UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GUILHERME BOIA DANTAS DE ALBUQUERQUE
IVAN GOMES DE ALCANTARA JUNIOR
JOÃO MATHEUS PINTO VILLARIM COUTINHO DE ALMEIDA
JULIO HSU

PROTOCOLO DE TRANSFERÊNCIA DE ARQUIVOS PERSONALIZADO (FTCP)

RELATÓRIO DO PROJETO

CAMPINA GRANDE
2025

Relatório de Análise de Tráfego com Wireshark (RELATORIO EXTRA)

Objetivo
O objetivo deste projeto é realizar a captura e análise do tráfego de rede gerado por um protocolo FTCP, bem como identificar e estudar pacotes dos protocolos DHCP e DNS usando a ferramenta Wireshark.

Ferramentas Utilizadas
- **Wireshark** (interface gráfica) para captura e análise dos pacotes
- **Sistema Operacional**: Windows
- **Cliente e Servidor FTCP** rodando na mesma máquina

Etapas Realizadas

1. Instalação do Wireshark
- Download realizado pelo site oficial: https://www.wireshark.org/
- Instalação com Npcap incluído para possibilitar a captura de pacotes

2. Execução do Cenário FTCP
- Iniciado o servidor FTCP local
- Cliente FTCP executado para solicitar o arquivo `a.txt` via TCP
- Trânsito de pacotes registrado usando o Wireshark
- Interface utilizada para captura: **Loopback** (por estarem na mesma máquina)

3. Captura do Tráfego
- A captura foi feita utilizando a interface gráfica do Wireshark
- A gravação foi salva no formato `.pcapng`
- Durante a execução do cliente, a transferência foi observada na interface

5. Análise DNS
- Filtro utilizado: `dns`
- Observadas requisições DNS ("Standard Query") e respostas ("Standard Query Response")
- Exemplos analisados:
  - Identificação de pacotes UDP na porta 53

6. Análise do Tráfego FTCP com TCP

6.1 Handshake TCP
- **Descrição**: Mostra o início da conexão TCP entre cliente e servidor FTCP.
- **Screenshot**: `imagens/handshake_tcp.png`
- **Pacotes esperados**:
  - SYN (cliente → servidor)
  - SYN-ACK (servidor → cliente)
  - ACK (cliente → servidor)

6.2 Comando `get`
- **Descrição**: Pacote onde o cliente envia o comando `get a.txt` para o servidor.
- **Screenshot**: `imagens/comando_get.png`

6.3 Dados do Arquivo
- **Descrição**: Vários pacotes contendo os dados do arquivo `a.txt` enviados pelo servidor ao cliente.
- **Screenshot**: `imagens/dados_arquivo.png`

6.4 Comando `ftcp_ack`
- **Descrição**: Pacote onde o cliente envia um ACK confirmando o recebimento dos dados.
- **Screenshot**: `imagens/ftcp_ack.png`
- **Explicação**:
  - O protocolo TCP garante entrega confiável através dos **números de sequência (Sequence Number)** e **números de confirmação (Acknowledgment Number)**.
  - Cada byte de dados transmitido tem um número de sequência.
  - O ACK enviado pelo cliente informa qual o próximo byte esperado.
  - Isso garante que, caso um pacote seja perdido, ele será retransmitido, assegurando a entrega completa e correta dos dados.

Capturas de Tela
As imagens estão localizadas na pasta `imagens/` e ilustram:
- Pacotes DHCP e seus detalhes
- Consultas DNS e respostas
- Tráfego FTCP (cliente e servidor)

Conclusão
A análise permitiu visualizar claramente os pacotes trocados no protocolo FTCP customizado, assim como o funcionamento dos protocolos DHCP e DNS na rede. O uso do Wireshark se mostrou eficiente para inspeção e compreensão do comportamento da rede.

Autor: Ivan Gomes De Alcantara Junior e Julio Hsu

Data: 09/10/2025

Observação: O arquivo `.pcapng` da captura está disponível neste repositório.

