

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS
GERAIS

Instituto de Ciências Exatas e de Informática

Trabalho Prático - Redes I

Lucas Carvalho Alves Nogueira

Belo Horizonte
2020

Sumário

1	Batalha Naval	2
1.1	Introdução ao problema	2
1.2	Implementação	2
1.3	Testes	2
2	Sistemas de Preços	6
2.1	Introdução ao problema	6
2.2	Implementação	6
2.3	Testes	7
3	Conclusão	11
4	Referências	12

1. Batalha Naval

1.1. Introdução ao problema

O programa desenvolvido tem como objetivo a prática do uso de bibliotecas socket utilizando o protocolo TCP. Essa é uma aplicação de um jogo de Batalha Naval, que é um jogo de tabuleiro no qual o objetivo é afundar toda a frota inimiga.

Um tabuleiro 10x10 foi definido como padrão. Os quadrados do tabuleiro são identificados na horizontal por numeros (1-10) e na vertical por letras (A-J). As configurações dos navios que são utilizadas no jogo são:

- Porta-Aviões (Tamanho = 5 | Quantidade = 1)
- Navio-Tanque (Tamanho = 4 | Quantidade = 2)
- Contratorpedeiro (Tamanho = 3 | Quantidade = 3)
- Submarino (Tamanho = 2 | Quantidade = 4)

O jogo consiste em cada jogador escolher onde posicionar seus navios e depois escolher em turnos uma posição do tabuleiro inimigo até encontrar e afundar toda a frota inimiga.

Por meio do protocolo TCP foi desenvolvido dois programas, um servidor e um cliente que deve receber os seguintes parâmetros:

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

1.2. Implementação


O código dos programas foram implementados em Python e foi utilizada a biblioteca socket. Utilizado o parâmetro adequado para sockets TCP.

As etapas para a execução do jogo são:

- Conectar o servidor em uma determinada porta.
- Conectar o cliente ao servidor informando o endereço IP e a porta.
- Começar a posicionar sua frota, escolhendo linha, coluna e orientação.
- Após o termino da inserção da frota começa o jogo.
- Em turnos é escolhida uma posição por vez
- Jogo termina até que todos os navios sejam afundados (necessário 30 acertos)

1.3. Testes

Conectar o servidor em uma determinada porta.



```
Porta: _
```

Servidor iniciado

```
Porta: 3333
Servidor iniciado. Aguardando conexões...
Host: 192.168.15.15
Porta: 3333
-
```

Agora no client, conectar o cliente ao servidor informando o endereço IP e a porta

```
Batalha Naval

Insira o IP do servidor: 192.168.15.15
Insira a porta para conexão: 3333
```

Servidor confirmando conexão

```
Porta: 3333
Servidor iniciado. Aguardando conexões...
Host: 192.168.15.15
Porta: 3333
192.168.15.15 conectado. Preparando novo jogo...
```

Voltando pro cliente, começa a posicionar sua frota, escolhendo linha, coluna e orientação.

```
Batalha Naval

Insira o IP do servidor: 192.168.15.15
Insira a porta para conexão: 3333
Escolha uma linha (A-J): -
```

Exibe os tabuleiros após cada inserção

```

Insira o IP do servidor: 192.168.15.15
Insira a porta para conexão: 3333
Escolha uma linha (A-J): a
Escolha uma coluna (1-10): 1
Insira a direção (Horizontal: H, Vertical: V): v

Tabuleiro Server
  1  2  3  4  5  6  7  8  9  10
A | -  -  -  -  -  -  -  -  -  -
B | -  -  -  -  -  -  -  -  -  -
C | -  -  -  -  -  -  -  -  -  -
D | -  -  -  -  -  -  -  -  -  -
E | -  -  -  -  -  -  -  -  -  -
F | -  -  -  -  -  -  -  -  -  -
G | -  -  -  -  -  -  -  -  -  -
H | -  -  -  -  -  -  -  -  -  -
I | -  -  -  -  -  -  -  -  -  -
J | -  -  -  -  -  -  -  -  -  -

O inimigo acertou 0 posições
Você acertou 0 posições

Legenda:
p: Porta-Aviões
t: Navio-Tanque
c: Contratorpedeiro
s: Submarino
*: Falha
x: Acerto
-: Posição livre

Meu Tabuleiro
  1  2  3  4  5  6  7  8  9  10
A | p1 -  -  -  -  -  -  -  -  -
B | p1 -  -  -  -  -  -  -  -  -
C | p1 -  -  -  -  -  -  -  -  -
D | p1 -  -  -  -  -  -  -  -  -
E | p1 -  -  -  -  -  -  -  -  -
F | -  -  -  -  -  -  -  -  -
G | -  -  -  -  -  -  -  -  -
H | -  -  -  -  -  -  -  -  -
I | -  -  -  -  -  -  -  -  -
J | -  -  -  -  -  -  -  -  -

