# Projeto e Simulação de Topologia de Rede - Projeto 2

Carlos Cauã Rocha da Silva - 231034304, José Neto Souza de Lima - 231018955,
Luca Heringer Megiorin - 231003390

***Resumo*—O segundo projeto da disciplina de Redes de Computadores tinha como o foco simular os comandos ping e traceroute em uma topologia de rede. Neste trabalho foi promovida uma melhor compreensão da camada de redes e enlace, do protocolo IP e comandos do protocolo ICMP.**

***Palavras-Chave*—IP, traceroute, ping, protocolo**

## I. Introdução

DESDE a concepção de Internet, a organização da forma em que dispositivos deveriam se conectar era essencial, de modo que foram criados protocolos para a Camada de Redes a fim de suprir essa necessidade, sendo o IP (ou *Internet Protocol [1]* aquele que se mais se destacou e do qual outros protocolos, como o ICMP (ou *Internet Control Message Protocol* [2]), se baseiam e extendem a sua funcionalidade (chamados de protocolos da suíte IP - *Internet Protocol suite*).

Com isso, o segundo trabalho de Redes de Computadores visa aplicar os conhecimentos da Camada de Redes para projetar uma rede e simular os comandos ping e traceroute nela. Para isso, o grupo decidiu usar da linguagem Python para fazer o design da rede em forma de grafo, incluir tabelas e um algoritmo básico de roteamento, e simular uma interface do terminal linux que aceitasse os comandos pedidos, pertencentes ao ICMP.

Este artigo está dividido em seções, sendo esta a Introdução I. A seguir tem-se a Fundamentação Téorica II, que, com um embasamento teórico, explica os conceitos utilizados para a realização do projeto e permite melhor compreensão dos resultados obtidos; Ambiente Experimental e Análise de Resultados III, que descreve a estrutura da rede desenvolvida e os resultados da simulação; Conclusões IV, que fornecem pensamentos finais quanto ao projeto; e Bibliografia IV, para as referências presentes no artigo. Após as referências consta o código fonte e o link para o vídeo explicativo no Apêndice A.

## II. Fundamentação Téorica

A rede projetada é capaz de demonstrar o caminho percorrido pelos pacotes desde o host de origem até o destino final, revelando os diferentes roteadores e dispositivos intermediários envolvidos no processo de transmissão, por meio da simulação dos comandos ping ou traceroute.

Para atingir esse objetivo, foi fundamental que o projeto integrasse conhecimentos aprofundados sobre a Camada de Redes, conforme definida pela pilha de protocolos TCP/IP. Essa camada é responsável pela entrega de pacotes de dados entre sistemas finais, e uma compreensão detalhada de sua operação foi essencial para a implementação de uma simulação precisa e eficaz. Nesta seção serão descritas as decisões quanto à linguagem (II-A), aos protocolos (II-B), e a processo de roteamento (II-C).

### A. Linguagem e Bibliotecas

Para a construção do código, foi escolhida a linguagem Python, pela facilidade que esta proporciona para o desenvolvimento. Não houve necessidade de utilizar bibliotecas no código, de modo que o grupo criou as próprias funcionalidades para grafos e comandos requisitados pelo projeto.

### B. Protocolo IP e ICMP

Definido originalmente pela RFC760 [1], o IP (ou *Internet Protocol*) permite que os dispositivos na rede se identifiquem de forma única. Em uma rede privada com múltiplos endereços IP, torna-se fundamental a criação de sub-redes, a fim de melhorar a escalabilidade e facilitar a gerência da rede como um todo. Nesse contexto, utiliza-se a máscara de sub-rede, que define quais dispositivos pertencem à mesma rede local (sub-rede). A máscara de sub-rede permite distinguir quantos bits do endereço serão utilizados para identificar a rede e quantos serão destinados ao host, proporcionando um controle eficiente dos endereços IP dentro da rede.

No contexto de endereçamento IPv4, os endereços IP são tradicionalmente divididos em classes (A, B, C, D e E), cada uma com intervalos específicos para redes públicas, privadas ou especiais. No entanto, com a adoção do CIDR (*Classless Inter-Domain Routing*), a divisão rígida em classes foi substituída por uma abordagem mais flexível, permitindo a alocação eficiente de endereços IP.

