Universidade Federal de Santa Maria - UFSM
Centro de Tecnologia - CT
Curso de Graduação em Sistemas de Informação
Disciplina: Redes de Computadores
Ana Clara Bordin, Felipe Colpo,
Marcos Eduardo Kaminski e Rafael Klaue

RELATÓRIO DO PRIMEIRO TRABALHO DA DISCIPLINA DE REDES DE COMPUTADORES: UMA SIMULAÇÃO DE REDE COM MININET

Sumário

Introdução

Neste trabalho, utilizamos o Mininet para criar um ambiente de simulação de redes, permitindo a análise e o teste de algoritmos de roteamento. Foram desenvolvidas diversas topologias de rede (linha, anel, estrela, malha e híbrida) para explorar diferentes cenários de conectividade e desempenho. O objetivo principal foi implementar e avaliar algoritmos de roteamento, como Distância-Vetor e Estado do Link, destacando sua capacidade de atualizar dinamicamente tabelas de rotas com base em informações trocadas entre os roteadores.

A simulação foi realizada em uma máquina virtual configurada com suporte ao Mininet e ao Python, utilizando o Scapy para manipulação de pacotes personalizados. Ferramentas como Ping, Traceroute e medição de latência permitiram avaliar o desempenho da rede nos diferentes cenários simulados.

Este relatório apresenta a metodologia utilizada, as topologias implementadas, os resultados obtidos e as conclusões alcançadas, bem como reflexões sobre desafios enfrentados e possibilidades de aprimoramento. A análise conduzida busca contribuir para a compreensão de algoritmos de roteamento em ambientes simulados e seu impacto no desempenho de redes de computadores.

Metodologia

O desenvolvimento do projeto foi realizado inteiramente em um ambiente integrado entre Visual Studio Code e a Máquina Virtual. Para a implementação da simulação foi escolhida a linguagem Python. A compilação e execução do código foram realizadas no terminal da Máquina Virtual, a fim de acompanhar cada evolução do projeto.

Inicialização do ambiente com o Mininet

O projeto foi desenvolvido utilizando uma Máquina Virtual configurada com suporte ao Mininet, seguindo as orientações fornecidas pelo professor. Para facilitar o desenvolvimento e a edição do código, integrou-se o ambiente ao Visual Studio Code, onde foram escritos os códigos em Python, permitindo um fluxo contínuo de trabalho entre a criação, teste e a execução do código. A primeira etapa envolveu a instalação das ferramentas necessárias:

Mininet: Utilizado para a simulação de redes e topologias.
Scapy: Foi necessária a instalação dessa biblioteca pois optamos por desenvolver o projeto inteiramente na linguagem Python.
VirtualBox: Utilizado para hospedar a Máquina Virtual porque oferece uma plataforma para rodar o nosso ambiente simulado.

Switches e hosts - a simulação do tráfego

Cada **host** foi configurado como uma máquina virtual dentro do Mininet, com uma interface de rede simulada. No código, os hosts foram criados dinamicamente com a função *self.addHost()*, permitindo a criação de múltiplos dispositivos conforme a necessidade de cada topologia. Em nosso caso, foi utilizado um número de 10 hosts em cada topologia para representar uma rede simples, mas escalável. Estes hosts se comunicam uns com os outros através de switches, trocando pacotes e executando comandos como ping e traceroute para testar a conectividade entre eles.

Os **switches** são os dispositivos responsáveis pelo encaminhamento de pacotes dentro da rede, conectando os hosts uns aos outros. No Mininet, os switches foram definidos como instâncias do OVSKernelSwitch, que utilizam o Open vSwitch (OVS) para gerenciar as tabelas de encaminhamento de pacotes. A configuração dos switches nas topologias foi feita com a função *self.addSwitch*(), e a interconexão entre switches foi organizada conforme o tipo de topologia simulada (linha, anel, estrela, malha ou híbrida). Cada switch, quando conectado a um host ou a outro switch, simula

o comportamento de dispositivos de rede reais em um ambiente controlado, permitindo a análise do desempenho da rede.

Os switches foram conectados de maneira estratégica de acordo com a topologia escolhida. Em topologias como a estrela, todos os hosts foram ligados a um único switch central, enquanto em topologias como a malha completa, cada host foi interconectado diretamente a todos os outros hosts.

Essa configuração de switches e hosts permitiu a criação de redes com diferentes características e topologias, possibilitando a análise do desempenho de cada uma, especialmente em relação ao algoritmo de roteamento desenvolvido.

