Aplicações em rede usando TCP e UDP

```
#define TRABALHO 1
#define SEMESTRE "2024.2"
#define PROFESSOR "Rafael Sachetto"
#define ESTUDANTES ["Breno Esteves", "Gabriel de Paula", "Guilherme Francis"]
```

🖥️ Como executar

Arquivos de teste

Por padrão os arquivos são incluídos na pasta `tests/server`, para que o servidor possa ler e enviar para o cliente em `tests/client`. É possível inserir seus próprios arquivos de teste no diretório.

Verificação de integridade

É interessante verificar a integridade dos arquivos usando hashs, uma sugestão é a utilização do MD5, que pode criar arquivos de hash da seguinte forma:

```
md5sum arquivo.txt > arquivo.md5
```

Automatização da geração de arquivos

Gere os arquivos de teste (diferentes tamanhos) usando o script em python:

```
python3 tests/generate.py
```

> São criados dez arquivos de dados (`.bin`) de 1,2,3...,10 MB e os respectivos dez arquivos de verificação usando hash (`.md5`).

Executáveis

Os executáveis por padrão são gerados na pasta `bin` e os objetos de compilação na pasta `build`.

Compilando

Gere os executáveis de cada protocolo usando `make`:

X	TCP	UDP
Servidor	`make tcp_server`	`make udp_server`
Cliente	`make tcp_client`	`make udp_client`

Executando o servidor

Para executar o servidor é necessário informar o endereço de IP e a porta a qual o servidor deve escutar por conexões.

```
./bin/tcp_server -i <ip> -p <porta>
```

```
./bin/udp_server -i <ip> -p <porta>
```

Executando o cliente

Para executar o cliente também é necessário definir o IP e a porta (devem ser iguais aos do servidor).

```
./bin/tcp_client -i <ip> -p <porta> -f <arquivo>
```

```
./bin/udp_client -i <ip> -p <porta> -f <arquivo>
```

> Informar o arquivo diretamente da linha de comando é opcional, caso não informado, o programa irá solicitar que o usuário insira o nome manualmente.

Relatórios

Automatização da execução de múltiplos clientes

É possível automatizar a execução de clientes para facilitar o processo de coleta de dados, fazendo com que o mesmo arquivo seja solicitado um determinado número inteiro de iterações. Basta executar o script em Python conforme demonstrado abaixo:

```
python3 tests/run.py <executavel> <ip> <porta> <iteracoes>
```

> É necessário que os executáveis estejam compilados e o servidor do respectivo protocolo esteja rodando no mesmo IP e porta

Gráficos dos resultados

Após a execução do comando anterior, basta executar o script abaixo fornecendo o caminho do arquivo de resultados gerado. São exibidos alguns gráficos que podem ser salvos localmente.

```
python3 tests/report.py <arquivo>
```

> Algumas bibliotecas externas devem ser instaladas: `pandas`, `matplotlib` e `seaborn`