de uso geral que usa Intérprete e pode ser usado em um vasto domínio de aplicativos” [5][6].

O Python é ideal para a prototipagem, implantação e aplicações de rede. Isso torna o processo de configuração mais rápido e eficaz e menos susceptível a erros [7].

O trabalho está definido da seguinte maneira: na etapa Justificativa, é apresentada a pesquisa na área de automação e seus objetivos; em Metodologia, o modelo de configuração realizada no emulador GNS3 e no cenário real, e a organização do código em Python; e por fim, os resultados obtidos e as conclusões do trabalho realizadas.

JUSTIFICATIVA E OBJETIVOS

Como foi apresentado na seção anterior, um campo de pesquisa que tem atraído grande interesse na área de Redes de Computadores tem sido o de automação de redes. Diferentemente do modelo de gerência das redes tradicional, no qual o administrador realiza a configuração manual e individualizada dos equipamentos, fazendo uso de um conjunto de comandos de baixo nível e específicos para cada equipamento, nas redes programáveis, é possível elevar o nível de abstração do processo de elaboração de políticas de gerência e favorecer a instalação destas regras em dispositivos distintos de rede [8].

Em linhas gerais, este recente paradigma propõe uma abordagem inovadora, a fim de controlar e reconfigurar o comportamento global de uma rede de computadores: retirar a lógica de controle da infraestrutura da rede e centralizá-la em um sistema operacional, deixando os equipamentos de rede como meros comutadores de pacotes, os quais serão orquestrados por uma interface não proprietária. O “cérebro” de uma rede utilizando automação em Python estará, portanto, completamente desacoplado dos vários dispositivos de baixo nível da rede o que representa uma ruptura expressiva da tradicional verticalização (o acoplamento entre os planos de controle e dados) observada nas arquiteturas de redes convencionais [9].

METODOLOGIA

O GNS3 é um aplicativo gratuito sob licença da GNU que prevê uma interface gráfica e permite ao usuário construir a topologia de rede que pretende configurar. Ele reúne diversos emuladores de sistemas operacionais.

As etapas aplicadas nesse trabalho seguem essa sequência: automação com Script em Python; emulação de equipamentos dos fabricantes Cisco, *Hewlett-Packard*, Mikrotik e Juniper no GNS3; e um cenário real com equipamentos do fabricante Cisco.

EMULAÇÃO NO GNS3

A Figura 1 exibe a topologia criada no GNS3, e está definida da seguinte maneira: foi utilizada uma máquina com o sistema operacional Ubuntu 16.04 LTS para fazer o gerenciamento da rede e sua interface está configurada como Acesso; os Switches receberam endereçamento IPv4 192.168.0.1/24 a 192.168.0.4/24 na interface virtual Vlan 1; os Roteadores receberam endereçamento IPv4 192.168.0.21/24 a 192.168.0.24/24 na interface GigabitEthernet 1/0; todos os equipamentos foram configurados com acesso Telnet; na execução do Script foi utilizado o Python 2.7; as interfaces GigabitEthernet 2/0 foram configurados como interface Acesso e as portas dos Switches conectadas aos roteadores nas interfaces GigabitEthernet 1/0 foram configurados como Tronco; no GNS3 foi utilizado um cenário mais simples como teste para automação em Python.