As topologias implementadas

Neste projeto, foram implementadas cinco diferentes topologias de rede, cada uma com características específicas que impactam a forma como os dados são roteados e a rede se comporta em termos de desempenho e escalabilidade. As topologias implementadas são **linha**, **anel**, **estrela**, **mesh** e **híbrida (linha e estrela)**. Cada uma delas foi escolhida para simular diferentes cenários de rede, com o objetivo de avaliar o comportamento do algoritmo de roteamento sob condições variadas.

Da implementação das topologias

Topologia em Linha:

A topologia em linha foi implementada utilizando uma sequência de hosts e switches conectados de maneira linear. No código, os hosts são conectados a switches intermediários, e a comunicação entre os hosts segue uma rota fixa através dos switches. A principal característica dessa topologia é que os dados precisam percorrer múltiplos saltos para alcançar o destino, o que a torna ideal para testar o algoritmo de roteamento em cenários de comunicação sequencial.
O código cria uma série de X hosts e X-1 switches, onde cada host é conectado ao switch mais próximo, e os switches são interligados em sequência. A latência entre os links também é definida aleatoriamente, o que permite a simulação de diferentes condições de rede. A escolha dessa topologia foi motivada pela sua simplicidade e pelo fato de que ela facilita a análise da latência e do desempenho do algoritmo de roteamento, já que os pacotes de dados precisam passar por múltiplos dispositivos (switches) para chegar ao destino.

Topologia em Anel:

A topologia em anel foi implementada com a ideia de formar uma rede onde cada switch é conectado ao próximo de forma circular, com os hosts conectados a cada switch. Nesse cenário, a comunicação entre os dispositivos segue um caminho cíclico, o que pode introduzir desafios no roteamento, especialmente se houver falhas em algum dos links. A topologia em anel é útil para testar o comportamento do algoritmo de roteamento quando a rede possui redundâncias e falhas potenciais.

Topologia em Estrela:

A topologia em estrela foi projetada para garantir que todos os hosts se conectem a um único switch central. Esse tipo de topologia é amplamente utilizado em redes locais, pois oferece facilidade de gerenciamento e simplicidade na implementação. No código, foi utilizado um único switch para conectar os 10 hosts, e todos os links entre hosts e o switch central têm um atraso aleatório.

Topologia Mesh:

A topologia em malha foi implementada com o objetivo de criar uma rede onde cada host está diretamente conectado a todos os outros hosts. Essa topologia oferece redundância e permite múltiplos caminhos para o tráfego de dados, o que a torna ideal para testar o algoritmo de roteamento em cenários de alta disponibilidade. No código, a topologia de malha é construída conectando todos os hosts uns aos outros, o que garante que os dados possam seguir vários caminhos até o destino.

Topologia Híbrida (Linha e Estrela):

A topologia híbrida combina as topologias em linha e estrela, com o objetivo de criar uma rede mais complexa e escalável. Ela foi implementada no código conectando três switches entre si e distribuindo os hosts entre esses switches, com 5 hosts conectados a cada switch. Essa topologia é interessante para testar o algoritmo de roteamento em um cenário mais realista, onde diferentes tipos de conectividade e topologias coexistem.

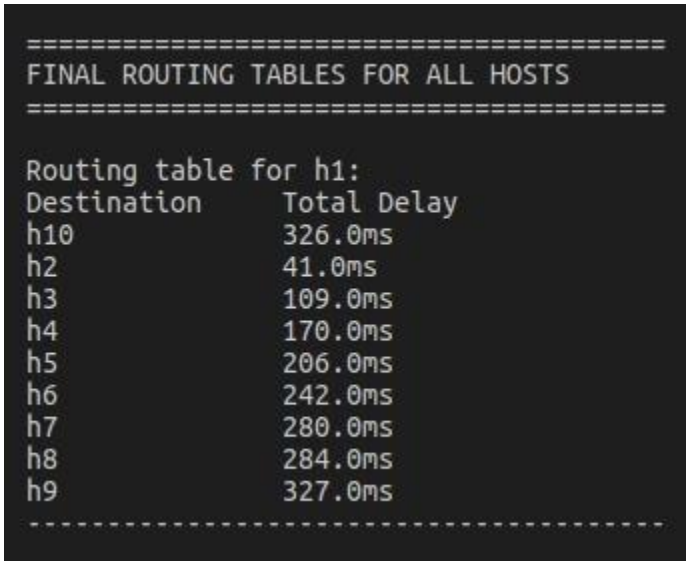
A implementação de uma tabela de distância

Para fins de comparação, foi implementada uma tabela de distâncias, que tem como objetivo representar as rotas e seus custos dentro da rede. Essa tabela foi essencial para simular o comportamento da rede e validar a funcionalidade do algoritmo de roteamento implementado. A partir dela, foi possível verificar como as distâncias entre os hosts e switches mudam durante a execução das simulações.

