

REDES DE COMPUTADORES

TRABALHO PRÁTICO III - DOCUMENTAÇÃO

Wilson Moreira Tavares

2014068334

1. INTRODUÇÃO

Este trabalho prático envolve os conceitos trabalhados em sala, sobretudo o protocolo TCP e a biblioteca Socket. Foi implementado um sistema de armazenamento e leitura de arquivos no formato de pasta pessoal que possui suporte a múltiplas conexões.

Um socket de rede é o ponto-final de um fluxo de comunicação entre 2 aplicativos através de uma rede. Em cada ponta da comunicação implementada há um aplicativo, cliente e servidor, que compartilham informações. Um socket de rede em um computador é definido como a combinação de:

- um endereço IP
- o número de uma porta

2. SISTEMA

Foi implementado um sistema de armazenamento e leitura de arquivos, que é gerenciado por um servidor, que se conecta com múltiplos clientes e gerencia as mensagens recebidas. A aplicação cliente é controlada pelo usuário, que digita um comando seguindo os padrões descritos na especificação. O servidor, por sua vez, envia respostas para os clientes com um feedback predeterminado pela especificação. Na aplicação cliente, somente as respostas recebidas do servidor são imprimidas em tela.

Existem quatro tipos de comandos que podem ser enviados pelo cliente:

- Registro de usuário
- Envio de arquivo
- Recebimento (ou leitura) de conteúdo de arquivo
- Listagem de arquivos pertencentes a um usuário específico

[Registro de usuário] A mensagem é enviada da seguinte maneira:

N [nome_do_usuario] [senha]

Exemplo: N wilson 123

O servidor responde ao cliente, informando o sucesso ou não da operação. Abaixo estão as respostas possíveis:

Mensagem na tela	Explicação
N 0	Sucesso, usuário criado
N -1	Erro, usuário já existe

[Envio de arquivo] A mensagem é lida da seguinte maneira:

S [nome_do_usuario] [senha] [nome_do_arquivo] [conteudo_do_arquivo]

Exemplo: S wilson 123 meu_texto Mensagem importante

O servidor deve verificar se o usuário existe, confirmar se a senha é válida e se o arquivo já existe. Caso o arquivo exista, deve-se sobrescrevê-lo com a nova mensagem. Abaixo estão as respostas do servidor de acordo com os cenários descritos acima:

Mensagem na tela	Explicação
S 1	Sucesso, mensagem foi salva. Porém substitui um conteúdo já existente.
S 0	Sucesso, mensagem salva.
S -1	Erro, usuário não existe.
S -2	Erro, senha incorreta.

[Recebimento de conteúdo de arquivo] A mensagem é lida da seguinte maneira:

R [nome_do_usuario] [senha] [nome_do_arquivo]

Exemplo: R gabriel 12345 meu_diario

O servidor deve verificar se o usuário existe, confirmar se a senha é válida e se o arquivo existe. Abaixo estão as respostas do servidor de acordo com os cenários descritos acima:

Mensagem na tela	Explicação
R 0 [mensagem]	Sucesso, no lugar de [mensagem] está o conteúdo da mensagem.
R -1	Erro, usuário não existe.
R -2	Erro, senha incorreta.
R -3	Erro, mensagem não encontrada.

[Listagem de arquivos] A mensagem é lida da seguinte maneira:

L [nome_do_usuario] [senha]

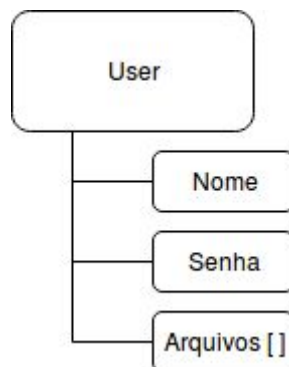
Exemplo: R gabriel 12345

Mensagem na tela	Explicação
L 0 [arquivo 1] [arquivo 2] [...]	Sucesso, no lugar de [arquivo...] está o nome dos arquivos. Se não houver nenhum, apenas "L 0" é retornado.
L -1	Erro, usuário não existe.
L -2	Erro, senha incorreta.

Existem instruções para o uso do software no arquivo "readme.txt", como foi definido em especificação.

