Professors Review

Antonio Barros Coelho*, Davi Galvão Guerra†, Leonardo Mileo Moreira Krauss Guimarães‡

*Matrícula: 241038100
†Matrícula: 241038577
‡Matrícula: 241032690
Universidade de Brasília, Departamento de Ciência da Computação

***Abstract*—Este relatório detalha o processo de desenvolvimento e análise de uma aplicação web cliente-servidor, concebida como uma rede social para a avaliação de docentes. O projeto abrange desde a concepção da arquitetura, utilizando frameworks modernos que abstraem a comunicação em rede, até a análise do tráfego gerado. Através da ferramenta Wireshark, os pacotes trocados entre os clientes e o servidor foram inspecionados para verificar a correta implementação dos protocolos das camadas de aplicação e transporte, aprofundando o conhecimento prático sobre o tema.**

***Index Terms*—Arquitetura Cliente-Servidor, Protocolo TCP/IP, Protocolo HTTP, Wireshark**

I. Introdução

O estudo de redes de computadores frequentemente se baseia em modelos teóricos e abstratos. No entanto, a compreensão aprofundada dos mecanismos que governam a comunicação digital é mais eficazmente alcançada ao se observar os protocolos em sua operação real. Este projeto prático, inserido no contexto da disciplina de Redes de Computadores, busca solidificar os conhecimentos teóricos através do desenvolvimento de uma aplicação cliente-servidor funcional em um ambiente de rede real.

O objetivo foi projetar e implementar um novo serviço de rede: uma plataforma web que funciona como uma rede social para que alunos possam publicar avaliações e comentários sobre a atuação de professores. A execução desta aplicação em uma rede privada, composta por um servidor e múltiplos clientes, permitiu a análise detalhada da troca de mensagens e do comportamento dos protocolos subjacentes, cumprindo os requisitos de aprofundamento na camada de aplicação. Para ilustrar as principais funcionalidades da aplicação ”Professor Reviews”, as figuras no final do relatório apresentam as telas da interface de usuário 1.

Este relatório está estruturado da seguinte forma: a Seção II descreve os conceitos teóricos que fundamentam o projeto, como a arquitetura cliente-servidor, os frameworks utilizados e os protocolos de rede. A Seção III detalha o ambiente experimental, a metodologia de análise e os resultados obtidos com o uso do Wireshark. Por fim, a Seção IV apresentará as conclusões técnicas do trabalho, seguida pela Bibliografia consultada.

Seguem, em anexo, os links para o Pitch Técnico do projeto e para o Repositório do GitHub, onde está o código-fonte do projeto.

II. Fundamentação Teórica

Para o desenvolvimento e para a análise da aplicação, foram utilizados conceitos e tecnologias essenciais da área de redes de computadores e desenvolvimento web, que são descritos a seguir.

A. Arquitetura Cliente-Servidor

A aplicação foi desenvolvida sobre a arquitetura cliente-servidor, um modelo de estrutura de aplicação distribuída que distribui as tarefas entre os provedores de um recurso ou serviço, chamados de servidores, e os requisitantes do serviço, chamados de clientes. Neste projeto, o servidor é um processo responsável por aguardar e responder às requisições dos clientes, gerenciando a lógica de negócio e os dados da aplicação. Os clientes são os navegadores web utilizados pelos usuários para interagir com a plataforma, enviando solicitações de visualização ou submissão de dados.

B. Frameworks de Desenvolvimento e Abstração de Rede

Embora a comunicação em rede seja fundamentalmente estabelecida via sockets, o desenvolvimento de aplicações modernas frequentemente utiliza frameworks de alto nível que abstraem essa complexidade. Neste projeto, optou-se por essa abordagem para agilizar o desenvolvimento e focar na lógica da aplicação.