Além disso, foram utilizados três comandos principais no terminal da Máquina Virtual para avaliar o desempenho e a conectividade da rede. O comando “**HX Ping HY**” (por exemplo, para testar o Host 1 e 2 teríamos: “H1 Ping H2”) foi utilizado para testar a

latência entre os hosts, permitindo verificar a resposta dos dispositivos em diferentes condições de rede. O **Ping All** foi executado para checar a conectividade entre todos os hosts na rede, fornecendo uma visão geral da rede e identificando possíveis falhas de comunicação. Já o comando **Traceroute** foi utilizado para mapear o caminho percorrido pelos pacotes até os destinos, fornecendo informações detalhadas sobre os roteadores e switches intermediários envolvidos na transmissão dos dados.

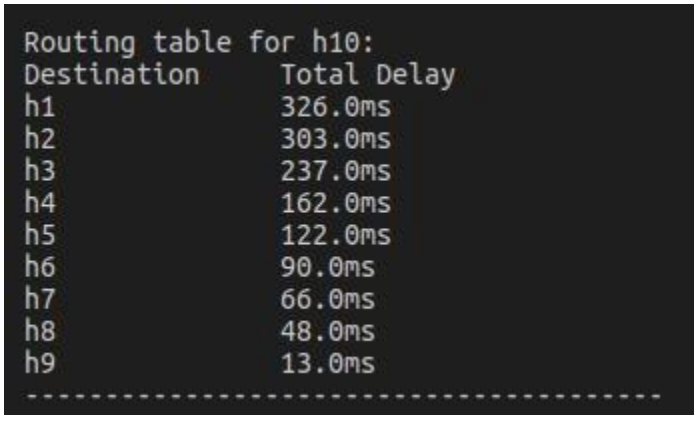
Esses comandos foram fundamentais para a análise da eficácia das rotas e para a verificação de como o algoritmo de roteamento gerenciava o tráfego na rede. As métricas obtidas a partir dos testes ajudaram a ajustar o comportamento do algoritmo, permitindo a avaliação da robustez e da eficiência das rotas calculadas. Com isso, foi possível validar a atualização dinâmica das tabelas de rotas e garantir a transmissão eficiente de dados na rede simulada.



```
========================================
FINAL ROUTING TABLES FOR ALL HOSTS
========================================

Routing table for h1:
Destination     Total Delay
h10             326.0ms
h2              41.0ms
h3              109.0ms
h4              170.0ms
h5              206.0ms
h6              242.0ms
h7              280.0ms
h8              284.0ms
h9              327.0ms
----------------------------------------
```

Tabela de distância da topologia em linha do Host 1



```
Routing table for h10:
Destination     Total Delay
h1              326.0ms
h2              303.0ms
h3              237.0ms
h4              162.0ms
h5              122.0ms
h6              90.0ms
h7              66.0ms
h8              48.0ms
h9              13.0ms
----------------------------------------
```

Tabela de distância da topologia em linha do Host 10

```
========================================
FINAL ROUTING TABLES FOR ALL HOSTS
========================================

Routing table for h1:
Destination     Total Delay
h10             133.0ms
h2              120.0ms
h3              86.0ms
h4              53.0ms
h5              116.0ms
h6              49.0ms
h7              53.0ms
h8              84.0ms
h9              65.0ms
----------------------------------------
```

Tabela de distância da topologia em Estrela do Host 1

```
Routing table for h10:
Destination     Total Delay
h1              133.0ms
h2              187.0ms
h3              153.0ms
h4              120.0ms
h5              183.0ms
h6              116.0ms
h7              120.0ms
h8              151.0ms
h9              132.0ms
----------------------------------------
```

Tabela de distância da topologia em Estrela do Host 10

```
Routing table for h1:
Destination     Total Delay
h10             11.0ms
h11             95.0ms
h12             92.0ms
h13             95.0ms
h14             92.0ms
h15             95.0ms
h2              6.0ms
h3              11.0ms
h4              7.0ms
h5              6.0ms
h6              12.0ms
h7              14.0ms
h8              11.0ms
h9              9.0ms
----------------------------------------
```

