Universidade Federal
de São João del-Rei

Universidade Federal
de São João Del Rei

UFSJ

Sistemas Distribuídos - ZeusFTP

https://github.com/exterminus/myftp

Professor:
Daniel Ludovico Guidoni

Desenvolvido por:
Carlos Magno Geraldo
Barbosa

Conteúdo

1 O problema

O protocolo de transferências de arquivos - FTP é um protocolo que facilita a transferência de arquivos entre computadores. O mesmo segue um paradigma de cliente e servidor. Este trabalho apresenta a construção de um servidor FTP distribuído, em que os servidor esta distribuído em um servidor de conexão, senha e arquivos. A comunicação entre os servidores é realizada através de um recurso chamado chamada de procedimento remoto RPC que permite a comunicação entre processos rodando em espaços de endereçamentos distintos[1].

2 Recursos

O zeusFTP foi implementado na linguagem python na sua versão 3. A conexão entre o cliente e o servidor é protegido através do protocolo de comunicação segura SSL. Os dados dos usuários são armazenados cifrados em um banco de dados não relacional. Os principais recursos presentes nesta implementação são os seguintes:

- Implementa protocolo de comunicação segura SSL.
- Permite acesso de múltiplos usuários.
- Permite a transferência de arquivos de diversos tipos.
- Implementa uma serie de comandos básicos de clientes FTP.

3 Arquitetura e Implementação

O presente trabalho esta organizado da seguinte maneira: I) Cliente, II)Servidor conexão, III)Servidor de arquivos e um V) servidor de senha. Esta divisão pode ser observada na figura 1.

3.1 Cliente

O modulo do cliente contém a classe Cliente que contém todos os métodos relacionadas a interação do usuário com servidor. Os principais métodos desta classe podem ser observados na figura 2.

Os seguintes métodos estão presentes nesta classe:

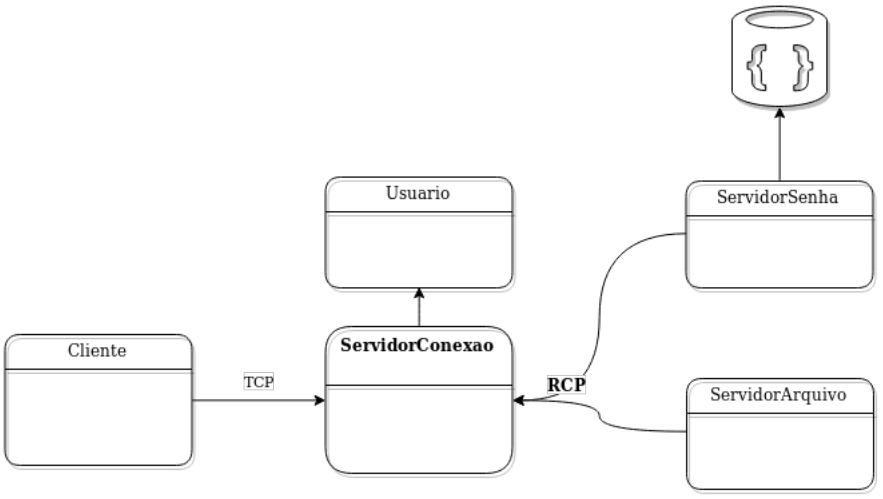

Figura 1: Diagrama Completo

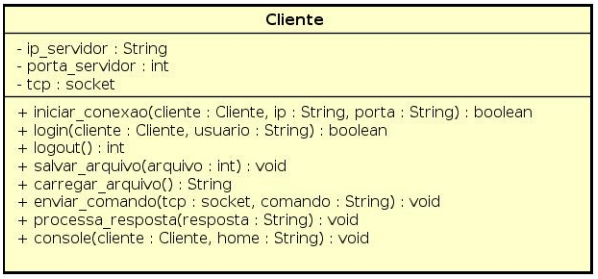

Figura 2: Diagrama Cliente

- **inicia_conexao**:Inicia a conexão com o servidor de conexão.