1. Introdução

1.1. Objetivo

Este relatório tem como objetivo apresentar a análise dos protocolos de comunicação utilizados na implementação de um protocolo de transferência de arquivos, com ênfase nas camadas de transporte, rede e aplicação do modelo *Open Systems Interconnection* (OSI). Através da captura e inspeção do tráfego de rede por meio da ferramenta *WIreshark*, buscamos avaliar e compreender o funcionamento dos protocolos envolvidos.

1.2. Motivação

No contexto da transferência de arquivos, é crucial entender a diferença entre os protocolos da camada de transporte *User Datagram Protocol* (UDP) e *Transmission Control Protocol* (TCP), uma vez que a escolha entre estes dois protocolos afeta diretamente a confiabilidade, “velocidade” (i.e., taxa de transferência de dados) e eficiência da comunicação.

O TCP, comumente utilizado, tem como vantagens em relação ao UDP: maior confiabilidade, garantia de ordem correta dos dados, e reenvio automático para correção de erros; o UDP, por outro lado, apresenta como vantagens em relação ao TCP maior “velocidade” devido a menor sobrecarga.

1.3. Visão geral do documento

Este documento está organizado da seguinte forma:

- na seção 1, são apresentados o objetivo e a motivação deste documento, bem como o sumário de suas seções;
- na seção 2, são descritos os conteúdos dos arquivos de configuração e testes;
- na seção 3, são demonstrados os dados obtidos a partir da utilização da aplicação desenvolvida, bem como a análise dos aspectos técnicos referentes aos conceitos observados;
- e na seção 4, são relatadas observações sobre o processo de desenvolvimento deste projeto.

2. Descrição da implementação

2.1. Arquivo de configuração

Em config.ini estão definidos os seguintes parâmetros de configuração:

```
[SERVER]
udp_port = 5005
tcp_port_range_start = 6000
tcp_port_range_end = 6005
file_a = a.txt
file_b = b.txt
```

Dessa forma, temos que: a porta que o servidor usará para comunicação via UDP é 5005; o intervalo de portas TCP que o servidor pode usar é [6000, 6005]; e os arquivos “file_a” e “file_b” estão nomeados como “a.txt” e “b.txt”, neste mesmo diretório.

2.2. Arquivos de teste

O conteúdo dos arquivos “a.txt” e “b.txt” podem ser acessados através dos *hiperlinks*.

3. Análise do protocolo

Uma vez que executamos ambos o servidor e o cliente simultaneamente, podemos então utilizar o *Wireshark* para realizar transferências dos arquivos a.txt e b.txt, usando TCP, ou ainda, realizar uma solicitação inválida para a avaliação da mensagem de erro.

Todas as imagens anexadas nesta seção podem também ser acessadas através deste link, para melhor visualização.

3.1. Negociação inicial via UDP

Demonstraremos a troca de mensagens UDP entre cliente e servidor para negociar a porta e o protocolo de transferência na nossa aplicação.

