Projeto de Redes 2: Projeto e Simulação de Rede Hierárquica

João Victor Silva Araruna - 232012956
Departamento de Ciência da Computação
Universidade de Brasília
Brasília, Brasil
232012956@aluno.unb.br

***Abstract*—Este relatório detalha o projeto e a simulação de uma rede de data center com topologia hierárquica. Foi realizado o planejamento de endereçamento IP utilizando VLSM, a definição de tabelas de roteamento estático e a implementação de um simulador em Python para validar a conectividade da rede através dos comandos xping e xtraceroute.**

***Index Terms*—Roteamento Estático, VLSM, Simulação de Redes, Topologia Hierárquica**

I. Introdução

Na era digital atual, a vasta maioria dos serviços online, desde redes sociais a aplicações empresariais, depende da infraestrutura robusta e escalável dos data centers. O desempenho e a confiabilidade destes data centers estão intrinsecamente ligados à eficiência de sua rede interna. Portanto, o planejamento cuidadoso da topologia de rede, do endereçamento IP e das políticas de roteamento é um pilar fundamental na área de redes de computadores [1].

Muitos data centers modernos empregam uma topologia de rede hierárquica, ou em árvore, composta por camadas de Core, Agregação e Borda para gerenciar o tráfego de forma eficiente. Este trabalho, desenvolvido no âmbito da disciplina de Redes de Computadores, teve como objetivo principal aprofundar os conceitos das camadas de rede e de enlace através da aplicação prática em um projeto de rede. O desafio consistiu em projetar um plano de endereçamento IP utilizando a técnica de Máscaras de Sub-rede de Tamanho Variável (VLSM) para atender a requisitos específicos de hosts em diferentes sub-redes, bem como definir as tabelas de roteamento estático para garantir a conectividade completa entre todos os dispositivos.

Para validar o projeto teórico da primeira fase, foi desenvolvida uma simulação da topologia em Python, utilizando a biblioteca `networkx` para a representação do grafo da rede [2]. Esta simulação permite a execução de comandos customizados, `xping` e `xtraceroute`, que verificam a alcançabilidade entre hosts e traçam as rotas que os pacotes percorrem, respectivamente. A execução bem-sucedida destes comandos demonstra a corretude do plano de endereçamento e das tabelas de roteamento implementadas.

Este relatório está organizado da seguinte forma. A Seção II aborda a fundamentação teórica que serviu de base para o projeto, incluindo conceitos de endereçamento e roteamento. A Seção III descreve o ambiente experimental, detalha a topologia configurada e apresenta a análise dos resultados obtidos com a simulação. Finalmente, a Seção IV sumariza as conclusões técnicas do trabalho realizado.

II. Fundamentação Teórica

Nesta seção, são detalhados os conceitos teóricos essenciais que foram aplicados para o projeto e implementação da rede simulada. O entendimento destes pilares foi crucial para o planejamento do endereçamento, a definição das rotas e a validação do sistema.

A. Endereçamento IP e VLSM

O Protocolo de Internet (IP), em sua versão 4 (IPv4), utiliza um endereço de 32 bits para identificar unicamente cada dispositivo em uma rede [1]. Este endereço é dividido em uma porção de rede e uma porção de host, cuja separação é definida pela máscara de sub-rede. Para otimizar a distribuição de endereços em redes com necessidades variadas, a técnica de Máscaras de Sub-rede de Tamanho Variável (VLSM) foi empregada. O VLSM permite que um único bloco de endereços seja subdividido em múltiplas sub-redes de tamanhos diferentes, evitando o desperdício de IPs. No presente projeto, esta técnica foi fundamental para alocar eficientemente sub-redes com máscara /27, para as redes de borda com maior número de hosts, e /30, para os links ponto-a-ponto entre os roteadores. A validação matemática dessas redes na simulação foi realizada com o auxílio da biblioteca `ipaddress` do Python [3].