  Parâmetro entrada: ip,porta

  Retorno: boolean,mensagem:dicionario

- **login**:Envia os dados de login para o servidor de conexão.

  Parâmetro entrada: login:String,senha:String

  Retorno: boolean

- **logout**:Encerra a conexão com servidor.

  Parâmetro entrada: comando:String

  Retorno: boolean

- **salvar_arquivo**:Salva um arquivo recebido do servidor localmente.

Parâmetro entrada: nome:String, arquivo:String

Retorno: void

- **carregar_arquivo**:Carrega um arquivo.

  Parâmetro entrada: nome: String

  Retorno: String

- **enviar_comando**:Envia os comandos do usuário para o servidor.
- **processa_resposta**:Trata e exibe a resposta recebida do servidor.

  Parâmetro entrada: self:Cliente,resposta:dicionario

  Retorno: void

- **console**:Exibe e formata os comandos digitados pelo usuário. Esta função somente é chamada após a confirmação de login do cliente.

  Parâmetro entrada: self:Cliente, home:String

  Retorno: void

3.2 Servidor de Conexão

Este módulo tem como principal função realizar a conexão entre o cliente e os servidores de arquivo e senha. A comunicação entre este módulo e o cliente é realizada por meio do protocolo TCP com uma camada segura utilizando o SSL.Os principais métodos deste módulo podem ser observados na figura 3.A maioria dos métodos presentes neste modulo estão implementados em outras classes e até mesmo em outras máquinas e são instanciadas através do protocolo RCP.

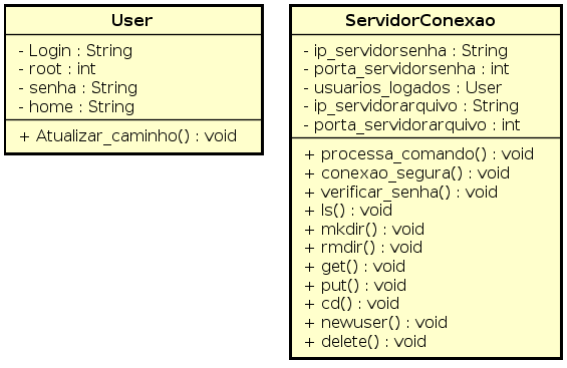

Figura 3: Servidor de Conexão

Podemos destacar os seguintes métodos implementados neste módulo:

- **processar_comando**: Este método processa os comandos recebidos do do cliente. Neste método as funções do servidor de senha e de arquivos são instanciadas.

  Parâmetro entrada: conexao:tcp ,comando:dicionário, user: Usuario
  Retorno: void

- **ConexaoSegura**: Este método é responsável por estabelecer a conexão segura com os clientes.

  Parâmetro entrada: self: ServidorConexao

  Retorno: void

- **verifica_senha**: Este método é instanciado através do RCP, o mesmo esta implementado no módulo de senha. Esta função tem como objetivo validar as credencias de acesso do usuário.

  Parâmetro entrada: login:String,Senha:String

  Retorno: boolean

- **ls**: Este método está implementado no servidor de arquivo. O mesmo tem como função lista os arquivos de um diretório.

  Parâmetro entrada: caminho:String

  Retorno: boolean

- **mkdir**:Este método é instanciado através da chamada de procedimento remoto, o mesmo está implementado no servidor de arquivos. A sua função é permitir a criação de um diretório.

  Parâmetro entrada: nome:String

  Retorno: boolean

- **rmdir**:Este método é instanciado através da chamada de procedimento remoto, o mesmo está implementado no servidor de arquivos. A sua função é permitir a remoção de um diretório.

  Parâmetro entrada: nome:String

  Retorno: boolean

- **get**:Este método é instanciado através da chamada de procedimento remoto, o mesmo está implementado no servidor de arquivos. A sua função é permitir

a transferência de um arquivo do servidor remoto para o diretório local.