3.1.1. Pacote de requisição (cliente -> servidor)

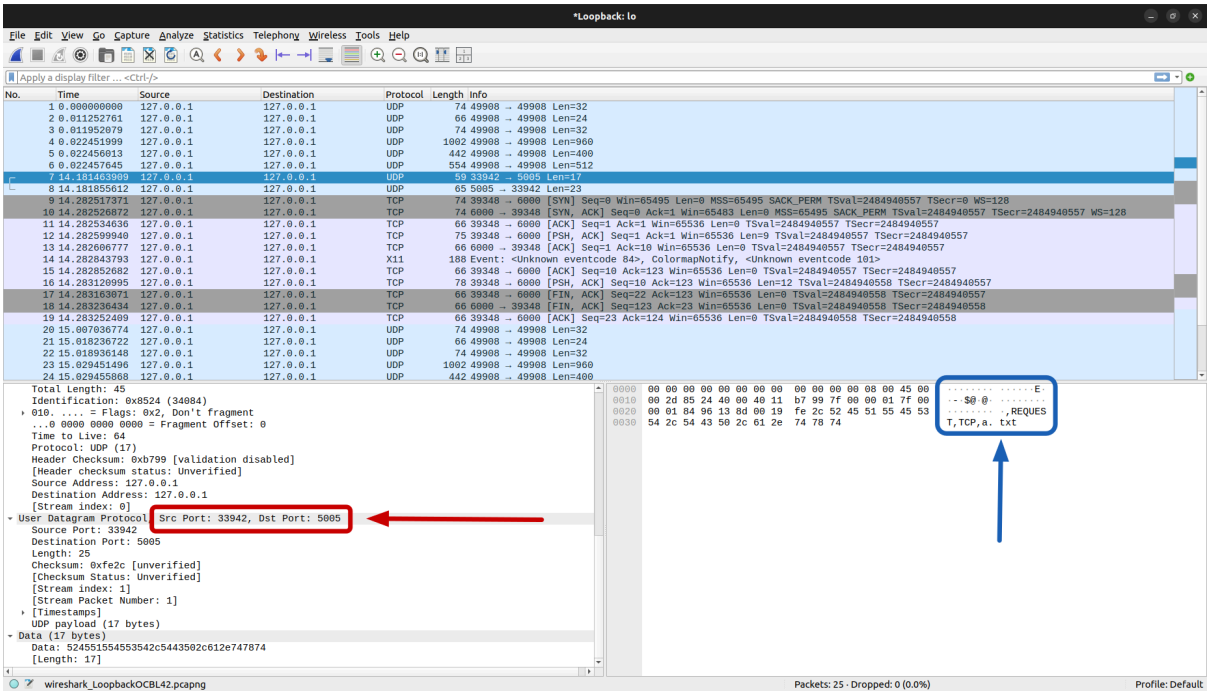

Figura 1: Captura de tela do pacote UDP enviado pelo cliente.

Na figura 1, estão destacadas em vermelho as portas de origem (porta efêmera 33942) e destino (5005), e em azul o *payload* do datagrama UDP com a codificação esperada (“REQUEST,TCP,a.txt”).

3.1.2. Pacote de resposta (servidor -> cliente)

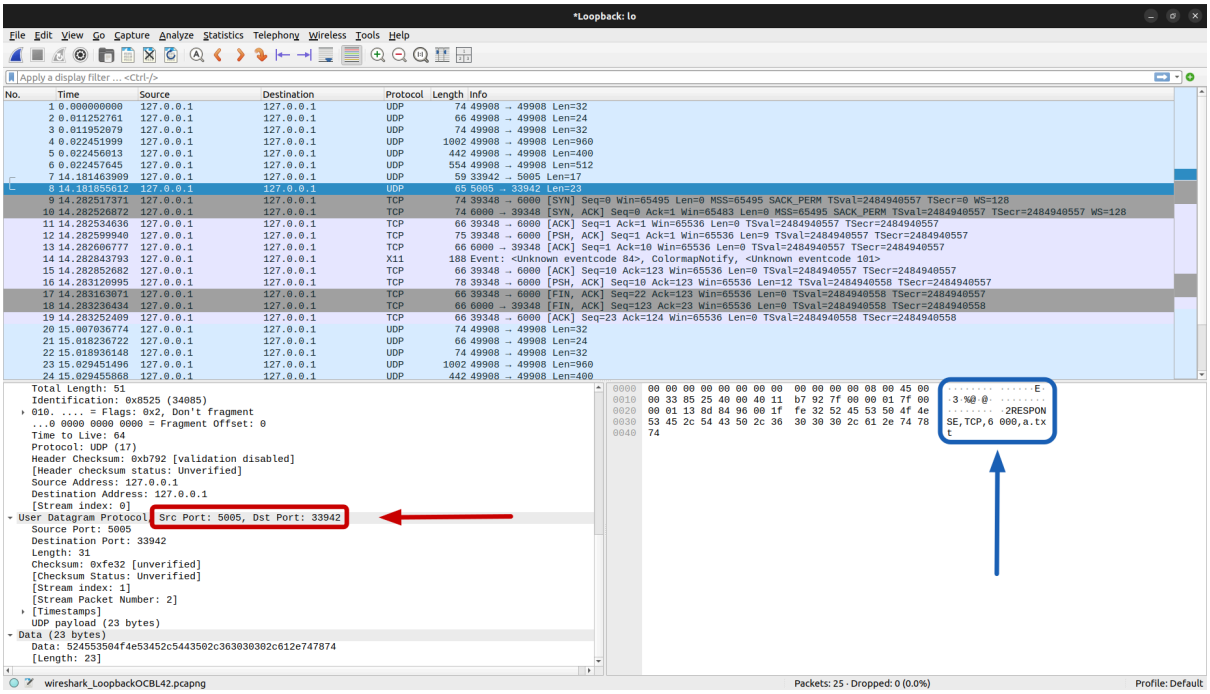

Figura 2: Captura de tela do pacote UDP de resposta do servidor

Na figura 2, estão destacadas em vermelho as portas de origem (5005) e destino (porta efêmera 33942), e em azul o *payload* do datagrama UDP com a codificação esperada (“RESPONSE,TCP,6000,a.txt”).