B. Roteamento Estático

O roteamento é o processo de encaminhar pacotes entre redes distintas. O roteamento estático consiste na configuração manual das rotas na tabela de roteamento de cada roteador por um administrador de rede [1]. Diferente dos protocolos de roteamento dinâmico, que descobrem rotas automaticamente, o roteamento estático oferece simplicidade, segurança e menor consumo de recursos do roteador. Devido à topologia fixa e ao ambiente controlado do data center simulado, o roteamento estático foi a escolha ideal para este projeto, garantindo que os caminhos dos pacotes fossem predefinidos e determinísticos.

C. Simulação de Redes com Grafos

A topologia de uma rede de computadores pode ser elegantemente modelada como um grafo, uma estrutura matemática

composta por nós (vértices) e links (arestas) [2]. Nesta representação, os roteadores e hosts são os nós, e as conexões físicas ou lógicas são as arestas. Para a Fase 2 deste projeto, a biblioteca `networkx` do Python foi utilizada para construir essa representação em memória. Esta abordagem permitiu a implementação de algoritmos de travessia de grafo para simular o comportamento de ferramentas como o `traceroute`, validando o caminho que um pacote percorreria através da rede com base nas tabelas de roteamento definidas.

III. Ambiente Experimental e Análise de Resultados

Esta seção descreve em detalhes o ambiente utilizado para o projeto, as configurações de rede aplicadas e a análise dos resultados obtidos através da simulação, validando o planejamento teórico.

A. Descrição do Cenário

A construção e validação do projeto foram realizadas utilizando um conjunto de softwares de código aberto em um ambiente Linux. A coerência entre os conceitos da Seção II e a prática descrita aqui é um pilar deste trabalho.

1) Hardware e Software: O ambiente experimental foi configurado com os seguintes componentes:

- **Sistema Operacional:** Ubuntu 22.04 LTS
- **Linguagem de Programação:** Python 3.10
- **Ambiente Virtual:** `venv` do Python, para isolamento de pacotes.
- **Bibliotecas Python:** `networkx` para a modelagem da rede como um grafo e `ipaddress` para os cálculos de rede.
- **Ferramenta de Diagramação:** draw.io para a criação da topologia de rede.
- **Editor de Relatório:** Overleaf (LaTeX).
- **Repositório do Código:** O código-fonte e os arquivos do projeto estão disponíveis publicamente no GitHub em este link.
- **Vídeo de Demonstração:** A apresentação do projeto e a demonstração da simulação estão disponíveis em este link do Google Drive.

2) Topologia de Rede: A rede projetada segue uma topologia hierárquica em árvore, comum em data centers, composta por três camadas: Core, Agregação e Borda. O diagrama completo da rede, incluindo a disposição dos equipamentos e o endereçamento IP de cada interface, é apresentado na Figura 1.

3) Configurações: A configuração central do ambiente consistiu no plano de endereçamento IP e nas tabelas de roteamento estático. O plano, baseado em VLSM, foi projetado para atender aos requisitos específicos de hosts de cada sub-rede. As tabelas de roteamento estático foram definidas manualmente para cada um dos 7 roteadores, garantindo um fluxo de pacotes determinístico e controlado. A Figura 2 apresenta um resumo consolidado de todas as tabelas de roteamento definidas para o projeto.

B. Análise de Resultados

IV. Conclusões

Este projeto demonstrou com sucesso a aplicação prática de conceitos fundamentais de redes de computadores em um cenário de data center. A utilização da técnica VLSM provou ser essencial para o planejamento e a distribuição eficiente de endereços IPv4, atendendo a múltiplos requisitos com o mínimo de desperdício. A definição de tabelas de roteamento estático, por sua vez, garantiu um fluxo de dados determinístico e controlado, ideal para a topologia fixa proposta. A validação de todo o projeto através de um simulador em Python foi crucial, não apenas para verificar a conectividade, mas também para depurar a lógica de roteamento de forma interativa, confirmando que o planejamento teórico se traduziu em uma rede funcional. O trabalho evidencia a importância do planejamento detalhado como pilar para a implementação de redes robustas e eficientes.

References

[1] J. F. Kurose e K. W. Ross, *Redes de Computadores e a Internet: Uma Abordagem Top-Down*, 6ª ed. São Paulo: Pearson, 2013.

