

Relatório Técnico: Protocolo SchaiLay

Curso: Engenharia de Telecomunicações

Disciplina: Projetos de Protocolos

Alunas: Layssa Pacheco e Schaiana Sonaglio

Introdução

Este trabalho consiste no desenvolvimento do protocolo SchaiLay, baseado no protocolo TFTP, com o objetivo de viabilizar a transferência de arquivos simplificada, sem uso de autenticação e de maneira confiável, através de datagramas UDP.

Para isso, especificou-se as características do protocolo SchaiLay construído e a implementação foi feita em formato de API, utilizou-se para o protótipo as aplicações servidora e cliente, sendo ambas para realização de testes.

Em síntese, este relatório contém as especificações do protocolo SchaiLay, a documentação da API e o manual de usuário.

Especificação do Protocolo SchaiLay

Nesta parte do documento são descritos os comportamentos do Protocolo SchaiLay, através da visão cliente-servidor. Desta maneira, deve-se entender que, ao citar cliente e servidor, não é especificado nenhum cliente ou servidor, mas apenas são descritas as ações esperadas por cada um dos lados da comunicação.

O Protocolo SchaiLay tem por finalidade e serviço a transferência de arquivos através do modelo cliente-servidor com conexão única do tipo 1:1, sem autenticação, com garantia de entrega e tratamento de erro, em um ambiente que usa datagramas UDP. A codificação usada é do tipo binária.

A única comunicação estruturada é do tipo escrita, no qual o cliente solicita ao servidor a escrita de um arquivo e o envia sequencialmente dividido em pacotes através do Protocolo SchaiLay. Em contrapartida, quando o servidor recebe a solicitação de escrita ou os pacotes de dados, o Protocolo SchaiLay responde com uma mensagem de ACK específico confirmando o recebimento dos pacotes.

O pacote de solicitação de escrita é composto de:

- **Código de operação:** identifica que tipo de pacote é esse que, no caso, é o primeiro pacote e corresponde à solicitação de escrita do cliente no servidor. Corresponde a 2 bytes representando o número 2, o qual identifica essa operação.

- Nome do arquivo: string correspondente ao nome do arquivo que será enviado.
- Fim do nome do arquivo: identifica que a string do nome do arquivo acabou. Corresponde a 1 byte representado pelo número zero.
- Modo: Diz qual é o formato do arquivo que será enviado. Exemplo: octeto ou *netascii*. No caso deste projeto, foi utilizado o modo octeto.
- Fim do modo: identifica que o modo acabou. Corresponde a 1 byte representado pelo número zero.

A exemplificação do pacote de solicitação de escrita é apresentado a seguir:

1 byte	1 byte	string	1 byte	1 byte

0	2	packet	0	0

O pacote de dados é mais simples do que o de solicitação de escrita, sendo composto de:

- Código de operação: identifica que tipo de pacote é esse, que no caso é um pacote de dados, portanto, corresponde a 2 bytes representando o número 3, o qual identifica essa operação.
- Identificação do pacote: aponta qual é o pacote enviado na sequência de envio do arquivo, portanto, corresponde a 2 bytes representando o número de identificação do pacote. A quantidade de pacotes enviados, que é N, é definido pela divisão do tamanho do arquivo em 512 bytes, que é o tamanho máximo de dados que cada pacote comporta.
- Dados: corresponde aos dados do arquivo que será enviado em formato binário. Esta parte do pacote tem o limite de 512 bytes.
- Flag último pacote: identifica se esse é ou não o último pacote. Corresponde a 1 byte representando:
 - O número zero, caso não seja o último pacote.
 - O número um, caso seja o último pacote.

Esta sinalização de último pacote foi feita utilizando o primeiro byte do código de operação, uma vez que ele é sempre zero, devido ao número de operações. Quando é o último pacote, se altera este byte zero para um, a fim de sinalizar o fim do arquivo.

A exemplificação do pacote de dados é apresentado a seguir:

1 byte	1 byte	1 byte	1 byte	n bytes

0	3	0	5	dados

Quando o recebimento de um pacote esperado pelo servidor demora mais do que 10 segundos (10 segundos é padrão, porém pode ser configurado diferente), que foi o tempo especificado de timeout, a última mensagem de ACK é reenviada. O mesmo acontece quando uma mensagem de ACK esperada pelo cliente não é recebida no tempo especificado de timeout, que também é 10 segundos (padrão, porém configurável), o último pacote é reenviado. Caso aconteça, em ambas as situações, cinco reenvios seguidos (padrão, porém configurável), entende-se que houve algum problema com a conexão e, portanto, ela é fechada.