3.1.3. Pacote de erro (servidor -> cliente)

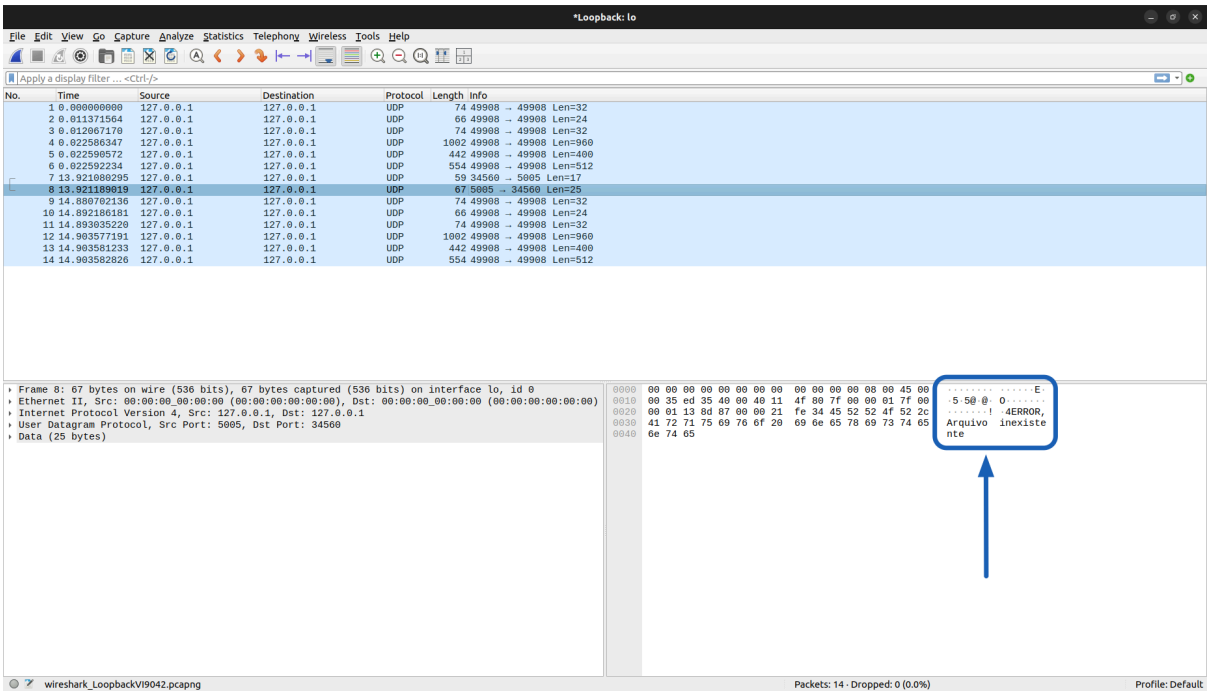

Figura 3: Captura de tela do pacote UDP de tentativa inválida

Na figura 3, está destacado o *payload* do datagrama UDP com a mensagem “ERROR,arquivo inexistente”, resultado esperado ao passar como parâmetro para o cliente da aplicação o nome de um arquivo não existente em seu diretório.

3.2. Transferência de dados via TCP

Demonstraremos o estabelecimento da conexão TCP, a troca de comandos da aplicação FTCP e a transferência confiável do arquivo no nosso projeto.