[2] A. A. Hagberg, D. A. Schult, e P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," em *Proceedings of the 7th Python in Science Conference*, 2008, pp. 11–15.

[3] Python Software Foundation. *ipaddress — IPv4/IPv6 manipulation library.* Documentação Oficial do Python 3. Acessado em: 15 de julho de 2025. [Online]. Disponível: https://docs.python.org/3/library/ipaddress.html

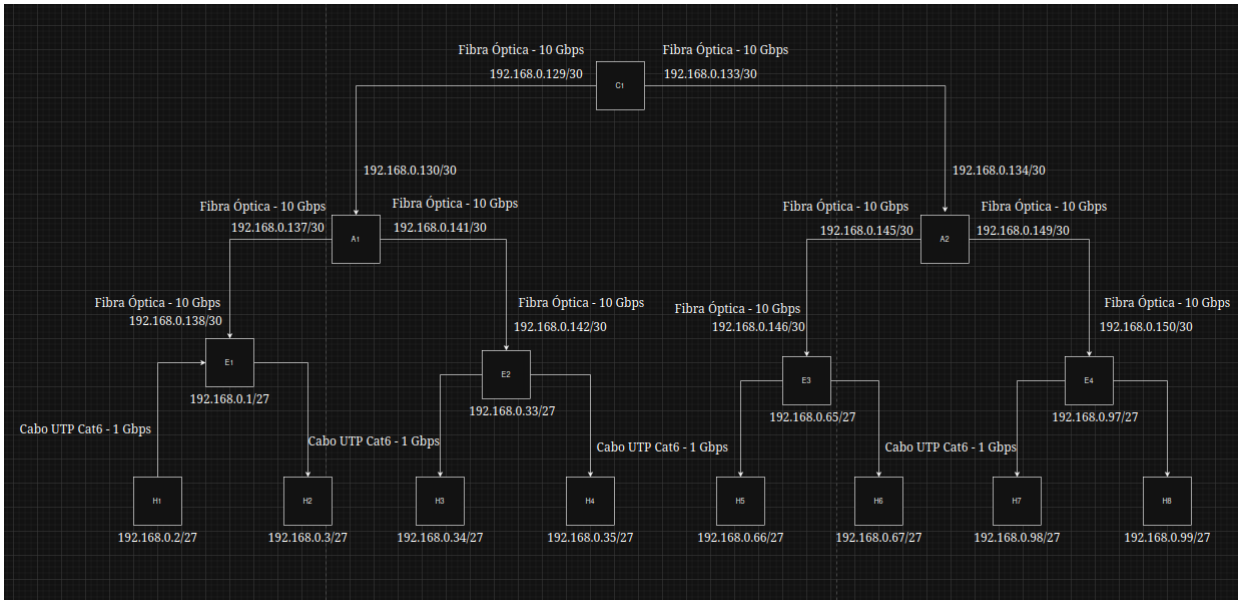

Fig. 1. Diagrama da topologia de rede projetada, com endereçamento IP e especificações de enlace.

Fig. 2. Tabelas de roteamento estático para todos os roteadores da rede.

Tabela A1

Destino	Máscara	Próximo Salto
192.168.0.0	/27	192.168.0.138
192.168.0.32	/27	192.168.0.142
0.0.0.0	/0	192.168.0.129

Tabela A2

Destino	Máscara	Próximo Salto
192.168.0.64	/27	192.168.0.146
192.168.0.96	/27	192.168.0.150
0.0.0.0	/0	192.168.0.133

Tabela E1

Destino	Máscara	Próximo Salto
0.0.0.0	/0	192.168.0.137

Tabela E3

Destino	Máscara	Próximo Salto
0.0.0.0	/0	192.168.0.145

Tabela E2

Destino	Máscara	Próximo Salto
0.0.0.0	/0	192.168.0.141

Tabela E4

Destino	Máscara	Próximo Salto
0.0.0.0	/0	192.168.0.149

Tabela C1

Destino	Máscara	Próximo Salto
192.168.0.0	/27	192.168.0.130
192.168.0.32	/27	192.168.0.130
192.168.0.64	/27	192.168.0.134
192.168.0.96	/27	192.168.0.134