3. IMPLEMENTAÇÃO

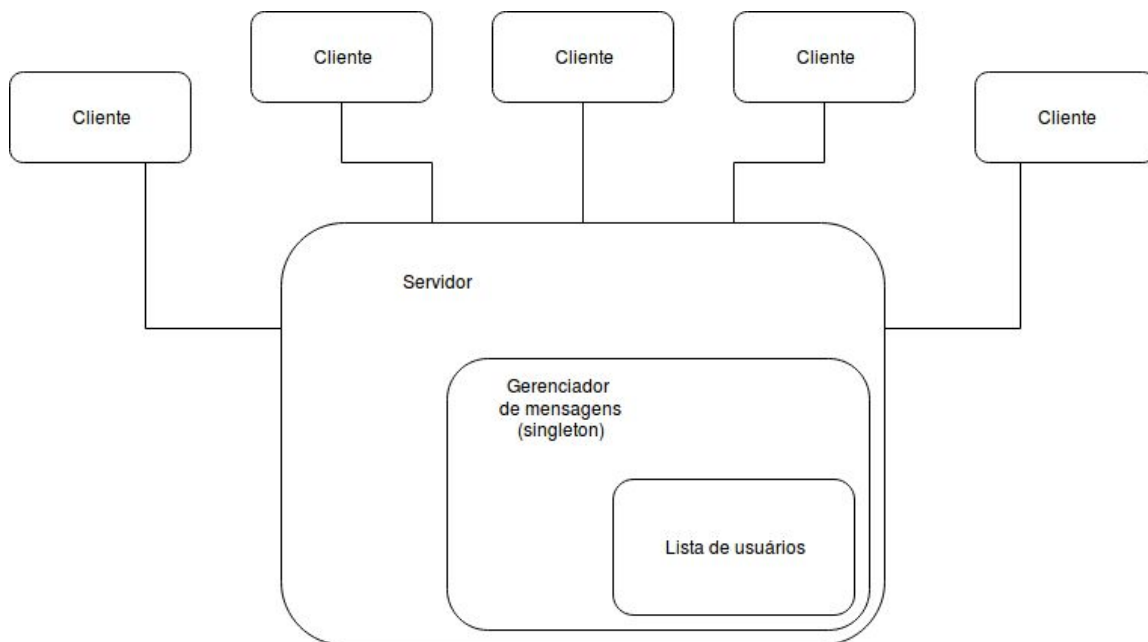
Para a implementação, foi usado o paradigma de orientação a objetos. A classe mais simples é a de **usuário** (User.py), que se constitui de login, senha e lista de arquivos referentes ao usuário. Desta forma, os dados são armazenados e consultados partindo desta estrutura de dados.



Para gerenciar as mensagens recebidas, armazenar e consultar dados sobre diversos usuários, foi criada **classe gerenciadora de mensagens** (GerenciadorDeMensagens.py). Esta classe contém uma lista de objetos *User* e uma série de métodos para lidar com as entradas enviadas pelos clientes.

Como o servidor deve lidar com múltiplas conexões de clientes, foram usadas threads para gerenciar as conexões. Além disso, a classe gerenciadora de mensagens foi implementada utilizando o padrão de projeto **singleton**, de forma a evitar inconsistências causadas pelo

acesso simultâneo. Desta forma, foi possível se conectar com múltiplos clientes e realizar as operações descritas acima de maneira concorrente, sem perdas de informação.



A implementação seguiu os padrões definidos pela especificação. A aplicação servidor faz uso de IPv6, assim como a aplicação cliente. Caso seja passado um endereço IPv4 para a aplicação cliente, é identificado que o endereço é IPv4 e o mesmo é convertido para IPv6.