3.2.1. Estabelecimento da conexão (*three-way handshake*)

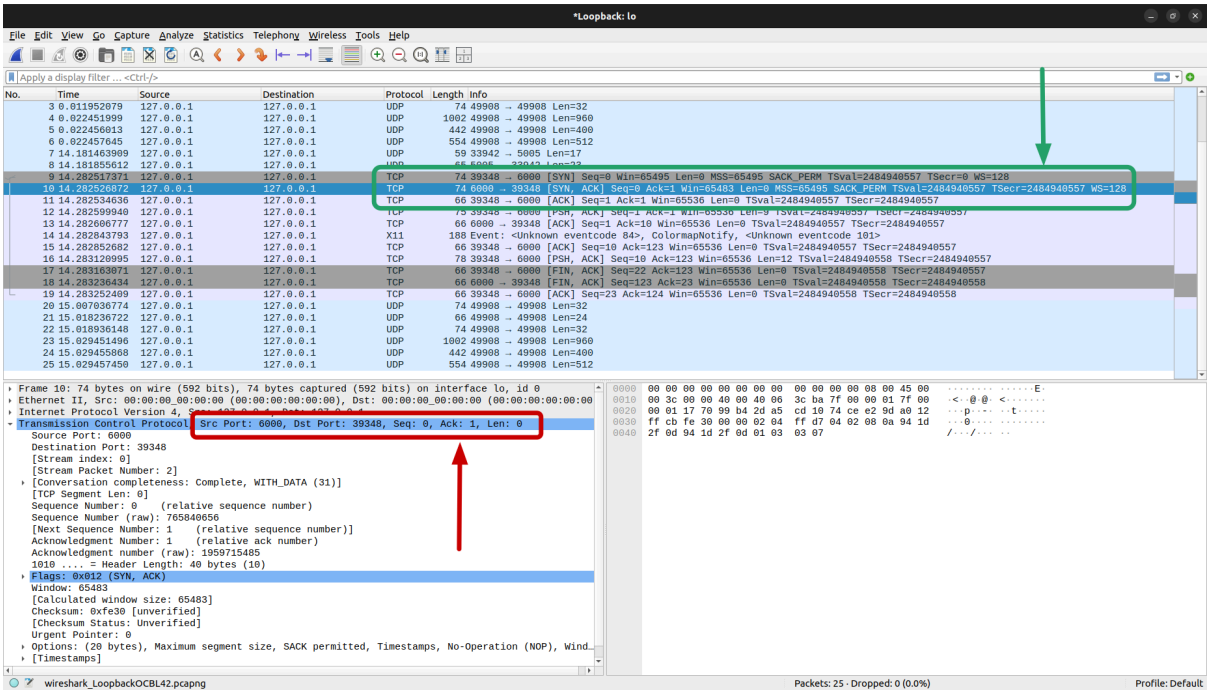

Figura 4: Captura de tela dos pacotes SYN, SYN-ACK e ACK

Na figura 4, estão destacadas em verde as *flags* “SYN” e “ACK”, e em vermelho os números de sequência e confirmação iniciais, bem como as portas de origem (6000) e destino (porta efêmera 39348). Estas *flags* são utilizadas para o procedimento de *three-way handshake*, que verifica bidirecionalmente se ambos os lados estão prontos para a troca de dados, assim evitando a perda de pacotes.