```

Início de jogo. Em turnos é escolhida uma posição por vez

```

Escolha uma linha (A-J): a
Escolha uma coluna (1-10): 10
Insira a direção (Horizontal: H, Vertical: V): v

Tabuleiro Server
  1  2  3  4  5  6  7  8  9  10
A | -  -  -  -  -  -  -  -  -  -
B | -  -  -  -  -  -  -  -  -  -
C | -  -  -  -  -  -  -  -  -  -
D | -  -  -  -  -  -  -  -  -  -
E | -  -  -  -  -  -  -  -  -  -
F | -  -  -  -  -  -  -  -  -  -
G | -  -  -  -  -  -  -  -  -  -
H | -  -  -  -  -  -  -  -  -  -
I | -  -  -  -  -  -  -  -  -  -
J | -  -  -  -  -  -  -  -  -  -

O inimigo acertou 0 posições
Você acertou 0 posições

Legenda:
p: Porta-Aviões
t: Navio-Tanque
c: Contratorpedeiro
s: Submarino
*: Falha
x: Acerto
-: Posição livre

Meu Tabuleiro
  1  2  3  4  5  6  7  8  9  10
A | p1 t1 t2 c1 c2 c3 s1 s2 s3 s4
B | p1 t1 t2 c1 c2 c3 s1 s2 s3 s4
C | p1 t1 t2 c1 c2 c3 -  -  -
D | p1 t1 t2 -  -  -  -  -  -
E | p1 -  -  -  -  -  -  -  -
F | -  -  -  -  -  -  -  -
G | -  -  -  -  -  -  -  -
H | -  -  -  -  -  -  -  -
I | -  -  -  -  -  -  -  -
J | -  -  -  -  -  -  -  -

Jogo começou!
Escolha uma linha (A-J):

```

Em caso de erro, marcado no tabuleiro inimigo posição já escolhida

```

Jogo começou!
Escolha uma linha (A-J): a
Escolha uma coluna (1-10): 1
Resultado da sua ultima jogada: Erro
Pressione ENTER para continuar!

Tabuleiro Server
  1  2  3  4  5  6  7  8  9  10
A | *  -  -  -  -  -  -  -  -  -
B | -  -  -  -  -  -  -  -  -  -
C | -  -  -  -  -  -  -  -  -  -
D | -  -  -  -  -  -  -  -  -  -
E | -  -  -  -  -  -  -  -  -  -
F | -  -  -  -  -  -  -  -  -  -
G | -  -  -  -  -  -  -  -  -  -
H | -  -  -  -  -  -  -  -  -  -
I | -  -  -  -  -  -  -  -  -  -
J | -  -  -  -  -  -  -  -  -  -

O inimigo acertou 0 posições
Você acertou 0 posições

Legenda:
p: Porta-Aviões
t: Navio-Tanque
c: Contratorpedeiro
s: Submarino
*: Falha
x: Acerto
-: Posição livre

Meu Tabuleiro
  1  2  3  4  5  6  7  8  9  10
A | p1 t1 t2 c1 c2 c3 s1 s2 s3 s4
B | p1 t1 t2 c1 c2 c3 s1 s2 s3 s4
C | p1 t1 t2 c1 c2 c3 -  -  -
D | p1 t1 t2 -  -  -  -  -  -
E | p1 -  -  -  -  -  -  -  -
F | -  -  -  -  -  -  -  -
G | -  -  -  -  -  -  -  -
H | -  -  -  -  -  -  -  -
I | -  -  -  -  -  -  -  -
J | -  -  -  -  -  -  -  -

Resultado da ultima jogada inimiga: Erro
Pressione ENTER para continuar!

```

Tela final após fim de jogo.

```
Resultado da ultima jogada inimiga: Acerto
Pressione ENTER para continuar!

Tabuleiro Server                               Meu Tabuleiro
  1  2  3  4  5  6  7  8  9 10                1  2  3  4  5  6  7  8  9 10
A | * | * | * | * | x | x | x | x | * | * | A | x | x | x | x | x | x | x | x | x |
B | x | * | * | x | x | x | * | * | * | * | B | x | x | x | x | x | x | x | x | x |
C | x | * | * | x | x | x | x | x | x | * | C | x | x | x | x | x | x | x | x | * |
D | x | * | * | x | x | x | x | x | * | * | D | x | x | * | * | * | * | * | * | - |
E | x | * | x | x | x | * | * | * | * | * | E | * | * | * | * | * | - | * | - | * |
F | * | * | * | * | * | * | * | * | * | * | F | * | * | * | - | * | * | - | * | * |
G | * | * | * | * | * | * | * | x | x | * | G | * | * | * | * | - | - | - | - | * |
H | * | * | * | - | - | - | - | - | - | - | H | - | * | * | * | - | * | - | - | - |
I | - | - | - | - | - | - | - | - | - | - | I | - | - | - | * | - | - | - | * | * |
J | - | - | - | - | - | - | - | - | - | - | J | - | * | * | * | * | - | - | - | - | * |

O inimigo tem 20 acerto(s)
Você tem 27 acerto(s)

Legenda:
p: Porta-Aviões
t: Navio-Tanque
c: Contratorpedeiro
s: Submarino
*: Falha
x: Acerto
-: Posição livre
```

