Pontifícia Universidade Católica de Minas Gerais
Bacharelado em Ciência da Computação
Redes de Computadores I – 2º Semestre de 2019
Profª. Raquel Mini

Trabalho Prático

Valor: 20 pontos (trabalho em dupla)
Data de entrega: 17/11/2019

Batalha Naval

O principal objetivo do trabalho é praticar a programação com a biblioteca socket e usando o protocolo TCP. O trabalho consiste em desenvolver o jogo batalha naval para o paradigma cliente-servidor. Esta versão pode funcionar com apenas um cliente conectado. Você deverá criar um programa cliente e outro programa servidor. Batalha naval é um jogo de tabuleiro no qual o objetivo é afundar a frota de navios do inimigo. Inicialmente é definido um tabuleiro (matriz) de 10 x 10. Os quadrados do tabuleiro são identificados na horizontal por números e na vertical por letras. Os tabuleiros não são visíveis entre os jogadores. Os tipos de navios são, ocupando quadrados adjacentes na horizontal ou vertical: porta-aviões (cinco quadrados), navios-tanque (quatro quadrados), contratorpedeiros (três quadrados) e submarinos (dois quadrados). O número de navios são: 1, 2, 3 e 4 respectivamente. O jogo consiste em escolher um espaço do tabuleiro oponente e tentar acertar um dos navios da frota. O navio afunda quando todos quadrados forem adivinhados pelo oponente, quem perder todos os navios primeiro perde o jogo.
Os programas irão funcionar utilizando o TCP.

O trabalho prático consistirá na implementação de dois programas, o cliente e o servidor. Ambos vão receber parâmetros de entrada como explicado a seguir:

```
cliente <ip/nome> <porta>
servidor <porta>
```

O primeiro programa, o cliente, irá conectar no servidor definido pelo endereço IP e porta passados por parâmetro. Já o servidor irá executar um serviço, que irá tratar uma conexão por vez. A porta a ser escutada é dada pelo parâmetro único do programa.

O cliente irá se conectar ao servidor e o jogo irá se iniciar. O cliente deve posicionar sua frota. Isso pode ser feito lendo um arquivo de entrada. O servidor pode posicionar sua frota escolhendo aleatoriamente a posição inicial e a orientação (vertical, horizontal) dos navios. O cliente irá enviar a posição escolhida para o tiro, que pode ser lida do teclado. O servidor irá responder se acertou ou não e também a posição do seu tiro, e assim por diante até o jogo terminar. O servidor pode implementar a escolha da posição do tiro aleatoriamente, e se acertar, ele deve escolher um quadrado vizinho. No cliente, se pressionado a letra P, ele deve imprimir (em ASCII) seu tabuleiro e o tabuleiro com o que ele já aprendeu do servidor.

A documentação deve ser entregue em pdf junto ao código. Implementações modularizadas deverão mencionar quais funções são implementadas em cada módulo ou classe. A documentação deve conter os seguintes itens:

- Sumário do problema a ser tratado.
- Uma descrição sucinta dos algoritmos e dos tipos abstratos de dados, das principais funções, e procedimentos e as decisões de implementação.
- Decisões de implementação que porventura estejam omissos na especificação.

- Testes, mostrando que o programa está funcionando de acordo com a especificação, seguidos da sua análise. Print screens mostrando o correto funcionando do cliente e servidor e exemplos de testes executados.
- Conclusão e referências bibliográficas.

Sistema de Preços

O principal objetivo do trabalho é praticar a programação com a biblioteca Socket e utilizando o protocolo UCP. O trabalho consiste em desenvolver um sistema para enviar o preço de vários postos de combustíveis e requisitar o preço mais barato em uma dada região. Você deverá criar um programa cliente e outro programa servidor. Os programas irão funcionar utilizando o UDP.

O trabalho prático consistirá na implementação de dois programas, o cliente e o servidor. Ambos vão receber parâmetros de entrada como explicado a seguir:

```
cliente <ip/nome> <porta>
servidor <porta>
```

O primeiro programa, o cliente, irá conectar no servidor definido pelo endereço IP e porta passados por parâmetro. Já o servidor irá executar um serviço, que irá tratar uma comunicação por vez, mas que poderá comunicar com vários clientes. A porta a ser escutada é dada pelo parâmetro único do programa.

O cliente poderá enviar dois tipos de mensagens ao servidor: dados (D) e pesquisa (P). Como o UDP não garante a entrega de mensagens, o cliente deve implementar pelo menos uma retransmissão, caso a mensagem não seja recebida a primeira vez, para tentar garantir a entrega de mensagens. O servidor deve confirmar a recepção de mensagens. As mensagens de dados começam com a letra D seguida de um inteiro identificador da mensagem, um inteiro que indica o tipo de combustível (0 - diesel, 1 - álcool, 2- gasolina), um inteiro com o preço x 1000 (ex: R$3,299 fica 3299) e as coordenadas do posto de combustível (latitude e longitude). O servidor deverá confirmar a recepção da mensagem e adicionar a informação em um arquivo. Vários clientes podem enviar dados e mesmo com o término da comunicação entre um cliente e um servidor os dados enviados devem ser salvos no arquivo para consultas de outros clientes. As mensagens de pesquisa começam com a letra P seguida de um inteiro identificador da mensagem, um inteiro que indica o tipo de combustível (0 - diesel, 1 - álcool, 2- gasolina), um inteiro com o raio de busca e as coordenadas do centro de busca (latitude e longitude). O servidor deverá responder com o menor preço para aquele combustível para postos de combustível que estejam no centro de busca mais raio de busca. Para verificar o correto funcionamento dos programas, o cliente e o servidor devem imprimir na tela o conteúdo das mensagens que eles receberem.

A documentação deve ser entregue em pdf junto ao código. Implementações modularizadas deverão mencionar quais funções são implementadas em cada módulo ou classe. A documentação deve conter os seguintes itens:

- Sumário do problema a ser tratado.
- Uma descrição sucinta dos algoritmos e dos tipos abstratos de dados, das principais funções, e procedimentos e as decisões de implementação.
- Decisões de implementação que porventura estejam omissos na especificação.
- Testes, mostrando que o programa está funcionando de acordo com a especificação, seguidos da sua análise. Print screens mostrando o correto funcionando do cliente e servidor e exemplos de testes executados.
- Conclusão e referências bibliográficas.