3.2.2. Solicitação do arquivo (cliente -> servidor)

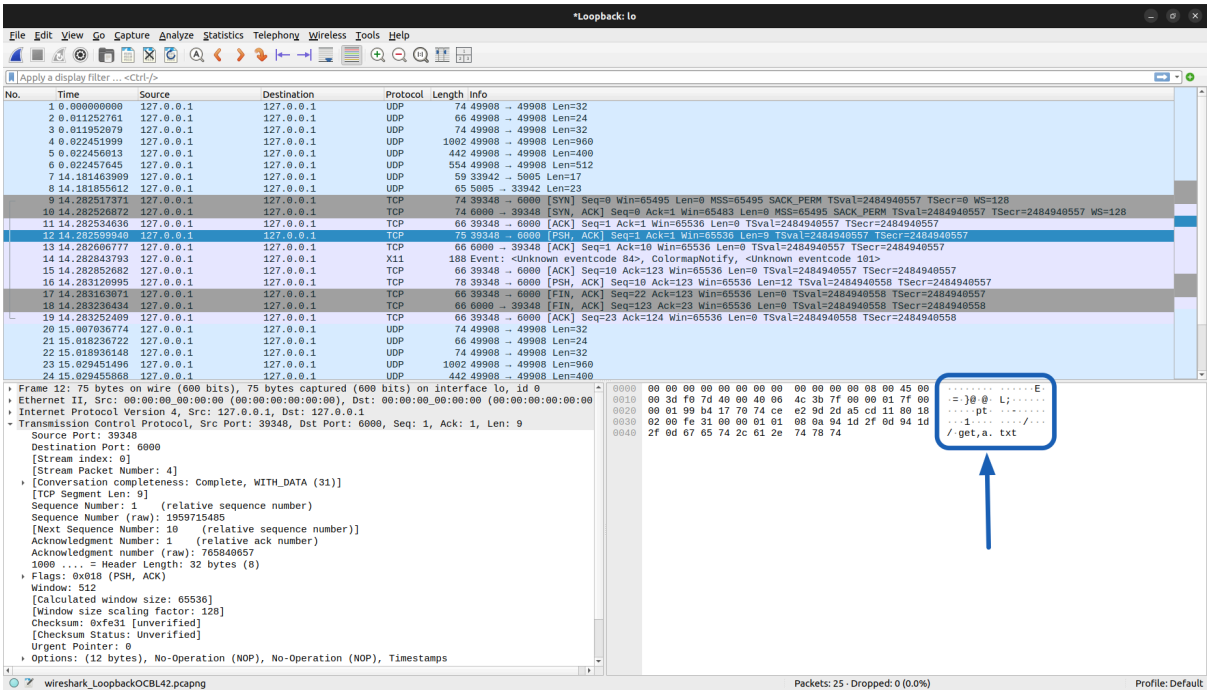

Figura 5: captura de tela do pacote TCP contendo o comando "*get*"

Na figura 5, está destacado em azul o *payload* do pacote TCP contendo o comando "get,a.txt", que faz parte do protocolo FTCP criado para a aplicação.