- **NestJS (Backend)**: Para o desenvolvimento do servidor, foi utilizado o framework Nest.js. Ele gerencia as conexões de rede e o protocolo HTTP, expondo a lógica da aplicação através de endpoints de uma API. Internamente, o Nest.js lida com as operações de baixo nível, como a criação de sockets e o tratamento de requisições, permitindo que o desenvolvedor se concentre em criar os serviços e controladores da aplicação.
- **Next.js (Frontend)**: No lado do cliente, o framework Next.js foi utilizado para construir a interface de usuário. Ele simplifica a comunicação com o backend ao facilitar o envio de requisições HTTP para a API do servidor, tratando as respostas para renderizar as páginas e componentes dinamicamente, sem a necessidade de gerenciar diretamente a comunicação em nível de transporte.

C. Protocolos das Camadas de Transporte e Aplicação

1) TCP (Transmission Control Protocol): Por baixo da abstração dos frameworks, o protocolo TCP foi o escolhido

para a camada de transporte. O TCP garante a entrega confiável e ordenada dos dados entre o cliente e o servidor, sendo o padrão para o tráfego HTTP [5]. Suas características de controle de fluxo, congestionamento e verificação de erros são cruciais para aplicações como a que foi desenvolvida, onde a integridade dos dados (posts, comentários) é essencial [1].

2) HTTP (Hypertext Transfer Protocol): Na camada de aplicação, o protocolo utilizado foi o HTTP [6]. Sendo o protocolo padrão para a World Wide Web, o HTTP define como as mensagens são formatadas e transmitidas e quais ações os servidores web e os navegadores devem tomar em resposta a vários comandos. A aplicação utiliza métodos HTTP como 'GET' e 'POST', para submeter novos feedbacks e comentários ao servidor [1].

D. Análise de Tráfego com Wireshark

O Wireshark é uma ferramenta de análise de protocolos de rede que permite capturar e inspecionar o tráfego que passa por uma interface de rede. Neste trabalho, o software foi fundamental para analisar o funcionamento prático dos protocolos e verificar se a comunicação entre os clientes e o servidor ocorria como o esperado. Com ele, foi possível filtrar e examinar os pacotes HTTP e TCP, analisar seus cabeçalhos, a carga útil (payload) e validar a correta implementação das funcionalidades da aplicação em nível de rede.

III. Ambiente Experimental e Análise de Resultados

Esta seção descreve o cenário utilizado para os testes e a análise dos resultados obtidos a partir da captura de pacotes, conforme solicitado no roteiro do laboratório.

A. Descrição do Cenário

O ambiente foi configurado para simular uma interação real entre múltiplos clientes e um servidor em uma rede privada.

- **Hardware**: Foram utilizados três computadores. Um atuou como servidor (executando o backend em Nest.js) com sistema operacional Windows 11. Os outros dois atuaram como clientes, sendo um com Windows 11 e o outro com uma distribuição Linux (executando o frontend em Next.js no navegador).
- **Software**: A aplicação é composta por um backend desenvolvido com o framework Nest.js e um frontend desenvolvido com o framework Next.js. O Wireshark, em sua versão 4.4.7, foi utilizado para a captura e análise de pacotes.
- **Topologia de Rede**: Os três computadores foram conectados à mesma rede local através de uma conexão Wi-Fi. Essa configuração permitiu que os clientes acessassem o servidor utilizando seu endereço IP privado na rede, estabelecendo um ambiente de teste controlado.

B. Análise de Resultados

A análise a seguir foi realizada com base na captura de pacotes com o Wireshark, utilizando o filtro '(ip.addr == 192.168.0.65 or ip.addr == 192.168.0.5) and tcp.port == 3001' para isolar a comunicação entre os clientes e o servidor na porta da aplicação.

1) Identificação do tipo e versão de software: A identificação do software do cliente (navegador) e do servidor pode ser feita inspecionando os cabeçalhos das mensagens HTTP. O cabeçalho 'User-Agent' em uma requisição HTTP revela o navegador e o sistema operacional do cliente. De forma análoga, o cabeçalho 'Server' em uma resposta HTTP pode indicar o software do servidor web ou o framework utilizado. Embora as capturas não expandam esses cabeçalhos específicos, o tráfego HTTP e WebSocket observado é consistente com a comunicação entre um navegador moderno e um servidor backend baseado em Node.js, como o Nest.js.

