Pontifícia Universidade Católica do Rio Grande do Sul

João Miguel Bonaldo Meier
Matheus Bueno de Oliveira

Trabalho Final

Laboratório de Redes de Computadores

Porto Alegre
2025

1. Introdução

Este relatório tem como objetivo apresentar os detalhes da implementação de uma ferramenta de monitoramento de tráfego de rede em tempo real, desenvolvida para capturar, interpretar e classificar pacotes de dados em uma rede local. A aplicação foi projetada para operar em um ambiente de servidor proxy, onde o tráfego dos clientes é analisado por meio de sockets raw. Durante o desenvolvimento do projeto, foram explorados os protocolos de rede, como IP, TCP, UDP, e ICMP, além de se estudar a estrutura e a análise dos pacotes de dados. O relatório descreve a arquitetura da solução implementada, a interface de monitoramento utilizada, os arquivos de log gerados e os resultados obtidos durante os testes realizados no ambiente de rede. O objetivo final foi proporcionar uma ferramenta eficiente para a análise e o registro do tráfego de rede, facilitando a visualização e o acompanhamento das informações de rede em tempo real.

2. Desenvolvimento

2.1. Implementação

A aplicação implementada é um monitor de tráfego em tempo real escrito em Python, composto por módulos responsáveis pela captura de pacotes, parsing das camadas IP/ICMP/TCP/UDP, identificação simples de aplicações (HTTP, DNS, DHCP, NTP), gravação imediata em arquivos CSV (logging_csv.py) e agregação de estatísticas por cliente e endpoint (stats.py).

- Capture.py: captura pacotes brutos da interface configurada (AF_PACKET) e separa L2/L3 (suporta TUN).
- Pasta de Parsing: funções que extraem campos essenciais de IPv4/IPv6, ICMP, TCP e UDP; fornece payload para identificação de aplicação.
- Módulo de Identificação de Aplicação: heurísticas simples (porta + assinatura) para detectar HTTP, DNS, DHCP e NTP.
- Módulo de Logging: grava eventos em internet.csv, transporte.csv e aplicacao.csv de forma imediata e thread-safe.
- Stats.py (Módulo de agregação de estatísticas): agrega por cliente (IP na sub-rede configurada) e por endpoint remoto: pacotes, bytes, portas e conexões TCP (via SYN).
- ui.py e main.py: loop de captura em thread, UI de terminal atualizando periodicamente e tratamento de SIGINT para encerramento limpo.

2.2. Execução do Programa

A execução do monitoramento de tráfego foi realizada em um ambiente Docker.

Durante a execução, foram gerados arquivos CSV com registros de pacotes de dados capturados. A seguir estão as amostras das primeiras linhas de cada arquivo:

logs/aplicacao.csv

- Registros de pacotes DNS e HTTP:
    - timestamp,protocolo,info
    - 2025-11-24T21:47:29.692516,DNS,DNS tid=17999 qr=0 opcode=0 rcode=0 qd=1 an=0
    - 2025-11-24T21:47:29.761106,HTTP,GET / HTTP/1.1 Host: example.com User-Agent: curl/8.14.1 Accept: */*