A mensagem de ACK é composta de:

- Código de operação: identifica que tipo de mensagem é esta que, no caso, é um ACK, portanto, corresponde a 2 bytes representando o número 4, o qual identifica essa operação.
- Identificação do pacote: aponta qual pacote esse ACK corresponde, portanto, consiste em 2 bytes representando o número de identificação do pacote. Esse valor pode variar de 0 até N, sendo:
 - 0: o ACK do pacote de solicitação de escrita;
 - 1 até N: os ACKs dos pacotes de dado. A quantidade de pacotes enviados, que é N, é definido pela divisão do tamanho do arquivo em 512 bytes, que é o tamanho máximo de dados que cada pacote comporta.

A exemplificação da mensagem de ACK é apresentada a seguir:

1 byte 1 byte 1 byte 1 byte

| 0 | 4 | 0 | 5 |

Além disso, quando o servidor recebe mais de uma solicitação de escrita vinda de um cliente diferente do que solicitou a operação inicialmente, é gerado um mensagem de erro e o envio de dados não é realizado.

A mensagem de erro é composta de:

- Código de operação: identifica que tipo de mensagem é essa que, no caso, é uma mensagem de erro, portanto, corresponde a 2 bytes representando o número 5, o qual identifica essa operação.
- Identificação do erro: aponta qual é o tipo de erro e consiste em 2 bytes representando o número zero, pois é o código do erro “não definido” (baseado no TFTP original), onde se olha a mensagem mandada junto, caso ela existir.
- Mensagem do erro: corresponde à mensagem de erro em si.
- Flag fim da mensagem de erro: identifica que a string acabou. Corresponde a 1 byte representando o número zero.

A exemplificação da mensagem de erro é apresentada a seguir:

```
1 byte 1 byte 1 byte 1 byte string 1 byte
-----
| 0 | 5 | 0 | 0 | mensagem | 0 |
-----
```

Documentação da API

A Tabela 1 apresenta a utilização da API SchaiLay descrevendo suas funções e aplicação de uso.

Tabela 1: Funções da API SchaiLay

Função	Descrição	Aplicação
SchaiLay.sendFile(serverIp, serverPort, fileName)	Função que envia um arquivo ao servidor, contendo o IP do servidor (serverIp), a porta (serverPort) e o nome do arquivo (fileName).	Cliente
SchaiLay.startServer(port)	Função que inicia o servidor através de uma porta especificada (port).	Servidor
SchaiLay.setTimeout(nSeconds)	Função que define o tempo de timeout em segundos. O padrão é que NSeconds seja 10 segundos.	Cliente e Servidor
SchaiLay.nRetries(nRetries)	Função que estabelece a quantidade de vezes que o cliente/servidor irá tentar enviar/receber o pacote em caso de timeout. O padrão é que nRetries seja 5.	Cliente e Servidor

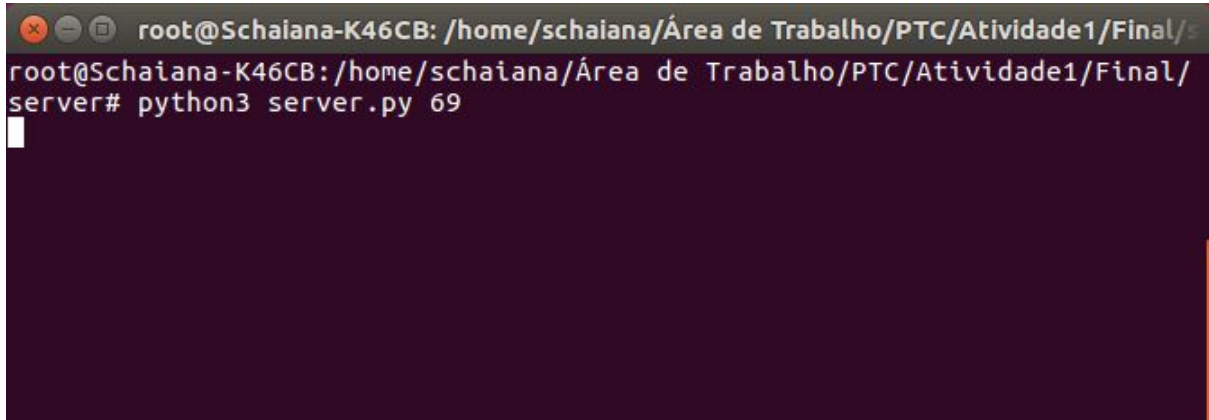
Manual do Usuário

Para executar as aplicações teste e verificar o funcionamento do protocolo SchaiLay, siga os passos abaixo:

