**Aplicação cliente servidor utilizando sockets**

**Émerson Teixeira, Mateus Gündel, Patrick Martini**

Departamento de Computação - Universidade de Santa Cruz do Sul (UNISC)

Santa Cruz do Sul, RS

emersonrt@mx2.unisc.br, gundel5@mx2.unisc.br, plmartini@mx2.unis.br

*Abstract. This paper describes the system, technologies and features implemented in a client server application, using TCP, UDP, SMTP, POP and IMAP protocols.*

*Resumo. Neste paper são descritos o sistema, as tecnologias e as funcionalidades implementadas em uma aplicação cliente servidor, sendo utilizados os protocolos TCP, UDP, SMTP, POP e IMAP.*

# **INTRODUÇÃO**

O sistema desenvolvido se trata de uma aplicação cliente servidor, utilizando sockets (TCP e UDP), na linguagem Python. Sua principal funcionalidade é, utilizar cliente e servidor como backups remotos, ou seja, ao ocorrer quaisquer alterações, exclusões ou adições em uma pasta específica, o sistema realiza um backup remoto e sincronizado no cliente ou no servidor. Outra funcionalidade da aplicação é, sempre que ocorrerem alterações destes arquivos, um e-mail é enviado como notificação ao administrador, com dados da modificação (nome do arquivo, data e hora), utilizando os protocolos SMTP, POP e IMAP. A última funcionalidade é um sistema de monitoramento, utilizando sistema de ping com socket UDP/ICMP.

1. **INTRODUÇÃO AOS PROTOCOLOS**

**TCP (Transmission Control Protocol) –** é um protocolo que define como estabelecer e manter uma comunicação de rede, faz parte da camada de transporte.

É orientado a conexão, o que significa que uma conexão deve ser estabelecida e mantida até os programas da aplicação de cada sistema finalizarem a conexão. Ele determina como dividir os dados do aplicativo em pacotes que as redes possam entregar, controla o fluxo de dados, lida com a retransmissão de dados perdidos ou inelegíveis, e com o reconhecimento de todos pacotes que chegam.

**UDP (User Datagram Protocol) –** é uma alternativa ao protocolo TCP, usado primordialmente para aplicações que necessitam de baixa latência e não necessitam garantia de entrega.

Por permitir que datagramas sejam recebidos e descartados em uma ordem diferente da que foram transmitidos, é o protocolo mais adequado para DNS e aplicações onde muitos clientes está conectado e a correção de erros em tempo real não é necessária, como jogos, videoconferências e mídia de streaming.

Existem grandes diferenças entre os protocolos TCP e UDP. Enquanto TCP estabelece conexão host-to-host, UDP estabelece comunicação process-to-process. O TCP envia pacotes individuais e é considerado confiável o UDP envia mensagens (chamadas datagramas), e é considerado um modo de comunicação de melhor esforço.

**SMTP (Simple Mail Transfer Protocol) –** é o protocolo usado para envio de e-mails, para o recebimento de e-mails necessita dos protocolos POP ou IMAP. Só é utilizado sozinho no caso de disparo de e-mails em massa (e-mail marketing). Portas: 465, 587.

**POP (Post Office Protocol) –** é o protocolo que permite conectar-se ao servidor e realizar o download de todas as mensagens existentes desde a última verificação. Porta POP3: 995

**IMAP (Internet Message Access Protocol) –** diferente do protocolo pop, este permite sincronizar pastas e outras marcações de mensagens, como tags, por exemplo. Desta maneira, o programa leitor de e-mail será um espelho das mensagens e da sua organização no servidor. Porta: 993

1. **FUNCIONALIDADES DO SISTEMA**

A funcionalidade principal do código, o monitoramento e transferência de arquivos utilizando os protocolos TCP e UDP, foram implementadas com a biblioteca “socket”. Alguns detalhes da estrutura básica de programas para rede:

* A interface de rede utiliza os fundamentos da API de Socket
* Todo socket pode estar em modo ativo ou passivo
* Quando é criado está em moto ativo
* Para o servidor “escutar” uma porta é necessário colocar o socket do servidor em modo passivo
* Uma sequência de passos para se estabelecer a comunicação é executada, sendo diferente para TCP e UDP.
* Sequência no Cliente:

1. Se foi fornecido um nome de hospedeiro converter em endereço IP;
2. Se foi fornecido um nome de protocolo de transporte converter em número;
3. Criar o socket (função socket);
4. Conecta com o servidor (função connect);
5. Enviar/Receber dados (permanecer nesse passo enquanto tiver dados para enviar/receber);
6. Fechar o socket.

* Sequência no Servidor:

1. Se foi fornecido um nome de protocolo de transporte converter em número;
2. Criar o socket (função socket);
3. Coloca um endereço local, endereço IP e porta, no socket (função bind);
4. Instrui o sistema operacional para colocar o socket em modo passivo (função listen); **SOMENTE EM TCP**
5. Aceita uma nova conexão (função accept); **SOMENTE EM TCP**
6. Enviar/Receber dados (permanecer nesse passo enquanto tiver dados para enviar/receber);
7. Fechar o socket.
8. Volta ao passo 5 para aceitar outra conexão.
9. **REFERÊNCIAS**
10. Orestes, Yan, TCP, UDP e protocolos: quais as diferenças. Acessado em 02/12/2019 em <https://www.alura.com.br/artigos/quais-as-diferencas-entre-o-tcp-e-o-udp>
11. H., Rafael, Portas POP3, SMTP e IMAP – Protocolos de Email Explicados. Acessado em 03/12/2019 em <https://www.hostinger.com.br/tutoriais/portas-pop3-smtp-e-imap/>