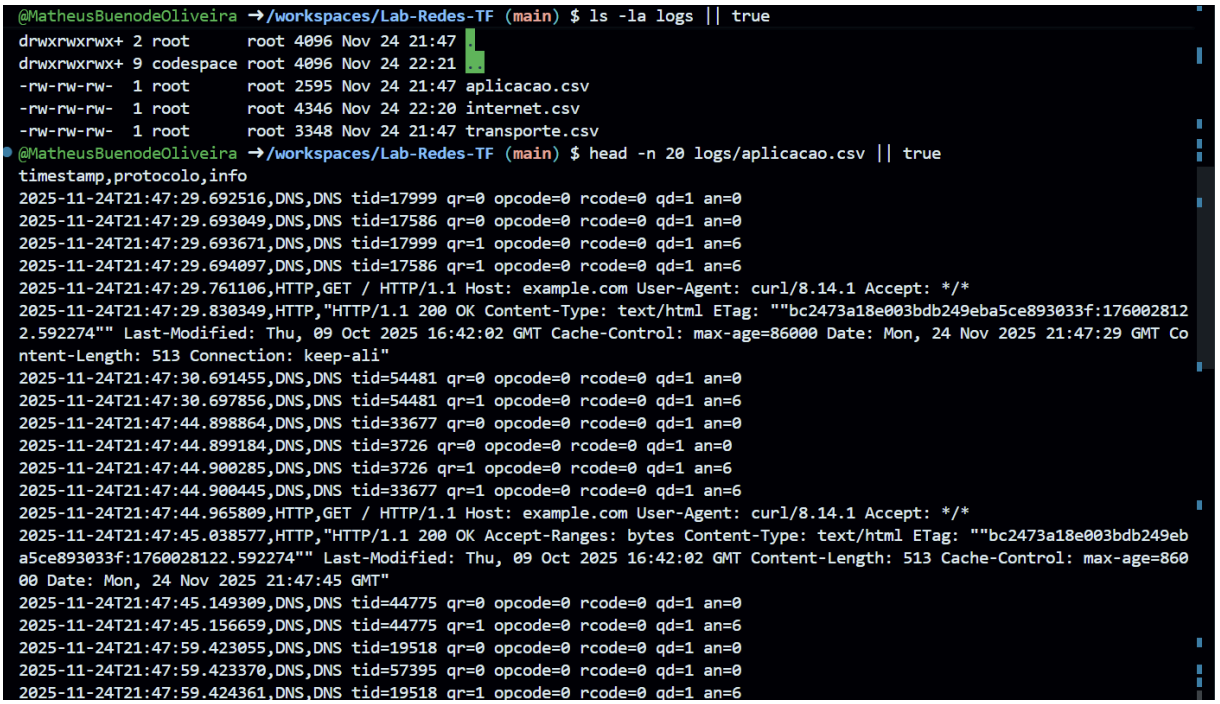
logs/internet.csv

- Captura de pacotes IPv4, com IPs de origem e destino:
    - timestamp,protocolo,src_ip,dst_ip,ip_proto,info,tamanho_bytes
    - 2025-11-24T21:47:29.692045,IPv4,172.17.0.3,168.63.129.16,17,,57
    - 2025-11-24T21:47:29.694739,IPv4,172.17.0.3,23.192.228.80,6,,60

logs/transporte.csv

- Captura de pacotes UDP e TCP, incluindo portas de origem e destino:
    - timestamp,protocolo,src_ip,src_port,dst_ip,dst_port,tamanho_bytes
    - 2025-11-24T21:47:29.692332,UDP,172.17.0.3,49689,168.63.129.16,53,57
    - 2025-11-24T21:47:29.694804,TCP,172.17.0.3,47220,23.192.228.80,80,60