3.2.3. Transferência dos dados do arquivo (servidor -> cliente)

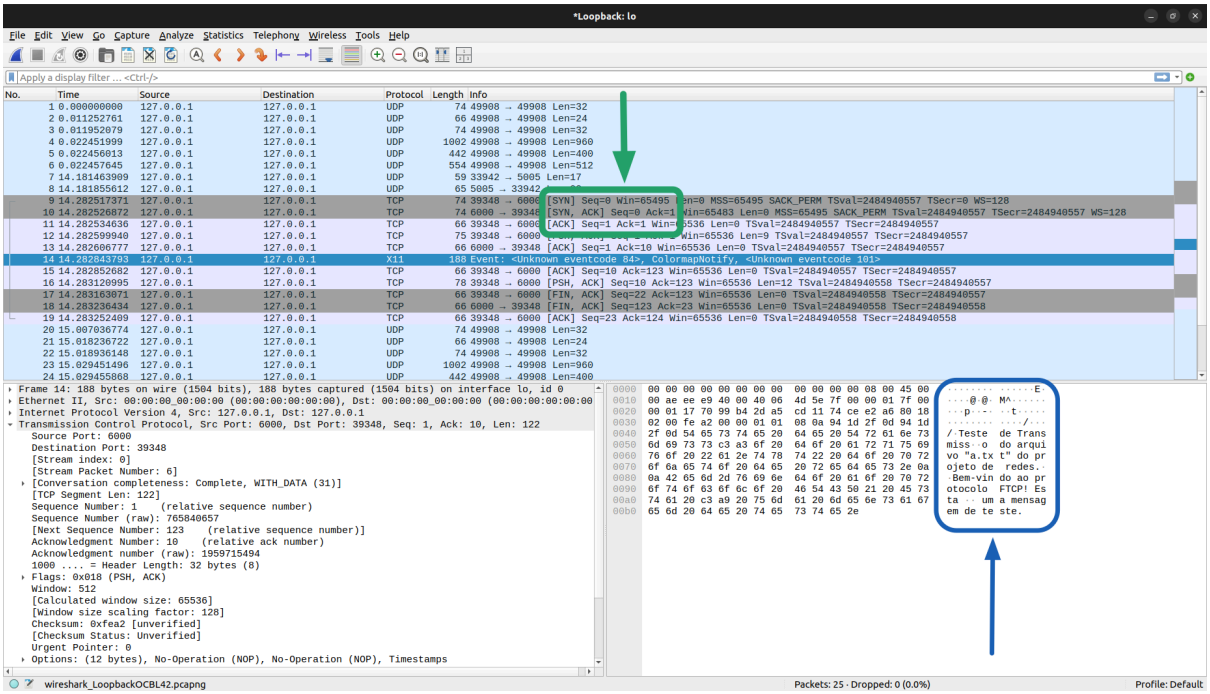

Figura 6: captura de tela do pacote TCP contendo parte do conteúdo do arquivo a.txt

Na figura 6, está destacado em azul o *payload* do pacote TCP que contém parte do arquivo a.txt, como esperado, e em verde os números de sequência e de confirmação (ACK). Nota-se que os números de sequência aumentam gradualmente, e os de confirmação correspondem aos enviados pelo cliente, garantindo a entrega ordenada e

confiável dos dados, uma vez que estes números são utilizados para detectar perdas de bytes e retransmitir pacotes, mas neste caso indicam sucesso.

3.2.4. Confirmação de recebimento (cliente -> servidor)

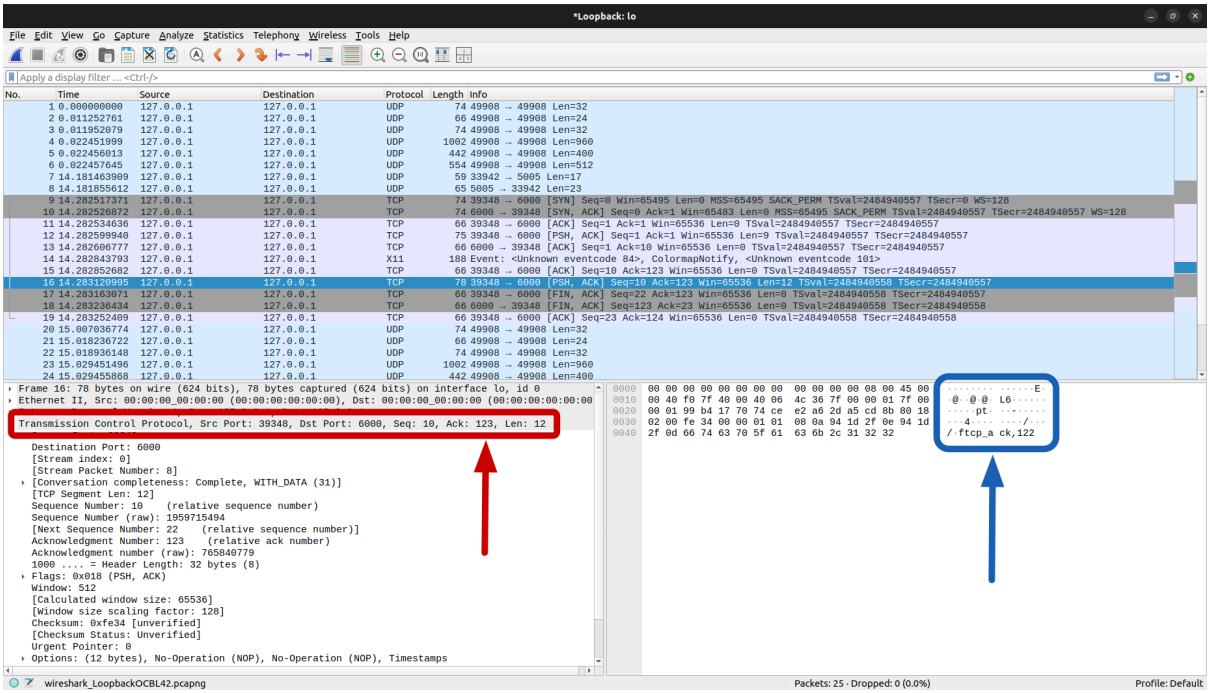

Figura 7: captura de tela do pacote TCP contendo o comando “ftcp_ack”

Na figura 7, está destacado em azul o *payload* do pacote TCP contendo o comando “ftcp_ack”, com o conteúdo esperado (“ftcp_ack, 122). Nota-se que “122” corresponde ao número de bytes de a.txt e que este comando é também transportado via TCP.