Com base nos dados estabelecidos para a simulação, foi adotado o endereçamento 172.16.x.y para hosts e 10.1.x.y para os roteadores, que são faixas de endereços privados conforme definido no RFC 1918 [3], o que as torna ideais para uso em ambientes internos, que é o caso da simulação.

A escolha dessas faixas também foi influenciada pelas características físicas e lógicas da rede em questão, que serão melhor abordadas na seção III-A. Para redes que abrangem distâncias maiores, como 700 km, a faixa 10.1.x.y foi selecionada devido à sua ampla capacidade de endereçamento, que permite a criação de sub-redes maiores e a conexão de dispositivos distribuídos geograficamente. Por outro lado, para redes locais ou de menor alcance, como 10 km, a faixa 172.16.x.y foi escolhida por ser adequada para redes de tamanho médio.

Além disso, a distinção entre endereçamento permite maior escalabilidade tanto para hosts quanto para roteadores, facilitando a inserção ou remoção de dispositivos na rede. Nessa notação, x representa a sub-rede à qual o endereço pertence, enquanto y identifica o a interface dentro dessa sub-rede.

Para melhor entendimento do protocolo IP, pode-se observar a estruturação do datagrama IPv4:

TABELA I
TABELA DA ESTRUTURA DO DATAGRAMA IPV4

<table>
<tr><td colspan="6"><– 32 bits –></td></tr>
<tr><td>Versão</td><td>Tamanho do Cabeçalho</td><td>Tipo de Serviço</td><td colspan="3">Comprimento Total</td></tr>
<tr><td colspan="4">Identificação 16-bits</td><td>Flags</td><td>Offset de Fragmentação</td></tr>
<tr><td colspan="2">Tempo de Vida</td><td>Protocolo</td><td colspan="3">Checksum</td></tr>
<tr><td colspan="6">Endereço IP de Origem</td></tr>
<tr><td colspan="6">Endereço IP de Destino</td></tr>
<tr><td colspan="6">Opções (tamanho variável)</td></tr>
<tr><td colspan="6">Dados (não compõem o cabeçalho)</td></tr>
</table>

As principais funcionalidades de cada campo do datagrama IPv4 são:

- **Versão**: Versão IP que esta sendo utilizada (IPv4 ou IPv6);
- **Tamanho do cabeçalho**: Comprimento do cabeçalho de 32 bits;
- **Tipo de serviço**: Define a prioridade e o tipo de serviço desejado para o datagrama;
- **Tamanho total**: Tamanho do datagrama que esta sendo enviado, sendo cabeçalho + conteúdo em bytes;
- **Identificação 16-bits**: Identifica fragmentos do datagrama;
- **Flags**: Controle de fragmentação;
- **Offset de fragmentação**: Posição do fragmento no datagrama original;
- **Tempo de vida**: Pulos disponíveis ate o roteador destino;
- **Protocolo**: Protocolo utilizado da camada de transporte, podendo ser o protocolo TCP ou UDP;
- **Checksum do cabeçalho)**: Verifica a integridade do cabeçalho;
- **Endereço IP de Origem**: Endereço origem do datagrama;
- **Endereço IP de Destino**: Endereço destino do datagrama.
- **Opções**: Opcional para funcionalidades especiais;
- **Dados**: Os dados da camada superior.

Para o projeto, foi de grande importância a identificação do endereço IP de Origem e Destino, pois estes são utilizados nas tabelas de encaminhamento, explicadas na seção II-C.

Além do protocolo IP, o entendimento quanto ao protocolo ICMP (Internet Control Message Protocol [2]) também foi essencial. Integrante da pilha de protocolos TCP/IP, ele é utilizado principalmente para diagnóstico em redes IP, facilitando a compreensão de o que esta ocorrendo na rede.