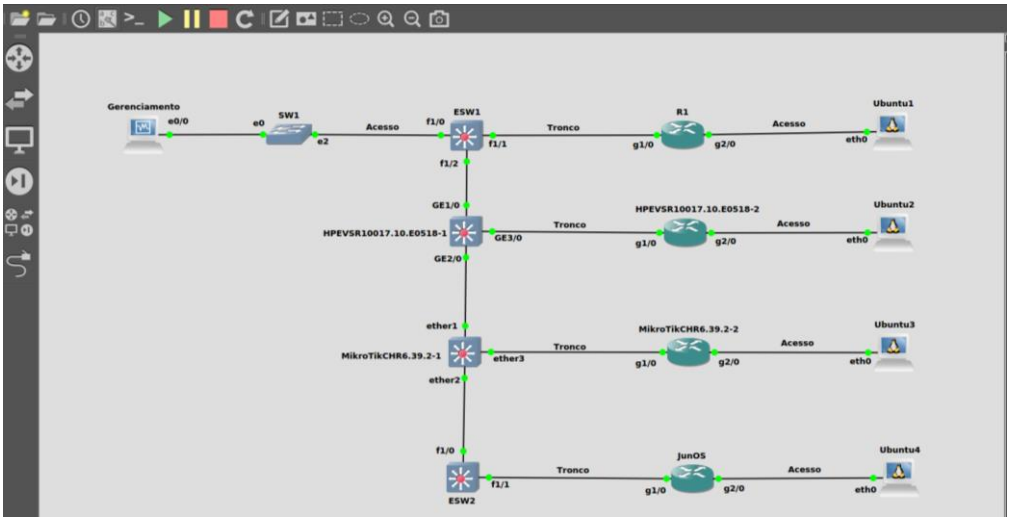

Figura 1 - Cenário Emulado Com Diferentes Fabricantes.

APLICAÇÃO EM UM CENÁRIO REAL

A Figura 2 apresenta o cenário real. O rack da esquerda possui 12 Switches do fabricante Cisco, modelo 2960, o rack da direta possui 12 Roteadores do fabricante Cisco, modelo 1941 e 12 PCs. Os Switches receberam endereçamento IPv4 192.168.0.1/24 a 192.168.0.12/24 na interface virtual Vlan 1; os Roteadores receberam endereçamento IPv4 192.168.0.21/24 a 192.168.0.32/24 na interface GigabitEthernet 1/0; o acesso Telnet foi configurado em todos os equipamentos.

Figura 2 - Cenário Real Com Equipamentos Cisco.

CÓDIGO EM PYTHON

Na Figura 3 o código em Python é descrito da seguinte maneira: na linha 1, a biblioteca de automação *pexpect* é chamada; na linha 7, as funções de configurações dos equipamento é definida; na linha 27, é criado um dicionário que relaciona cada função; na linha 31, a biblioteca *pexpect* é invocada, para realizar a conexão Telnet; e na linha 32, o comando *get* relaciona a função que implementa o dicionário.

ESQUEMA DE CONFIGURAÇÕES DOS SWITCHES E ROTEADORES

No Script as configurações dos dispositivos de redes segue essa sequência: o *hostname* foi configurado nos Switches; a criação de VLANs; atribuição de nome as VLANs; a Vlan 10 foi atribuída as interfaces FastEthernet 1/5 a 1/9; o *hostname* foi configurado nos Roteadores*;* as interfaces GigabitEthernet 2/0 receberam enderençamento IPv4 10.0.21.254/24 a 10.0.32.254/24; apenas uma área foi utilizada na configuração do OSPF do inglês (Open Shortest Path First) na (Área 0); e a interface GigabitEthernet 2/0 foi configurada como passiva.

```python
import pexpect
vl_dados = raw_input('Entre com a VLAN de Dados (ex: 10): ')
no_sw_rot = input('Entre com o numero de switches e roteadores a serem configurados: ')
user = raw_input('Entre com o usuario Telnet: ')
senha = raw_input('Entre com a senha para Telnet: ')
#Define as Funcoes
def sw1 (): # CISCO
    if user and senha:
        tn.expect('Username: ')
        tn.sendline(user)
        tn.expect('Password:')
        tn.sendline(senha)

        tn.sendline('conf t\n')
        tn.sendline('hostname S' + str(c1) + '\n')

        for vl in range (2,21):
            tn.sendline('vlan ' + str(vl) + '\n')
            tn.sendline('name VLAN' + str(vl) + '\n')

        for iface in range (5,10):
            tn.sendline('int fa1/' + str(iface) + '\n')
            tn.sendline('switchport mode access\n')
            tn.sendline('switchport access vlan ' + str(vl_dados) + '\n')

#Cria o Dicionario que Relaciona as Funcoes
switch = { 1: sw1, 2: sw2, 3: sw3, 4: sw4, 21: R1, 22: R2, 23: R3, 24: R4,}

for c1 in range (1,no_sw_rot + 1):
    #Configura os Switchs
    tn = pexpect.spawn('telnet 192.168.0.' + str(c1))
    switch.get(c1)()
    #Configura os Roteadores
    c2 = c1 + 20
    tn = pexpect.spawn('telnet 192.168.0.' + str(c2))
    switch.get(c2)() #Retorna o Dicionario
```

