Programa de Cálculo de Rede
Software que realiza cálculos básicos utilizados durante a implementação de redes de computadores

Rafael Rampim Soratto[1], **Vinicius Petris**[1]

[1] Bacharelado em Ciência da Computação UTFPR -
Universidade Tecnológica Federal do Paraná. Campus Campo Mourão.

soratto@alunos.utfpr.edu.br, viniciusbosa@gmail.com

Abstract. *The software must receive as input for processing a JSON file with the following information: IP and netmask, which should be respectively called 'ipAddr' and 'netMask'. Such file is passed by parameter via command line when running the software. The JSON file must have IP address and NetMask information for the software to perform checks and validations.*

Resumo. *O software precisa receber como entrada, para processamento, um arquivo JSON com as seguintes informações: IP e máscara de rede, que devem ser chamados respectivamente de 'ipAddr' e 'netMask'. Tal arquivo é passado por parâmetro via linha de comando, ao executar o software. O arquivo JSON deve obrigatóriamente ter as informações do endereço IP e da NetMask para o software realizar as verificações e validações.*

1. Introdução

1.1. JSON

JSON (JavaScript Object Notation - Notação de Objetos JavaScript) é uma formatação leve de troca de dados. Para seres humanos, é fácil de ler e escrever. Para máquinas, é fácil de interpretar e gerar. JSON está constituído em duas estruturas: Uma coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um object, record, struct, dicionário, hash table, keyed list, ou arrays associativas. Uma lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma array, vetor, lista ou sequência. [1].

1.2. JSON em Python

A biblioteca JSON em python permite a construção de estruturas semelhantes para a construção dos IP's e das máscaras. [2]

[1] https://www.json.org/json-pt.html
[2] https://docs.python.org/3/library/json.html

2. Endereçamento IP

No modelo TCP/IP o endereçamento IP possui grande importância na camada de inter-redes pois fornece uma forma simples e eficiente de endereçamento e roteamento para os pacotes na internet. O enderaçamento IP trata-se de um endereçamento lógico para indentificar dispositivos e a rede a qual ele pertence.

O IPV4 é utilizado hoje em dia e fornece 2^{32} endereços tirando os endereços reservados. Um endereço identifica uma conexão com a rede e não exatamente um computador. Duas máquinas nunca possuem um mesmo endereço IP. O endereço de LoopBack é um endereço que será tratado localmente pela máquina.

O Endereço IP permite classificar tanto a rede a qual um host pertence como também especificar o host. Isso chama netID e hostID. Quando ambos estão em 1 então trata-se de broadcast quando ambos estão em 0 trata-se do endereço de origem inicial.

2.1. Esquema original de roteamento IP Classfull

Neste esquema a classe de um endereço é determinada pelos 3 bits de alta ordem. Durante a implementação em Python foi utilizada a seguinte verificação para as classes:

```
def classeIp(ipAddr):
    if(ipAddr[0] <= 127):
        print("Classe A")
        string = "A"
    elif(ipAddr[0] >= 128 and ipAddr[0] <= 191):
        print("Classe B")
        string = "B"
    elif(ipAddr[0] >= 192 and ipAddr[0] <= 223):
        print("Classe C")
        string = "C"
    elif(ipAddr[0] >= 224 and ipAddr[0] <= 239):
        print("Classe D")
        string = "D"
    elif(ipAddr[0] >= 240 and ipAddr[0] <= 255):
        print("Classe E")
        string = "E"
    return string
```

Listing 1. Verificação de Classes de endereços

2.2. IP's reservados

```
def reservado(ipAddr):
    if(ipAddr[0] == 127):
        print("Endereco de loopback")
        string = "Endereco de loopback"
```

```
    elif(ipAddr[0] == 10):
        print("Ip reservado")
        string = "Ip reservado"
    elif(ipAddr[0] == 172 and (ipAddr[1] >= 16 or ipAddr[1] <= 31)):
        print("Ip reservado")
        string = "Ip reservado"
    elif(ipAddr[0] == 192 and ipAddr[1] == 168):
        print("Ip reservado")
        string = "Ip reservado"
    elif(ipAddr[0] == 169 and ipAddr[1] == 254):
        print("Ip reservado")
        string = "Ip reservado"
    else:
        string = "Ip nao reservado"
    return string
```

Listing 2. Verificação Endereços Reservados

2.3. Entrada

A entrada será apenas o arquivo JSON contendo as informações sobre o endereço IP e sobre a máscara net. O arquivo teste.json contém a seguinte entrada:

```
{
    "ipAddr": "192.168.0.34",
    "netMask": "255.255.192.0"
}
```