Assim, dentro do ICMP, pode-se utilizar do ping (*Packet InterNet Groper*) para diagnostico da rede, pois este tem a função de testar a conectividade entre dois dispositivos em uma rede, enviando uma mensagem ICMP "*echo request*"para o endereço IP de destino. Caso o dispositivo de destino esteja ativo e a rede esteja corretamente configurada, ele responderá com uma mensagem ICMP "*echo reply*". O dispositivo de origem, ao receber essa resposta, mede o tempo decorrido entre o envio do "*echo request*"e o recebimento do "*echo reply*". Esse tempo é conhecido como latência, isto é, o tempo de ida e volta (RTT – Round-Trip Time). Além da latência, o ping fornece outras funcionalidades importantes, como o número de pacotes enviados e recebidos, assim como a taxa de perda de pacotes, o que pode indicar congestionamento na rede ou problemas de conectividade e, como será visto na seção (III-D), a diferença de enlace coaxial e de fibra óptica irá afetar os resultados, dado que a velocidade na fibra óptica é maior que a de cabos coaxiais.

O ICMP também auxilia na execução do comando traceroute. O traceroute é uma ferramenta essencial em redes de computadores que revela os saltos (*hops*) pelos quais os pacotes passam ao viajar entre um *host* de origem e um *host* de destino. Ele permite visualizar o caminho real percorrido pelos pacotes na rede, identificando cada roteador ou dispositivo intermediário que participa do encaminhamento. Essa funcionalidade é extremamente útil para diagnosticar problemas de rede, como falhas em determinados pontos da rota. Para o funcionamento do traceroute, três pacotes são enviados por vez com um tempo de vida (TTL - *time to live*) crescente a cada iteração, de modo que, na primeira vez que são enviados, possuem um TTL = 1 e medem o tempo apenas de um *hop* de ida e outro de volta, e assim sucessivamente até chegar no *host* de destino, no qual será devolvida uma mensagem ICMP "*dest port unreachable*", o que indica à origem que o pacote chegou até o destino final.

### C. Plano de Controle e Plano de Dados

Como visto nos capítulos 4 e 5 do livro *Computer Networking: A Top-Down Approach*, 8ª edição, dos autores Jim Kurose e Keith Ross Pearson [4], a camada de redes é dividida em dois planos: plano de dados e plano de controle.

O plano de dados tem como função determinar para onde os datagramas que chegam em um roteador devem ser encaminhados, com base nas decisões tomadas pelo plano de controle. Quando um datagrama chega a um roteador, o seu IP de destino é comparado com os endereços definidos na tabela de encaminhamento, de modo que o roteador escolhe a interface na qual ele deve sair baseado na correspondência mais longa de prefixo do IP (LPM - *longest prefix match*). Esse método garante que datagramas com um mesmo IP de destino irão sempre ser encaminhados para a mesma interface de saída caso a tabela de encaminhamento não seja modificada.

Enquanto que o plano de dados é responsável por decisões locais, apenas interessado em encaminhar um datagrama à interface correta, o plano de controle é responsável pela tomada de decisões de roteamento, ou seja, do caminho em que o datagrama irá percorrer da origem até o destino. Assim, dependendo da forma que a provedora de internet prioriza o roteamento, é possível que o plano de controle possua algoritmos estáticos, ou seja, aqueles que produzem uma tabela

de encaminhamento fixa independente de fatores externos, ou dinâmicos, que, dependendo de informações como distâncias, uso e congestionamento de enlaces, podem alterar periodicamente a tabela de encaminhamento. Assim, a critério das provedoras de internet, o custo para um caminho é definido, e, assim, é possível calcular o caminho de menor custo em uma rede, utilizando algoritmos como o Dijkstra (presente, por exemplo no protocolo OSPF - *Open Shortest Path First*). As etapas do algoritmo de Dijkstra estão definidas abaixo:

*1) Inicialização:*

- Atribua um custo inicial de 0 para o nó de origem e infinito para todos os outros nós.
- Crie uma lista de nós não visitados e uma lista de nós visitados.

*2) Seleção do Nó Atual:*

- Escolha o nó com o menor custo atual na lista de nós não visitados. Esse será o **nó atual** de referência.

*3) Atualização dos Custos:*

- Para cada vizinho do nó atual, calcule o custo total para alcançá-lo a partir do nó de origem.
- Se o custo calculado for menor que o custo armazenado, atualize o custo e defina o nó atual como o antecessor desse vizinho.
- A fórmula utilizada para a atualização é:

$$\text{Min}(custo\_atual, custo\_vizinho + peso\_da\_aresta)$$

*4) Marcação do Nó Atual:*

- Marque o nó atual como visitado e remova-o da lista de nós não visitados.