1. Coloque o arquivo client.py, o arquivo SchaiLay.py e o arquivo que deseja transferir numa pasta. Em seguida, coloque o arquivo server.py e o arquivo SchaiLay.py em outra pasta, sendo esta diferente da pasta do cliente.

2. Execute a aplicação server.py passando como argumento o número da porta em que o servidor irá operar, conforme ilustrado na Figura 1. Exemplo: `python3 server.py 69`

Figura 1: Exemplo de uso da aplicação servidora.

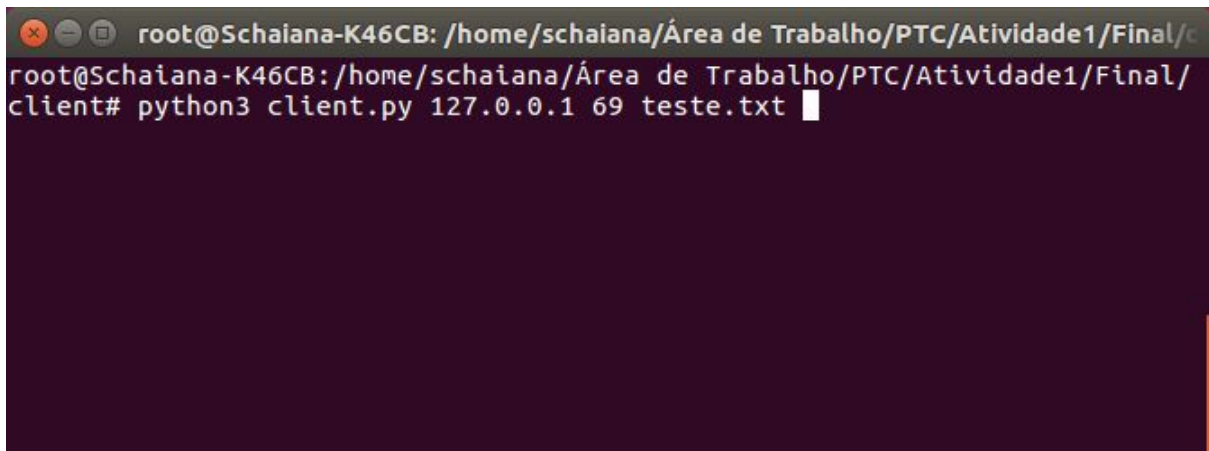
A terminal window with a dark purple background. The title bar shows window control icons and the text 'root@Schaiana-K46CB: /home/schaiana/Área de Trabalho/PTC/Atividade1/Final/'. The terminal text shows the command 'python3 server.py 69' being entered and executed, with a cursor on the line below.

```
root@Schaiana-K46CB: /home/schaiana/Área de Trabalho/PTC/Atividade1/Final/
root@Schaiana-K46CB: /home/schaiana/Área de Trabalho/PTC/Atividade1/Final/
server# python3 server.py 69
█
```

Fonte: Elaboração própria.

3. Execute a aplicação client.py passando como argumento o IP e a porta do servidor e o nome do arquivo a ser transferido, conforme ilustrado na Figura 2. Exemplo: `python3 127.0.0.1 69 teste.txt`

Figura 2: Exemplo de uso da aplicação cliente.

A terminal window with a dark purple background. The title bar shows window control icons and the text 'root@Schaiana-K46CB: /home/schaiana/Área de Trabalho/PTC/Atividade1/Final/c'. The terminal text shows the command 'python3 client.py 127.0.0.1 69 teste.txt' being entered and executed, with a cursor on the line below.

```
root@Schaiana-K46CB: /home/schaiana/Área de Trabalho/PTC/Atividade1/Final/c
root@Schaiana-K46CB: /home/schaiana/Área de Trabalho/PTC/Atividade1/Final/
client# python3 client.py 127.0.0.1 69 teste.txt █
```

Fonte: Elaboração própria.

Seguindo os passos acima, o arquivo será enviado do cliente para o servidor como esperado.