
**UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI - UFSJ**
**REDES DE COMPUTADORES 2024/1**
**RAFAEL SACHETTO OLIVEIRA**

**Trabalho prático 3 - Em Grupo (máximo 3 pessoas)**

# 1 Objetivo

O objetivo deste trabalho é implementar um **protocolo de parada e espera** para transferência confiável de dados, simulando um cenário de envio de pacotes entre um cliente e um servidor. O protocolo deverá garantir que os dados sejam entregues corretamente, lidando com perda de pacotes e retransmissão, além de simular os conceitos de ACKs e timeouts.

# 2 O Trabalho

Neste trabalho, os alunos devem implementar tanto o **cliente** quanto o **servidor**, utilizando **sockets UDP**, e simular o funcionamento de um **protocolo de parada e espera (Stop-and-Wait)**. O protocolo será responsável por transferir um arquivo do cliente para o servidor de forma confiável, mesmo sobre um canal de comunicação que pode perder pacotes ou introduzir atrasos.

## Funcionamento do Protocolo

- **Parada e Espera (Stop-and-Wait)**: O cliente deve enviar um pacote de dados e aguardar um **ACK** (confirmação) do servidor antes de enviar o próximo pacote. O servidor, ao receber um pacote, deve enviar um ACK de volta ao cliente.
- **Retransmissão em caso de perda**: Se o cliente não receber o ACK dentro de um tempo limite (timeout), ele deve retransmitir o pacote. Isso garante a entrega confiável mesmo em cenários onde há perda de pacotes.

- **Controle de Sequência**: Para evitar duplicação de pacotes devido à retransmissão, cada pacote de dados enviado pelo cliente deve conter um **número de sequência**. O servidor deve usar esse número para reconhecer pacotes duplicados e descartar pacotes já recebidos.

# Requisitos Técnicos

## Cliente

- O cliente deverá ler um arquivo local e enviá-lo ao servidor, dividindo-o em pacotes de tamanho fixo (ex.: 1024 bytes por pacote).
- O cliente deverá usar sockets **UDP** para transmitir os pacotes, e esperar um **ACK** do servidor após cada envio.
- Se o cliente não receber o ACK dentro de um intervalo de tempo pré-definido (ex.: 2 segundos), ele deverá retransmitir o pacote.
- O cliente deverá manter o controle de quantos pacotes foram enviados e quantos foram retransmitidos.

## Servidor

- O servidor deverá escutar em uma porta UDP e receber pacotes de dados do cliente.
- O servidor deverá enviar um **ACK** de volta para o cliente após o recebimento correto de cada pacote.
- Se o servidor receber pacotes duplicados (mesmo número de sequência), ele deverá descartá-los e enviar o ACK correspondente.
- O servidor deverá reconstruir o arquivo recebido e salvá-lo em disco.

## Simulação de Perda de Pacotes

- Para tornar o protocolo mais interessante, deve ser implementada uma simulação de perda de pacotes. O cliente ou o servidor deve, aleatoriamente, ”perder” pacotes ou ACKs (por exemplo, deixando de enviar ou receber com uma probabilidade de 10%).

- Isso permitirá que a eficiência da retransmissão e o funcionamento correto do protocolo em um ambiente com falhas sejam testados.

### Timeout e Retransmissão

- O cliente deve implementar um mecanismo de **timeout** para retransmitir pacotes se não receber o ACK no tempo esperado.
- O timeout deve ser ajustável e definido em um valor razoável (ex.: 2 segundos).

## Critérios de Avaliação

- **Funcionamento correto**: O cliente deve ser capaz de enviar o arquivo ao servidor, que deverá recebê-lo completamente e de forma confiável, mesmo com perda de pacotes simulada.
- **Retransmissão correta**: O cliente deve retransmitir pacotes adequadamente quando um ACK não for recebido dentro do tempo esperado.
- **Controle de duplicatas**: O servidor deve identificar e descartar pacotes duplicados corretamente.
- **Simulação de perda de pacotes**: O sistema deve simular a perda de pacotes de forma realista, testando a robustez do protocolo implementado.
- **Organização e clareza do código**: O código deve ser bem documentado, modular e organizado, seguindo boas práticas de programação.

## Relatório

Deve ser apresentado um relatório explicando:

- O funcionamento do protocolo de parada e espera.
- A implementação dos sockets em C e como o controle de sequência e retransmissão foi implementado.
- Os testes realizados, incluindo a simulação de perda de pacotes e os resultados obtidos (por exemplo, número de pacotes retransmitidos).

- Discussão sobre os desafios encontrados durante a implementação.

## Entregáveis

- Código-fonte do cliente e servidor.
- Arquivo de teste utilizado (pode ser um arquivo de texto ou binário de tamanho moderado, ex.: 1MB).
- Relatório em formato PDF.