*5) Repetição:*

- Repita os passos 2 a 4 até que todos os nós tenham sido visitados ou até que o nó de destino seja alcançado.

*6) Reconstrução do Caminho:*

- Para encontrar o caminho mais curto, comece pelo nó de destino e siga os antecessores até o nó de origem.

Assim, ao utilizar o algoritmo acima, o plano de controle constrói uma tabela de encaminhamento, que armazenará agora o caminho ótimo para cada destino, pois, após o cálculo, o roteador saberá que o roteador ao qual ele deve encaminhar o seu datagrama recebido é o melhor caminho no momento. Esse algoritmo pode ou não ser repetido, de modo que as tabelas podem ser adaptadas conforme o tráfego aumenta ou são introduzidas ou retiradas conexões entre roteadores.

## III. Ambiente Experimental e Análise de Resultados

Com o embasamento teórico da seção II, o grupo projetou a rede de forma a atender as restrições topológicas propostas pelo roteiro do trabalho. Nesta seção, será discutido sobre a topologia da rede (III-A), hardware e software utilizados (III-B), uma explicação dos comandos ping e traceroute, bem como comandos que facilitam a simulação (III-C) e alguns exemplos da execução da simulação (III-D).

### A. Rede e Estrutura Básica

Seguindo as orientações do projeto a rede tem uma topologia em formato de árvore. A Figura 1 a seguir (imagem com melhor resolução no Apêndice A) esboça a rede projetada, de modo que as nuvens azuis indicam sub-redes (ou regiões com o mesmo endereço IP base). A partir desta figura, são definidos as seguintes categorias:

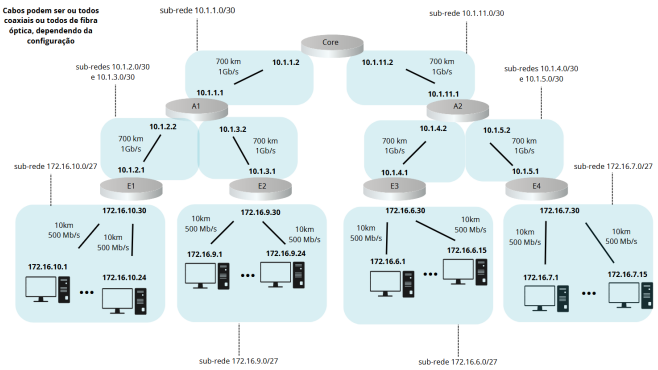


Figura 1: Topologia da rede desenvolvida: todos os cabos são do mesmo tipo

*1) Core:* Roteador que possui as interfaces definidas pelos IPs 10.1.1.2 e 10.1.11.2;

*2) Agregação:* Roteadores A1 e A2, cujas interfaces são definidas pelos IPs 10.1.1.1, 10.1.2.2 e 10.1.3.2 (para o roteador A1) e 10.1.11.1, 10.1.4.2 e 10.1.5.2 (para o roteador A2);

*3) Borda:* Roteadores E1, E2, E3 e E4, os quais possuem, respectivamente, as interfaces com IPs 10.1.2.1 e 172.16.10.30, 10.1.3.1 e 172.16.9.30, 10.1.4.1 e 172.16.6.30, e 10.1.5.1 e 172.16.7.30, de modo que a segunda interface de cada par definido corresponde à interface que interconecta os roteadores de borda com as suas respectivas sub-redes;

*4) Sub-redes e Sistemas Finais:* Para cada roteador de borda, há uma sub-rede. A sub-rede do roteador E1 é definida pelo endereço IP base 172.16.10.0/27, de forma que os seus sitemas finais (*hosts*) possuem IPs entre 172.16.10.1 e 172.16.10.24 (ou seja, a subrede aceita 24 *hosts*). Para a sub-rede do roteador E2, têm-se como endereço IP base 172.16.9.0/27 e aceita 24 *hosts* (entre 172.16.9.1 e 172.16.9.24), enquanto que a sub-rede do roteador E3 tem o endereço IP base 172.16.6.0/27 e aceita 15 *hosts* (entre 172.16.6.1 e 172.16.6.15). Por fim, a sub-rede do roteador E4 tem o endereço IP base 172.16.7.0/27 e aceita 15 *hosts* (entre 172.16.7.1 e 172.16.7.15).