2) Endereços IP do Cliente e Servidor: Os endereços IP foram identificados claramente durante a captura de pacotes. Conforme a Figura 10, que mostra a comunicação entre o cliente Windows e o servidor, os endereços são:

- **Endereço IP do Servidor**: '192.168.0.218'
- **Endereço IP do Cliente (Windows)**: '192.168.0.5'
- **Endereço IP do Cliente (Linux)**: '192.168.0.65'

Esses valores são consistentes em todas as capturas, aparecendo nas colunas de origem ('Source') e de destino ('Destination') do Wireshark.

3) Análise da Carga Útil dos Pacotes: A análise da carga útil (payload) dos pacotes capturados confirmou o comportamento esperado da aplicação. Os dados trocados entre cliente e servidor estão em formato JSON, o que é compatível com a arquitetura REST adotada.

Durante a submissão de uma nova avaliação 10, foram observadas requisições HTTP do tipo POST contendo objetos JSON com os dados do feedback. No painel de dados brutos do Wireshark, o campo "Data" apresenta a sequência de bytes iniciando com 7b 22 70 6f..., equivalente à cadeia "po..., confirmando o início de um objeto JSON enviado pelo cliente.

De forma semelhante, ao carregar o feed de avaliações 8, o cliente realiza uma requisição HTTP GET, e o servidor responde com um payload contendo um array de objetos JSON. A presença de tráfego do tipo WebSocket Text também foi registrada, indicando a possível atualização dinâmica de dados em tempo real pela aplicação.

Essas informações foram localizadas na aba inferior do Wireshark, especificamente na seção correspondente ao campo "Data" da camada de transporte TCP.

4) Analogia do Encapsulamento: A inspeção detalhada dos pacotes no Wireshark permite observar na prática o conceito de encapsulamento, fundamental na arquitetura em camadas da pilha TCP/IP.

Cada pacote registrado apresenta uma estrutura composta por múltiplos níveis:

- **Camada de Aplicação:** Os dados da aplicação aparecem como conteúdo das requisições HTTP ou mensagens WebSocket, contendo informações como avaliações ou comentários.
- **Camada de Transporte:** Esta camada adiciona o cabeçalho TCP, que inclui número de sequência, número

de reconhecimento, portas de origem e destino e flags de controle. O resultado é um segmento TCP.

- **Camada de Rede:** Envolve o segmento TCP com um cabeçalho IP, que especifica os endereços IP de origem e destino, formando um pacote IP.
- **Camada de Enlace:** Por fim, o pacote IP é encapsulado com um cabeçalho Ethernet, contendo os endereços MAC de origem e destino. O resultado é um quadro completo transmitido pela interface de rede.

Todas essas camadas são visíveis na aba de detalhes do Wireshark, por meio dos cabeçalhos Ethernet, IPv4, TCP e dos dados da aplicação, demonstrando de forma clara e estruturada a aplicação do modelo de encapsulamento estudado em sala de aula.

IV. Conclusões

A realização deste projeto permitiu a aplicação prática dos conceitos fundamentais de redes de computadores, consolidando o conhecimento teórico sobre a arquitetura cliente-servidor e a pilha de protocolos TCP/IP. O desenvolvimento da aplicação com frameworks modernos demonstrou a abstração das camadas inferiores, enquanto a análise de tráfego com o Wireshark possibilitou a visualização concreta do fluxo de comunicação. Foi possível verificar a troca de mensagens HTTP e os segmentos TCP, confirmando o encapsulamento dos dados e o funcionamento esperado dos protocolos. Conclui-se, por meio do trabalho, que a união do desenvolvimento prático com a análise de rede é uma ferramenta poderosa para a compreensão aprofundada dos sistemas de comunicação.

V. Bibliografia

References

[1] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*, 8th ed. Boston: Addison-Wesley, 2020.