Tabela de distância da topologia Híbrida do Host 1

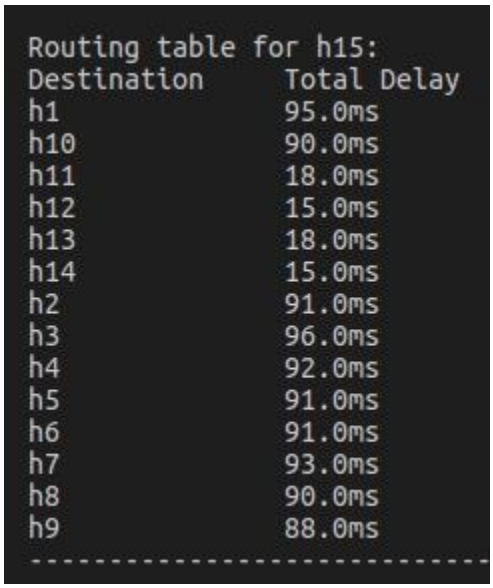


Tabela de distância da topologia Híbrida do Host 15

Resultados da tabela de distância e o que interpretamos

Ao termos as tabelas de distância executadas, foi possível enxergar como as topologias se comportam e analisar os resultados que mostram como os delays acumulados variam de acordo com a estrutura de cada rede. Para fins de interpretação, concentraremos nossa reflexão nas tabelas de distância das topologias de linha, estrela e híbrida (que aparecem nos prints acima), uma vez que estas representam diferentes padrões de conectividade e a híbrida que é uma junção das duas primeiras.

Topologia em Linha

A análise das tabelas de distância geradas para a topologia em linha revelou informações importantes sobre como os delays acumulam-se em uma rede com comunicação sequencial. Na topologia em questão, cada host está conectado de forma linear, o que significa que o tráfego de dados entre dois hosts precisa passar por todos os dispositivos intermediários, acumulando o delay de cada enlace:

H1: Está localizado no início da linha, e os tempos de delay aumentam progressivamente à medida que os pacotes percorrem mais saltos (hops) para alcançar hosts mais distantes.

H10: Está localizado no final da linha, e os tempos de delay diminuem à medida que os pacotes percorrem menos saltos para alcançar hosts mais próximos.

Por exemplo, a tabela de distâncias do host H1 mostra que o delay é menor para o host imediatamente próximo (H2: 41ms) e aumenta gradativamente até o host mais distante (H10: 326ms). Já a tabela de distâncias do host H10 apresenta o comportamento inverso, com o menor delay sendo registrado para o host próximo ao final da linha (H9: 13ms) e o maior delay para o host no início da linha (H1: 326ms).

Esse padrão de delays ocorre devido à natureza sequencial da topologia em linha, onde cada pacote precisa percorrer múltiplos saltos para alcançar o destino. A soma total dos delays é simétrica entre os pares de hosts, independentemente do sentido da comunicação, uma vez que os enlaces possuem delays fixos.

Essa observação permite validar o comportamento esperado da rede, demonstrando como a posição dos hosts em uma topologia linear impacta diretamente os tempos de resposta e a eficiência da comunicação. Além disso, os resultados reforçam a importância de um algoritmo de roteamento eficiente, capaz de calcular o caminho ideal mesmo em redes com alta latência cumulativa.

Dessa forma, interpretamos que os resultados obtidos são consistentes com a estrutura e os parâmetros definidos na implementação, e oferecem uma base sólida para análises futuras envolvendo outras topologias ou ajustes nos algoritmos de roteamento.

Topologia em Estrela

Já a análise das tabelas de distância na topologia em estrela evidencia diferenças significativas nos delays registrados para os hosts, dependendo do nó de origem da comunicação. Essas variações ocorrem devido à estrutura centralizada da topologia em estrela, onde todos os hosts estão conectados diretamente a um switch central.

Na tabela de distância do Host 1, os valores de delay para alcançar outros hosts refletem principalmente as latências dos enlaces diretos entre o switch central e cada host de destino. Por exemplo, para alcançar o Host 10, o delay total é de 133ms, que inclui apenas o delay do enlace de H1 ao switch e do switch até H10.