É importante notar que as sub-redes definidas no item 4, por possuirem a mesma máscara (27), possuem 32 possíveis endereços de IP. Ao excluir o endereço base (terminado em 0), o endereço de broadcast (terminado com 31) e o endereço da interface dos respectivos roteadores (escolhidos para terem o final 30), sobrariam 29 endereços de IP para cada sub-rede, sendo essa a capacidade máxima de *hosts* em cada sub-rede. Todavia, para a simulação, decidiu-se que a rede seria estática (ou seja, não entrariam nem sairiam *hosts*, nem haveria mudanças na topologia dos roteadores) e que as sub-redes

dos roteadores E1 e E2 teriam exatamente 24 *hosts*, enquanto que as sub-redes dos roteadores E3 e E4 teriam exatamente 15 *hosts*. Enquanto isso, as sub-redes entre os roteadores foram definidas com a máscara 30, de modo que há apenas 2 endereços IP disponíveis para as interfaces dos roteadores, suficiente para a topologia projetada. As demais justificativas quanto ao endereçamento dos IPs foram discorridas na seção II-B.

Quanto à forma de cabeamento, o grupo criou o comando changenetwork, o qual permite que o usuário escolha realizar a simulação em uma rede com cabeamento completamento de fibra óptica ou coaxial, de forma a permitir resultados mais interessantes na simulação III-D. As distâncias entre roteadores são iguais, então o roteador Core está a 700 km de distância do roteador A1 e do roteador A2, e estes estão a 700 km de distância dos roteadores E1 e E2 (no caso do A1) e E3 e E4 (no caso do A2). Cada roteador da sub-rede (E1 até E4) estão a 10 km de distância de cada sistema final da sua sub-rede. Em razão dessas distâncias, a taxa de transmissão das saídas para enlaces de longos trajetos (de 700 km) são maiores, sendo definidas para 1 Gb/s, o que permitiria vários usuários simultâneos, enquanto que aquelas de menor distância (de 100 km) possuem taxa de 200 Mb/s, dado que são direcionadas para sistemas finais.

### *B. Hardware e Software*

Quanto ao hardware utilizado, o código foi implementado usando windows, e a versão Python 3.10.10. O grupo criou o grafo da rede, definindo classes como *Vertex* e *Link* para representar os *hosts*/roteadores e os enlaces que os conectam, respectivamente. A partir dessas classes, é possível identificar as interfaces com endereços IP para cada um dos roteadores e sistemas finais, bem como a distância, tipo de enlace e taxa de transmissão entre dois pontos.

Além disso, foi escrito o algoritmo de Dijkstra (explicado na seção II-C), que seria executado no plano de controle de cada roteador para montar a tabela de encaminhamento no plano de dados. Por se tratar de uma abstração, o algoritmo é executado apenas uma vez na inicialização da rede, dado que ela não sofre alterações topológicas ou de tráfego. Ao finalizar a execução desse algoritmo, a tabela de encaminhamento de cada roteador é gerada, de modo que, ao receber um datagrama, seria possível identificar, por meio do LPM, a interface de saída para a qual ele seria destinado. As tabelas são simuladas por meio de um dicionário, que analisa o IP de destino do datagrama e seleciona o IP da interface correta.

### *C. Serviços Oferecidos*

A rede desenvolvida possui diversos comandos de serviço. Ao inicializar o programa `main.py`, na pasta `src`, o padrão da rede é definido para fibra óptica e o algoritmo de Dijkstra é executado, gerando um dicionário que é guardado como atributo de cada vértice do grafo. Posteriormente, o programa simula uma interface do terminal linux, esperando assim os comandos. Além dos comandos ping e traceroute, o grupo implementou comandos que facilitam a simulação, e vale lembrar que os pacotes enviados possuem o tamanho de 60 bytes (informação usada para calcular tempos). Os comandos são descritos a seguir:

*1) Ping:* Simula o comando `ping` tradicional (descrito em II-B), em que o usuário pode testar a conectividade entre a posição atual e um destino qualquer dentro do grafo. A partir do IP de origem e de destino, o comando ping é capaz de acessar a tabela de encaminhamento para verificar o caminho até o destino, atualizar o endereço atual (informação usada na simulação para saber onde o pacote está no momento) até chegar no destino final, somando o tempo de ida e volta em cada salto. Ao executar o comando, são enviados 4 pacotes, de 60 bytes cada, e, com base na taxa de transmissão, distância e velocidade do enlace (difere entre coaxial e fibra óptica), são calculados os RTTs de cada pacote, e, para obter resultados mais interessantes de latência, é gerado um valor entre 0.1 ms e 0.4 ms a ser adicionado aos tempos, simulando, assim, atrasos e perdas de pacotes. Durante essa simulação do comando, o código invocará um *sleep* de um tempo próximo ao calculado para simular o tempo de espera que seria notado ao executar o comando em uma rede real. Por fim, no momento em que os pacotes retornam, é exibido, para cada um, o tamanho de cada um, o IP de origem, o IP de destino, o tempo tomado por cada um, dentre outras infomações que apareceriam na execução do comando tradicional mas que foram padronizadas para a simulação. Além disso, são exibidas estatísticas como o tempo mais rápido, o tempo mais lento, o tempo médio e a taxa de variação entre os tempos de cada um.

*2) Traceroute:* Simula o comando `traceroute` tradicional (descrito em II-B). Ao invés de enviar 4 pacotes como o ping, o traceroute envia 3 pacotes por vez, sendo que em cada vez o tempo de vida do pacote é extendido até chegar no vértice final. Esse comando possui a mesma funcionalidade que o ping no quesito de roteamento, geração de atraso, e pausa entre os cálculos de tempo para uma maior verossimilhança. Para cada iteração, ou seja, para cada vez que 3 pacotes novos são enviados, é mostrado o resultado anterior, contendo o tempo tomado por cada pacote e o IP que eles acessaram naquele momento.

*3) Navigateto:* : Permite que o usuário altere a posição atual do *host* para outro nó no grafo, permitindo que seja feita simulações de outro ponto. Após a atualização da posição, o usuário pode realizar os comandos como `ping` ou `tracerouter` para outros destinos a partir da nova localização, obtendo, assim, resultados diferentes.

*4) Changenetwork:* : Permite que o usuário altere a topologia da rede entre uma rede com cabeamento inteiramente coaxial e uma rede com enlaces apenas de fibra óptica. Ao realizar a troca de topologias, será tomado um pequeno tempo para reconstruir o grafo com novas informações e recalcular as tabelas de encaminhamento, executando novamente o algoritmo de Dijkstra. Para as velocidades de enlace, tem-se $2 \times 10^8$ para a rede com cabos coaxiais e $3 \times 10^8$ para a rede com cabos de fibra óptica. Por padrão, o código inicia na rede com enlaces de fibra óptica.

*5) Clear:* : Limpa o terminal.

### *D. Simulação dos comandos ping e traceroute em redes com enlaces diferentes*

Ao simular a rede, o grupo realizou quatro experimentos, dois na rede com enlace coaxial, e dois na rede com enlace defibra óptica. Os resultados obtidos são vistos a seguir:

```
root@172.16.10.1:~$ ping 172.16.6.2

PING 172.16.6.2 60 data bytes

60 bytes from 172.16.6.2: icmp_seq=1 ttl=64 time=28.51 ms
60 bytes from 172.16.6.2: icmp_seq=2 ttl=64 time=28.42 ms
60 bytes from 172.16.6.2: icmp_seq=3 ttl=64 time=28.45 ms
60 bytes from 172.16.6.2: icmp_seq=4 ttl=64 time=28.46 ms

4 packets transmitted, 4 received, 0% packet loss, time 113.84ms
rtt min/avg/max/mdev = 28.42/28.46/28.51/0.03 ms
```

Figura 2: Comando ping em rede com enlace coaxial

```
root@172.16.10.1:~$ ping 172.16.6.2

PING 172.16.6.2 60 data bytes

60 bytes from 172.16.6.2: icmp_seq=1 ttl=64 time=19.18 ms
60 bytes from 172.16.6.2: icmp_seq=2 ttl=64 time=19.15 ms
60 bytes from 172.16.6.2: icmp_seq=3 ttl=64 time=19.04 ms
60 bytes from 172.16.6.2: icmp_seq=4 ttl=64 time=19.00 ms

4 packets transmitted, 4 received, 0% packet loss, time 76.38ms
rtt min/avg/max/mdev = 19.0/19.09/19.18/0.08 ms
```