[2] NestJS Documentation, "First steps." [Online]. Available: https://docs.nestjs.com/. Accessed: Jun. 07, 2025.

[3] Vercel, "Next.js by Vercel." [Online]. Available: https://nextjs.org/docs. Accessed: Jun. 07, 2025.

[4] Wireshark Foundation, "Wireshark User's Guide." [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked/. Accessed: Jun. 07, 2025.

[5] J. Postel, "Transmission Control Protocol," RFC 793, STD 7, IETF, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/rfc/rfc793.

[6] R. Fielding, et al., "HTTP/1.1," RFC 9112, STD 99, IETF, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9112.

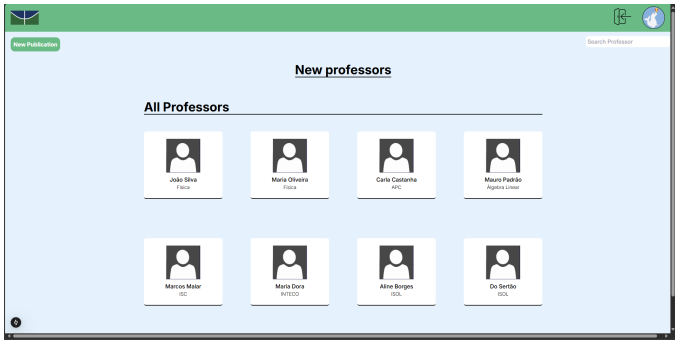

Fig. 1. Tela inicial da aplicação

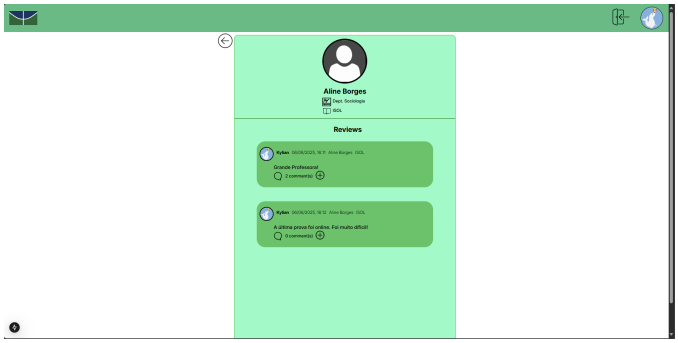

Fig. 2. Página de perfil do docente

Fig. 3. Página de perfil do usuário

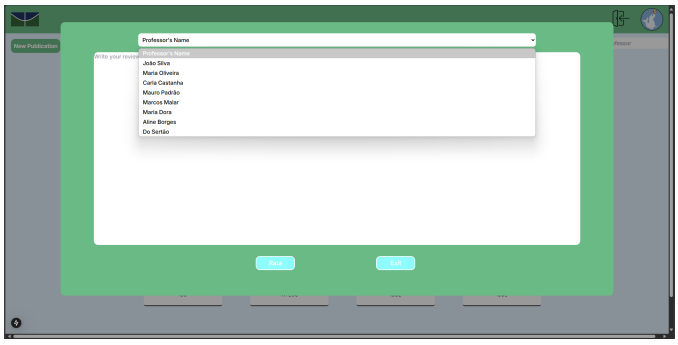

Fig. 4. Modal de avaliação

Fig. 5. Página da avaliação e de seus comentários

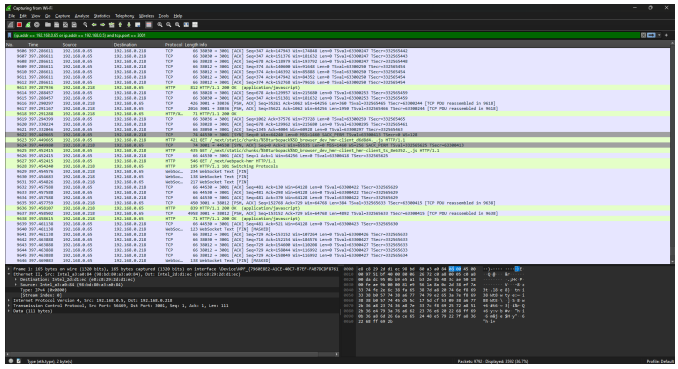

Fig. 6. Criação de conta

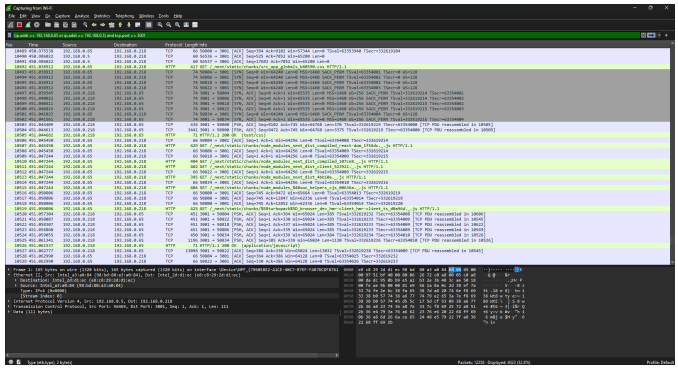

Fig. 7. Entrada na nova conta

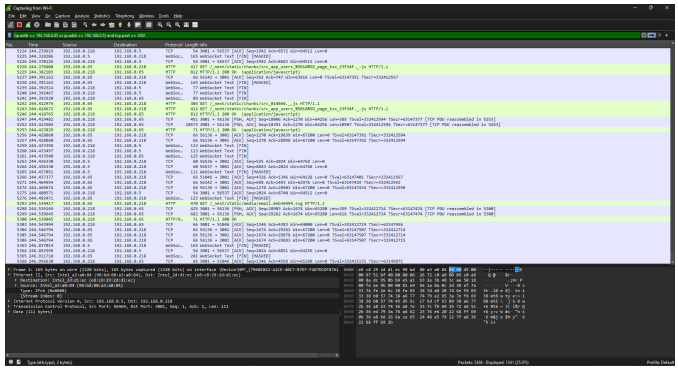

Fig. 8. Troca de página

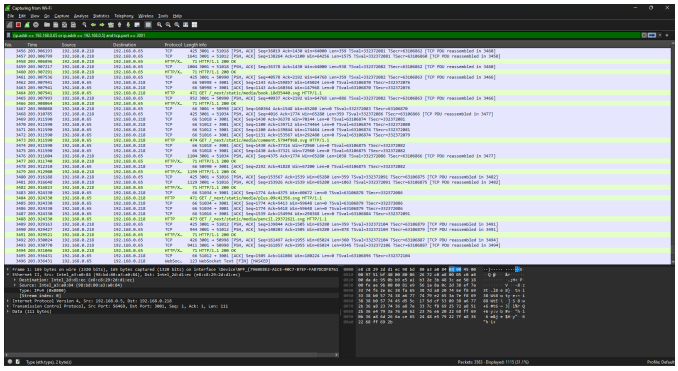

Fig. 9. Comentário

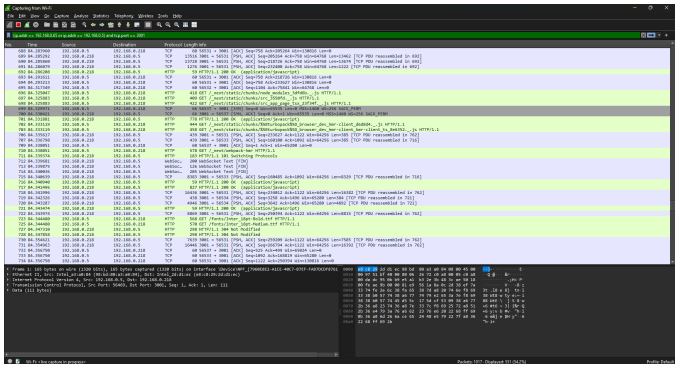

Fig. 10. Avaliação