Figura 3 - Código em Python

RESULTADOS E DISCUSSÃO

Na Tabela 1 apresenta o método de configuração manual, copiar e colar do inglês (*copy and paste)* que utilizou 769 linhas de código, para configurar 12 Switches Cisco e 176 linhas de código para configurar 12 Roteadores Cisco. Já no processo de configuração automatizado por Python foi necessário 27 linhas de código para configurar 12 Switches Cisco e 19 linhas de código para configurar 12 Roteadores Cisco. Além disso, foi calculado o percentual de decréscimo entre os dois procedimentos, resultando em uma melhoria expressiva quando se utiliza um Script para automação de redes com Python.

Número de Linhas de Configuração e Percentual de Decréscimo			
Procedimento	**Manual**	**Automatizado**	**Decréscimo**
Switches	769	27	96%
Roteadores	176	19	89%

Tabela 1 - Linhas de Configurações e Percentual de Decréscimo.

CONCLUSÕES

O correto funcionamento do cenário implantado no GNS3 possibilitou a realização em ambiente real. Assim, pode ser realizada a comparação entre o processo manual e automatizado, da

implantação ou a modificação da rede. O protocolo OSPF foi utilizado de maneira bem sucedida, tanto no ambiente emulado, quanto nos equipamentos reais. Assim, pode-se definir que, a automação com o uso de Script em Python obteve êxito, tanto no ambiente emulado como nos equipamentos reais e utilizando uma menor quantidade de código de configurações, como descreve a Tabela 1. Foi avaliado também o percentual de decréscimo, entre o procedimento manual e automatizando, para os Switches esse decréscimo foi de 96%, e para os Roteadores 89%.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BRITO, S. H. B. Laboratórios de Tecnologias Cisco em Infraestrutura de Redes: Laboratórios do Primeiro Brasileiro Certificado em IPv6. [s.l.] Novatec Editora, 2014.

[2] CARDOSO, W. S. E. O. G. A. M. T. G. S. M. P. S. et al. Uso de Redes Definidas por Software para Controle de Roteamento. 2016.

[3] SOUSA, M. P. T. O. J. S. W. S. Y. C. U. R. Simuladores e Emuladores de Rede para o Projeto e Solução de Problemas em Ambientes de Produção. Revista de Tecnologia da Informação e Comunicação, v. 6, n. 2, 2016.

[4] GORDON, M. An introduction to network programming the python way [book review]. IEEE Distributed Systems Online, v. 6, n. 10, 2005.

[5] JAMBUNATHA, K. Design and implement Automated Procedure to upgrade remote network devices using Python. Advance Computing Conference (IACC), IEEE International. Anais. 2015.

[6] Nosrati, M. 2011. Python: An appropriate language for real world programming. World Applied Programming 110–117, 1 (2011).

[7] Tischer, R. J. G. Programming and automating Cisco networks. Cisco Press. 2017.

[8] WELSH, C. GNS3 network simulation guide. [s.l.] Packt Publishing. 2013.

[9] Vasconcelos, R. C. M. C. R. Gomes, F. B. F. A. Costa, and C. D. D. Silva, “Towards an SDN-App Store: A Generic Northbound API for Software-Defined Networks,” in Proceedings of the 14th International Conference WWW/INTERNET 2015 (ICWI), (Maynooth, Dublin, Ireland), pp. 173–176, IADIS, October 2015.