Figura 3: Comando ping em rede com enlace de fibra óptica

```
root@172.16.10.1:~$ traceroute 172.16.6.2

Tracing route to 172.16.6.2 over a maximum of 30 hops:

1     0.32 ms     0.47 ms    0.4 ms    172.16.10.30
2     7.45 ms     7.36 ms    7.29 ms    10.1.2.2
3     14.38 ms     14.3 ms    14.24 ms    10.1.1.2
4     21.5 ms     21.41 ms    21.47 ms    10.1.11.1
5     28.4 ms     28.26 ms    28.24 ms    10.1.4.1
6     28.45 ms     28.57 ms    28.56 ms    172.16.6.2

Trace complete.
```

Figura 4: Comando traceroute em rede com enlace coaxial

```
root@172.16.10.1:~$ traceroute 172.16.6.2

Tracing route to 172.16.6.2 over a maximum of 30 hops:

1     0.39 ms     0.41 ms    0.35 ms    172.16.10.30
2     5.02 ms     4.98 ms    5.11 ms    10.1.2.2
3     9.71 ms     9.61 ms    9.63 ms    10.1.1.2
4     14.4 ms     14.43 ms    14.29 ms    10.1.11.1
5     18.95 ms     19.06 ms    19.12 ms    10.1.4.1
6     18.96 ms     18.97 ms    19.01 ms    172.16.6.2

Trace complete.
```

Figura 5: Comando traceroute em rede com enlace de fibra óptica

Os comandos foram todos executados do mesmo IP de origem para o mesmo IP de destino (para melhor visualização das imagens, refira-se ao Apêndice A), de modo que o tempo médio tomado na execução dos comandos ping se aproxima dos tempos obtidos na execução do traceroute. Como mencionado na seção anterior (III-C), ambos os comandos geram um número aleatório entre 0.1 ms e 0.4 ms para permitir que haja a variação entre os tempos vista na realidade (e percebida na simulação desses exemplos). Comparando os resultados obtidos, é corroborado o que foi explicado tanto na seção teórica (II-B) quanto na seção da apresentação dos comandos (III-C) que os tempos obtidos para as redes com topologia de enlaces coaxiais (Figura 2 e Figura 4) são maiores, em média, que os tempos médios obtidos nas redes com topologia de enlaces de fibra óptica (Figura 3 e Figura 5).

## IV. Conclusão

Com tudo o que foi descrito neste artigo foi possível compreender melhor como é o funcionamento da camada de redes, cujos protocolos são de extrema importância na estruturação da rede. Por meio da simulação e do projeto de uma topologia de redes também foi obtido um conhecimento prático e teórico do funcionamento do protocolo IP e ICMP, de algoritmos de roteamento e diagnostico da rede. Para melhor compreensão da simulação apresentada, foi feito um vídeo explicativo que pode ser encontrado no Apêndice A.

## Referências

[1] POSTEL, J. RFC 760 - Internet Protocol. Disponível em: <https://datatracker.ietf.org/doc/html/rfc760>. Acesso em: 19 de fevereiro de 2025.

[2] POSTEL, J. RFC 792 - Internet Control Message Protocol. Disponível em: <https://datatracker.ietf.org/doc/html/rfc792>. Acesso em: 19 de fevereiro de 2025.

[3] MOSKOWITZ R.; KARRENBERG D.; REKHTER Y.; LEAR E.; DE GROOT G. RFC 1918 - Internet Control Message Protocol. Disponível em: <https://datatracker.ietf.org/doc/html/rfc1918>. Acesso em: 20 de fevereiro de 2025.

[4] KUROSE, J. F.; ROSS, K. W. Computer networking : a top-down approach. 8. ed. Hoboken: Pearson, 2020.

## Apêndice A
## Links do Projeto

Aqui constam os links para o vídeo explicativo que mostra a simulação em ação, do repositório contendo o código fonte para ele e das figuras utilizadas neste relatório. Para executar o código, basta acesar a pasta `src` e executar o código python `main.py`.

- Link para vídeo: https://youtu.be/cgH3eFd6NOU.
- Link para repositório e imagens: https://github.com/Schatten900/RC_Trabalho2.