Listing 3. Exemplo de entrada JSON

Á partir das informações no arquivo teste.json deve-se realizar o processamento para verificar os seguintes items:

1. Verificar se IPs e máscaras são válidas.
2. Quantidade de bits da rede (netID) e quantidade de bits de host (hostID), da máscara.
3. Classe do IP.
4. IP da rede.
5. IP de broadcast.
6. Quantidade de hosts na referida rede.
7. Faixa de máquinas válidas - que podem ser utilizadas pelos hosts.
8. Se o IP é em questão é reservado (privado, loopback, etc).
9. Apresentar a saída, referente as questões anteriores: (i) na tela e (ii) em um arquivo JSON. O nome dos campos no arquivo JSON devem ser intuitivos.

3. Executar projeto

A implementação do projeto foi realizada na linguagem Python 3. Portanto, para execução do mesmo basta utilizar o comando:

python3 apsredes.py

4. Implementação

4.1. Leitura do arquivo

```
# json package
import json
# load file
def lerArquivo():
    try:
        # recebe a entrada json
        arquivo_json = open('teste.json', 'r')
        # converte para json
        dados_json = json.load(arquivo_json)
        # salva ipAddr
        ipAddr = dados_json['ipAddr']
        # salva netMask
        netMask = dados_json['netMask']
        # une ipAddr e netMask
        vetSplit =  split(ipAddr, netMask)
        # une e faz cast
        vetCast = cast(vetSplit[0], vetSplit[1])
        return vetCast[0], vetCast[1]
    # se houver erro
    except Exception as erro:
        print("Ocorreu um erro ao carregar o arquivo")
        # mostra o erro
        print("Erro: {}". format(erro))
```

Listing 4. Lendo arquivo json utilizando python

5. Função Principal

A função principal une as outras funções de leitura e validação dos IP's, portanto ela representa a estrutura do programa:

```
def salvarJSON():
    ler = lerArquivo()
    valida = validar(ler[0], ler[1])
    if(valida == 1):
        print("Ip invalido")
    elif(valida == 2):
        print("Mascara invalida")
    else:
        qtde = netID_hostID(ler[1])
        classe = classeIp(ler[0])
        rede = ipRede(ler[0], ler[1])
        broadcast = ipBroadcast(rede, ler[1])
        interv = intervalo(rede, broadcast)
        reserv = reservado(ler[0])
        dicionario = {
            'Bits_Rede': qtde[0],
            'Bits_Hosts': qtde[1],
            'Hosts_na_Rede': qtde[2],
            'Classe:': classe,
```

```
            'Ip_Rede': str(rede[0])+"."+str(rede[1])+"."+str(rede[2])+"."+str(rede[3]),
            'Ip_Broadcast': str(broadcast[0])+"."+str(broadcast[1])+"."+str(broadcast[2])+"."+str(broadcast[3]),
            'Faixa_Maquina_Validas_Inicial': str(interv[0][0])+"."+ str(interv[0][1])+"."+ str(interv[0][2])+"."+ str(interv[0][3]),
            'Faixa_Maquina_Validas_Final': str(interv[1][0])+"."+ str(interv[1][1])+"."+ str(interv[1][2])+"."+ str(interv[1][3]),
            'Tipo_IP:': reserv
        }
        with open('resultado.json', 'w') as f:
            json.dump(dicionario, f)
```

Listing 5. Gerando a resposta JSON

Após a execução do programa à cima com a entrada utilizada anteriormente, teremos o seguinte resultado:

```
{
    "Bits_Rede": 18,
    "Bits_Hosts": 14,
    "Hosts_na_Rede": 16382,
    "Classe:": "C",
    "Ip_Rede": "192.168.0.1",
    "Ip_Broadcast": "192.168.63.254",
    "Faixa_Maquina_Validas_Inicial": "192.168.0.1", "Faixa_Maquina_Validas_Final": "192.168.63.254",
    "Tipo_IP:": "Ip reservado"
}
```

Listing 6. Exemplo de saída JSON

6. Conclusão

De acordo com o presente trabalho é possível visualizar o funcionamento do endereçamento e roteamento utilizando a tecnologia IP. Durante o desenvolvimento foi apresentada uma solução para receber um endereço e uma máscara e retornar informações como bits da rede, bits de host, quantidade de hosts na rede, a classe do endereço (A,B, C ou D), o IP da rede, IP de broadcast, o endereço da máquina inicial e final e o tipo de IP. Com essas informações é possível visualizar como funciona o endereçamento e classificação dos endereços IP's.

Referências

[JSON PYTHON, 2019] Exemplos de JSON em python, disponível em : https://www.w3schools.com/python/python_json.asp