Por outro lado, a tabela de distância do Host 10 apresenta valores de delay maiores para vários hosts, como H2 (187ms) e H5 (183ms). Essa discrepância pode ser atribuída às latências variáveis configuradas nos enlaces individuais entre o switch central e os hosts. Como os valores de delay são gerados aleatoriamente no código, as rotas de ida e volta podem ter configurações de latência diferentes, ainda que a topologia seja simétrica.

Outra observação importante é que o delay do caminho entre dois hosts na topologia estrela não depende diretamente da distância física entre eles, mas sim da latência configurada nos enlaces entre o switch central e os hosts envolvidos. Isso explica por que os delays para alguns hosts próximos, podem ser maiores do que para hosts mais distantes no quesito de numeração, como h1 > h3 e h4 > h10.

Essas diferenças ilustram como a topologia em estrela distribui o impacto da latência em função da configuração específica dos enlaces ao switch central. O comportamento da rede nesta topologia destaca a importância do switch central como um ponto único de decisão e a influência significativa que os enlaces individuais têm no desempenho geral da comunicação.

Topologia Híbrida

A análise das tabelas de distância para a topologia híbrida revela diferenças significativas nos valores de *delay* entre os hosts, dependendo da parte da topologia (linha ou estrela) à qual os dispositivos estão mais próximos.

Na tabela do host 1, que está na extremidade da parte linha, os delays para os hosts vizinhos dessa mesma parte (H2 a H9) são baixos, variando de 6ms a 14ms. Isso reflete a proximidade lógica e física entre esses hosts, já que estão conectados sequencialmente.

Em contraste, os delays para os hosts da parte estrela (H10 a H15) são significativamente maiores, com valores em torno de 90ms a 95ms, devido à necessidade de atravessar toda a parte linha e o switch que conecta as duas topologias.

Já na tabela do host 15, que está na extremidade da parte estrela, os delays para os hosts vizinhos dessa mesma parte (H11 a H14) são os menores, variando de 15ms a 18ms. Isso se deve ao fato de que os hosts da estrela estão conectados diretamente ao switch central. Por outro lado, os delays para os hosts da parte linha são altos, na faixa de 90ms a 96ms, devido à maior distância lógica e ao número de saltos.

Também é possível enxergar uma certa simetria entre os maiores delays. Para ambos os hosts (1 e 15), os maiores valores de delay correspondem à comunicação com dispositivos localizados na parte oposta da topologia híbrida. Essa simetria reflete o impacto da distância e da estrutura lógica no tempo de resposta.

Interação entre linha e estrela:

A presença de um switch conectando as duas partes da topologia híbrida é um ponto crucial. Ele funciona como um intermediário que centraliza a comunicação entre a parte linha e a parte estrela. Essa característica garante a funcionalidade da rede, mas aumenta os delays devido à adição de saltos no trajeto.

As diferenças observadas são resultado direto do design da topologia híbrida, que combina as características de uma topologia linha e uma topologia estrela:

Parte linha: Prioriza uma comunicação sequencial eficiente, resultando em baixos delays para hosts conectados diretamente ou por poucos saltos. No entanto, a comunicação com a parte estrela é penalizada devido à necessidade de atravessar toda a linha.
Parte estrela: Permite uma comunicação rápida e eficiente entre hosts conectados ao switch central, mas apresenta altos delays para alcançar a parte linha, devido à distância lógica e ao trajeto adicional.

Conclusão e Perspectivas

Como resultado, desenvolvemos um código que atingiu quase todos os objetivos estipulados no trabalho, com algumas realizações a mais e outras a menos, além de resultados tanto esperados quanto inesperados. Por exemplo, ao analisar as tabelas de distância das topologias implementadas, foi possível identificar como os atrasos se acumulam e se distribuem de maneira característica em cada configuração de rede, algo que trouxe novas perspectivas para os conceitos estudados.

A implementação prática consolidou significativamente o entendimento dos temas abordados na disciplina, especialmente após a realização da primeira prova e com a chegada iminente da segunda. Ao observar o comportamento real dos algoritmos em execução, conseguimos ir além da teoria apresentada em slides e exemplos de aula, compreendendo de maneira mais profunda o impacto dos algoritmos de roteamento e das diferentes topologias no desempenho da rede.

Além disso, o trabalho proporcionou solidificar conhecimento sobre como diferentes parâmetros, como o número de hosts, a complexidade da conectividade e os atrasos nos links, influenciam a comunicação e a eficiência da rede. A experiência prática também destacou a importância do planejamento adequado das estruturas de rede e da escolha criteriosa dos algoritmos para atender às necessidades específicas de cada cenário.