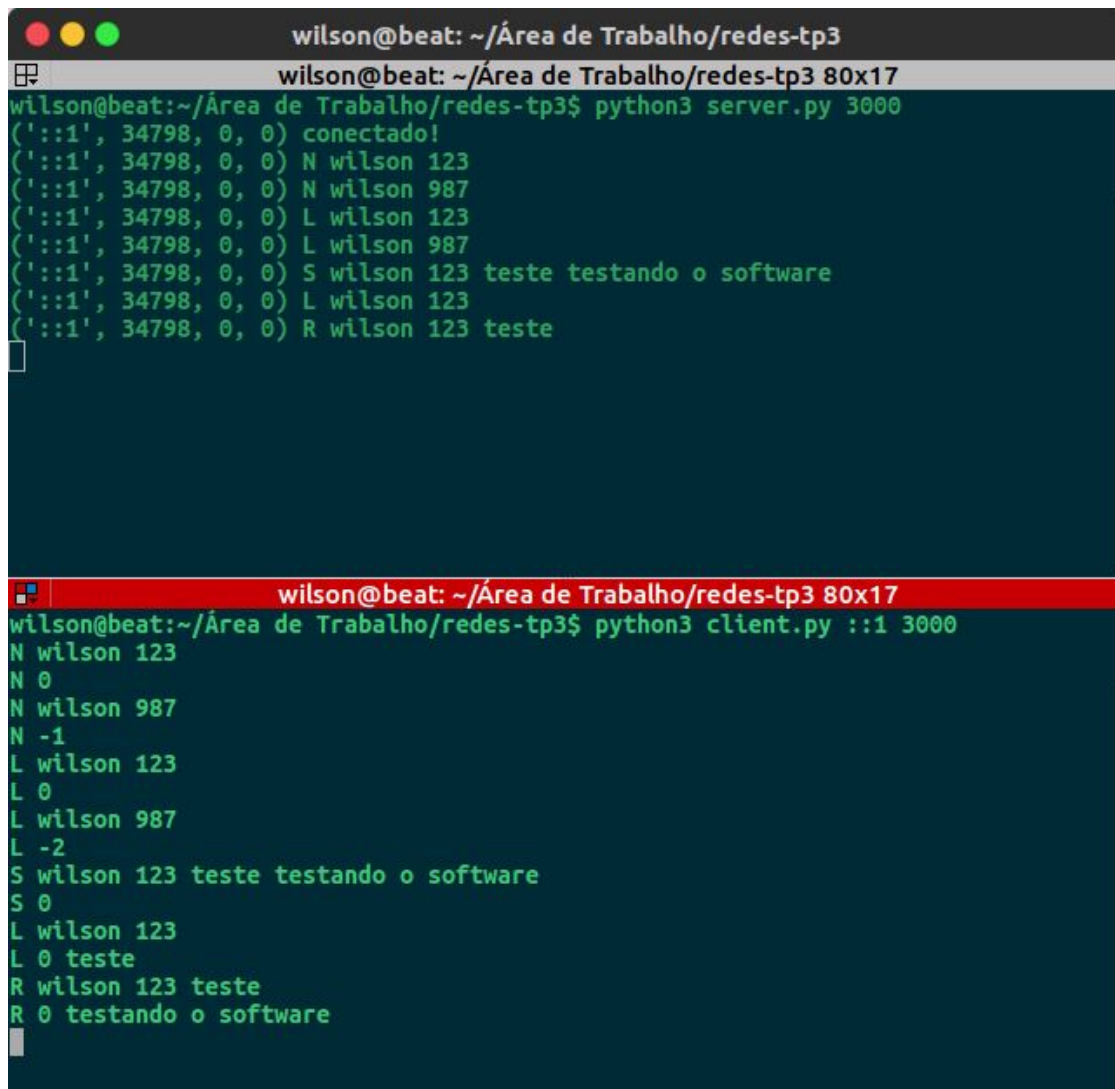
4. TESTES

Foram realizados testes com as seguintes situações:

- Inserção de usuário
 - novo usuário
 - tentativa de inserir usuário repetido
- Envio de arquivo
 - envio de arquivo com todos os parâmetros corretos
 - arquivo novo
 - arquivo existente
- Recebimento de conteúdo de arquivo
 - leitura de conteúdo de arquivo existente
 - tentativa de ler arquivo inexistente
- Listagem de arquivos por usuário

Além disso, os seguintes cenários foram testados:

- envio de mensagem com usuário inexistente
- envio de mensagem com senha errada



```
wilson@beat: ~/Área de Trabalho/redes-tp3
wilson@beat: ~/Área de Trabalho/redes-tp3 80x17
wilson@beat:~/Área de Trabalho/redes-tp3$ python3 server.py 3000
('::1', 34798, 0, 0) conectado!
('::1', 34798, 0, 0) N wilson 123
('::1', 34798, 0, 0) N wilson 987
('::1', 34798, 0, 0) L wilson 123
('::1', 34798, 0, 0) L wilson 987
('::1', 34798, 0, 0) S wilson 123 teste testando o software
('::1', 34798, 0, 0) L wilson 123
('::1', 34798, 0, 0) R wilson 123 teste

wilson@beat: ~/Área de Trabalho/redes-tp3 80x17
wilson@beat:~/Área de Trabalho/redes-tp3$ python3 client.py ::1 3000
N wilson 123
N 0
N wilson 987
N -1
L wilson 123
L 0
L wilson 987
L -2
S wilson 123 teste testando o software
S 0
L wilson 123
L 0 teste
R wilson 123 teste
R 0 testando o software
```

Em todos os cenários o servidor enviou respostas corretas, de acordo com o que foi especificado, seja com um ou com vários clientes. O número máximo de clientes testados foi 5. Não houveram tempos de espera, lentidões ou outros problemas do tipo durante a execução do programa.

5. CONCLUSÃO

A implementação deste trabalho prático foi de extrema importância para afixar e praticar os conhecimentos adquiridos em sala. Os maiores problemas que tive foram mais no que diz respeito à implementação de orientação a objetos em Python, tendo em vista que trabalho com a linguagem de programação Java e alguns pequenos detalhes são diferentes.

Acredito que o algoritmo de interpretação de mensagens pode ser melhorado, de forma a percorrer menos vezes a lista de usuários. De qualquer forma, o modo como foi implementado prioriza o reuso dos métodos em detrimento de uma perda de performance (não sentida em uma quantidade pequena/média de usuários).