2.3. Screenshots do Programa em Execução

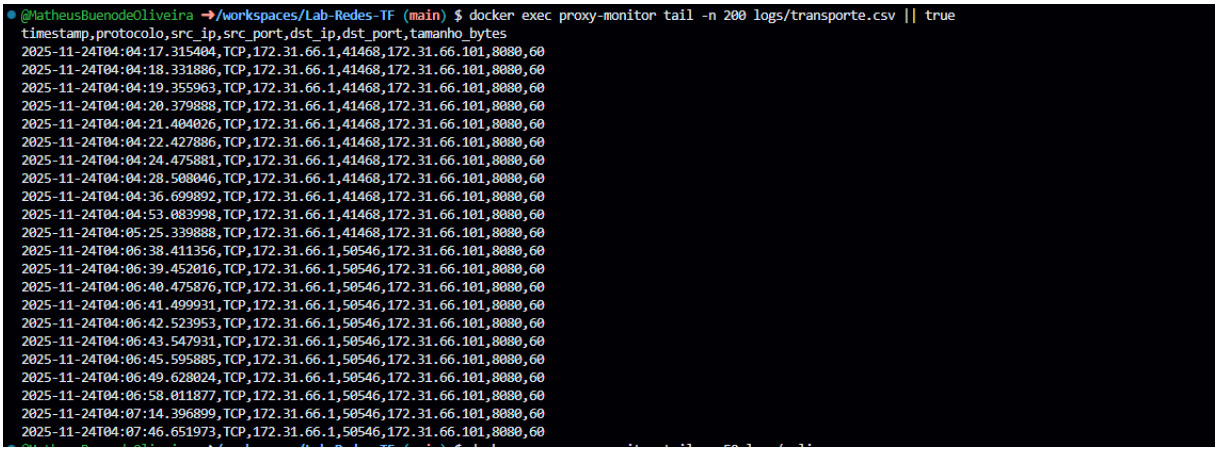


Na imagem, o terminal está exibindo a estrutura de arquivos do diretório logs/ e mostrando a execução de comandos para examinar os arquivos gerados durante o monitoramento de tráfego de rede.

1. Listagem de Arquivos: O comando ls -la logs lista os arquivos no diretório logs/, mostrando que os arquivos aplicacao.csv, internet.csv, e transporte.csv foram gerados às 21:47 de 24 de novembro de 2025.

2. Visualização das Primeiras Linhas do Arquivo aplicacao.csv: O comando head -n 20 logs/aplicacao.csv exibe as 20 primeiras linhas do arquivo aplicacao.csv, onde são registrados pacotes DNS e HTTP. É possível ver detalhes como o tipo de protocolo, informações de consulta e resposta (tid, qr, opcode) para DNS, além de requisições HTTP (GET) e suas respectivas respostas.

Esses comandos ajudam a monitorar e examinar os dados de tráfego coletados durante a execução, fornecendo detalhes sobre a comunicação de rede.



A imagem mostra a execução de um comando no terminal para monitorar o tráfego de rede em tempo real. O comando docker exec proxy-monitor tail -n 200 logs/transportes.csv exibe as últimas 200 linhas do arquivo transportes.csv no container proxy-monitor. O conteúdo das linhas exibidas no terminal mostra o registro de pacotes TCP, incluindo informações como timestamp, protocolo, endereços IP de origem e destino, portas de origem e destino, e o tamanho dos pacotes em bytes. Isso indica que o sistema está monitorando e registrando o tráfego TCP na rede.

3. Conclusão

O monitoramento de tráfego de rede foi implementado com sucesso utilizando um ambiente Docker, onde a ferramenta foi capaz de capturar e categorizar pacotes em tempo real. A aplicação processou pacotes de diversos protocolos, incluindo DNS, HTTP, TCP e UDP, gerando arquivos CSV (aplicacao.csv, internet.csv, e transporte.csv) com informações detalhadas sobre as consultas DNS, requisições HTTP e o tráfego de rede entre os containers, como endereços IP, portas de origem e destino, e tamanho dos pacotes.

A execução no terminal permitiu a monitorização eficaz dos arquivos gerados, confirmando que a coleta de dados de tráfego TCP estava funcionando corretamente.

O monitoramento foi realizado através da interface padrão AF_PACKET do Docker, já que a configuração do túnel (TUN) não pôde ser utilizada. Isso ocorreu porque o túnel disponibilizado não estava funcionando corretamente, impossibilitando sua implementação para a captura de pacotes através de uma rede virtual. Como alternativa, o tráfego entre os containers foi monitorado utilizando a rede bridge padrão do Docker, sem o encaminhamento de pacotes pelo túnel.

Portanto, embora o sistema tenha sido eficaz para capturar tráfego de rede nas interfaces padrão do Docker, a ausência do túnel limitou a coleta de dados mais complexos, como o tráfego entre diferentes redes externas ou isoladas, que seria possível através do uso do TUN. A solução se mostrou eficiente para capturar pacotes entre containers e redes locais.