3.2.5. Encerramento da conexão

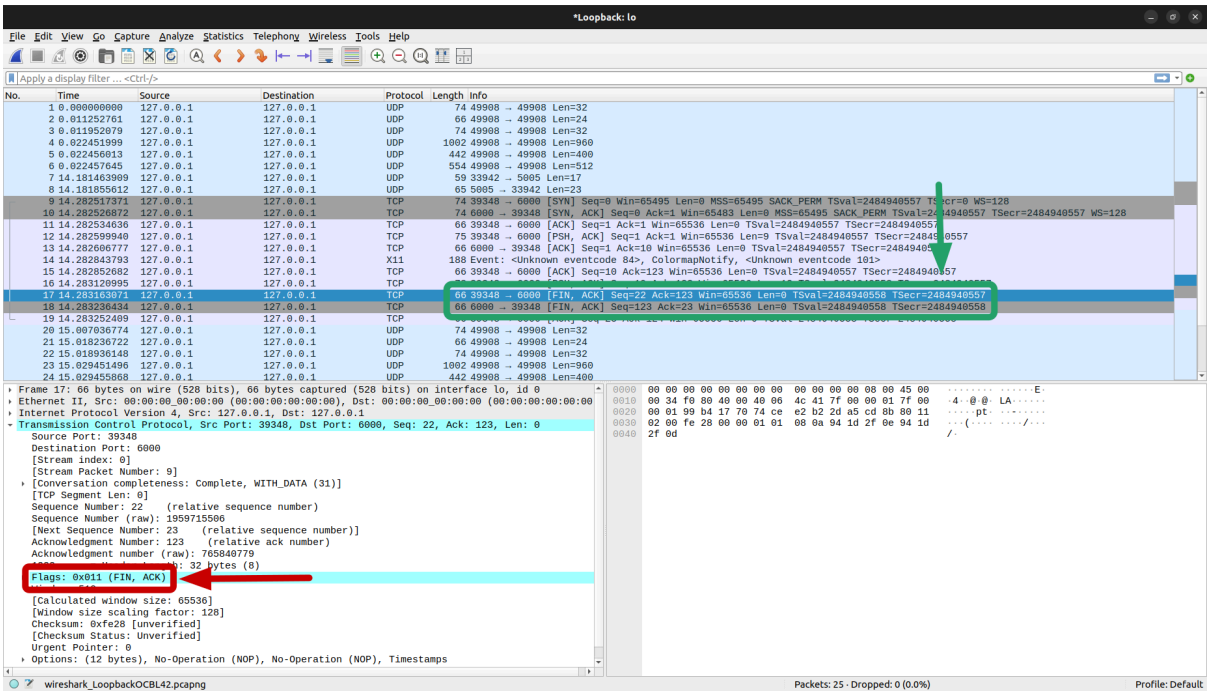

Figura 8: captura de tela de pacotes com flags “FIN” e “ACK”

Na figura 8, estão destacados em verde os pacotes com *flags* “FIN” e “ACK”. Esses pacotes são utilizados para encerrar a conexão TCP, através de um processo chamado *four-way handshake*. Normalmente, o *four-way handshake* ocorre em 4 turnos, no qual um dos lados envia FIN para sinalizar que não quer mais enviar dados, e outro envia ACK para confirmar o recebimento do FIN, e vice-versa.

4. Discussão

4.1. Desafios encontrados

Um dos desafios enfrentados pela equipe para a implementação deste projeto foi a difícil compreensão sobre portas com as quais serão iniciadas o *three-way handshake* para estabelecer a porta do TCP, e como os arquivos são enviados em segmentos.

4.2. Possíveis melhorias

Nota-se que na implementação desenvolvida para este projeto, a transferência de arquivos ocorre em texto claro (i.e., sem criptografia), permitindo com que um possível interceptador da rede acesse ou altere os dados em trânsito, portanto, uma possível melhoria para esta aplicação seria a adição de mecanismos de criptografia, para garantir a confidencialidade e integridade dos dados transmitidos.