Parâmetro entrada: caminho:String Retorno: boolean

- **put**:Este método é instanciado através da chamada de procedimento remoto, o mesmo está implementado no servidor de arquivos. A sua função é permitir o envio de um arquivo para o servidor remoto.

  Parâmetro entrada: self:ServidorConexao

  Retorno: Boolean,lista:list

- **newuser**:Este método é instanciado através da chamada de procedimento remoto, o mesmo está implementado no servidor de senha. A sua função é permitir a criação de um novo usuário.

  Parâmetro entrada: usuario:String,senha:String,permissao:String

  Retorno: boolean

- **delete**:Este método esta implementado no servidor de arquivos, a sua principal função é permitir a remoção de um arquivo.

- **help**:Este método esta implementado no servidor de arquivos, a sua principal função é exibir os comandos válidos.

  Parâmetro entrada: void

  Retorno: boolean

3.3 Servidor de Arquivo

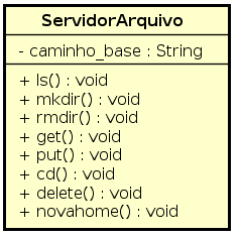

Figura 4: Servidor de Arquivo

Este modulo implementa os métodos responsáveis pela escrita e manipulação dos arquivos. Os métodos implementados nesta classe são instanciados na classe de conexão.

Podemos destacar o método **novahome** que é utilizado para criação da home de um novo usuário. O mesmo é chamado quando é necessário se criar um novo usuário no sistema.

3.4 Servidor de Senha

Este módulo implementa os método relacionados a criação e validação de credencias de um novo usuário. A figura 5 apresenta as classes presentes neste módulo.

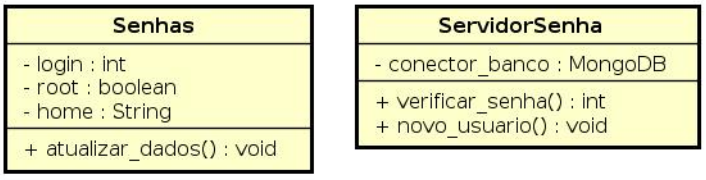

Figura 5: Servidor de Senha

A função **verifica_senha** realiza a verificação dos dados de login de um usuário.Este método é instanciado na classe de conexão. Por sua vez a função **novo_usuario** é responsável pela criação e registro de um novo usuário no banco de dados.

3.5 O protocolo de comunicação

Para permitir a comunicação entre o cliente e o servidor foi definido um padrão de comunicação baseado em um estrutura chave valor chamada dicionário.

Exemplo do padrão de comunicação:

```
mensagem={}
mensagem["comando"]="help"
mensagem['complemento']=""
mensagem['file']=None
```

Neste exemplo é enviado uma instrução de **help** para o servidor de conexão, que recebe, processa e responde a mensagem seguindo o mesmo padrão.

Os dicionários são convertidos para bytes permitindo a sua serialização através de um modulo python chamado pickles.

4 Como Utilizar

Para executar o ZeusFTP é necessário seguir os seguintes passos de configuração:

- Instalar o mongodb e o pymongo no computador a ser utilizado como servidor de senha. Instale o pymongo utilizando o pip.
- Configurar o IP do servidor de senha e do servidor de arquivos no servidor de conexão.
- Execute o código para criar um novo usuário:
  - python3 criar_usuario.py

Como executar:

1. Inicie o servidor de senha.

   - python3 ServidorSenha.py

2. Inicie o Servidor de Arquivos

   - python3 ServidorArquivo.py

3. Inicie o servidor de Conexão, verifique os ip's do servidor de arquivos e senha antes de executar.

   - python3 ServidorConexao.py [porta]

4. Inicie o cliente:

   - python3 Cliente.py [ip_servidor_conexao] [porta] [login]

5 Conclusão

Por meio deste trabalho foi possível se exercitar os conceitos básicos de redes e sistemas distribuídos ensinados em sala de aula. O principal ganho com este trabalho foi aprender a trabalhar com a chamada de procedimento remoto.

Referências

[1] Andrew S Tanenbaum and Maarten Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.