2. Sistemas de Preços

2.1. Introdução ao problema

O programa desenvolvido tem como objetivo a prática do uso de bibliotecas socket utilizando o protocolo UDP. Essa é uma aplicação de um sistema de preços que tem como funcionalidade receber uma série de preços de combustíveis em determinados postos e suas localizações. Com a informação dos preços e da localização deve ser determinado o menor preço de uma região.

Foram implementados dois programas (client-udp.py e server-udp.py) que recebem os parâmetros abaixo:

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

O cliente deve se conectar no servidor de acordo com o endereço IP definido e a porta estabelecida como parâmetro. O servidor vai receber uma informação por vez e tratá-la.

2.2. Implementação

O código dos programas foram implementados em Python e foi utilizada a biblioteca socket. Utilizado o parâmetro adequado para sockets UDP.

O usuário transmite as mensagens server-client na seguinte ordem:

Se escolhido o tipo de mensagem 'D'(dados):

- Tipo de mensagem = D
- Tipo de combustível (0-diesel, 1-álcool, 2-gasolina)
- Preço do combustível
- Latitude do posto
- Longitude do posto

Os dados são colocados em um json (exemplo abaixo) para ser enviado para o server.

```
1  {"combustivel": 0, "preco": 50, "coordenada": [5, 5]}
2  {"combustivel": 0, "preco": 40, "coordenada": [9, 9]}
3  {"combustivel": 0, "preco": 35, "coordenada": [5, 5]}
4  {"combustivel": 1, "preco": 10, "coordenada": [10, 10]}
5  {"combustivel": 1, "preco": 15, "coordenada": [16, 16]}
6  {"combustivel": 1, "preco": 11, "coordenada": [22, 22]}
7  {"combustivel": 2, "preco": 60, "coordenada": [7, 7]}
8  {"combustivel": 2, "preco": 66, "coordenada": [13, 13]}
9  {"combustivel": 2, "preco": 51, "coordenada": [25, 25]}
```

Se escolhido o tipo de mensagem 'P'(pesquisa):

- Tipo de mensagem = P
- Tipo de combustível (0-diesel, 1-álcool, 2-gasolina)
- O raio da busca

- Latitude do centro da busca
- Longitude do centro da busca

Retornando o menor valor de um combustível em uma determinada região.

O algoritmo utilizado para descobrir qual posto de gasolina tem o menor preço em uma determinada região demarcada. Ele compara o raio da busca com a distância das coordenadas do centro e as coordenadas dos postos.

```

26 def isInside(coord, center, radius):
27
28     lat = coord[0]
29     lon = coord[1]
30     x = center[0]
31     y = center[1]
32
33     # Compara o raio com a distancia do centro com a coordenada
34     if ((x - lat) * (x - lat) +
35         (y - lon) * (y - lon) <= radius * radius):
36         return True;
37     else:
38         return False;
39

```

2.3. Testes

Inicialmente inserimos uma porta no servidor para a comunicação

```

C:\Users\luca-\Desktop\TrabalhoRedesI\sistema de precos>python server_udp.py
Insira a porta para conexão: █

```

Servidor conectado

```

C:\Users\luca-\Desktop\TrabalhoRedesI\sistema de precos>python server_udp.py
Insira a porta para conexão: 3333
Servidor iniciado. Aguardando conexões...
Host: 192.168.15.15
Porta: 3333

```

Agora, no cliente inserir o endereço IP e porta para conectar com o servidor

```

C:\Users\luca-\Desktop\TrabalhoRedesI\sistema de precos>python client_udp.py

Sistema de Preços

Insira o IP do servidor: █

```


Conexão servidor-cliente estabelecida pelo protocolo UDP

```
C:\Users\luca-\Desktop\TrabalhoRedesI\sistema de precos>python server_udp.py
Insira a porta para conexão: 3333
Servidor iniciado. Aguardando conexões...
Host: 192.168.15.15
Porta: 3333
192.168.15.15 conectado. Preparando...
```

No cliente iniciamos a inserção de dados dos postos de combustível

```
C:\Users\luca-\Desktop\TrabalhoRedesI\sistema de precos>python client_udp.py

Sistema de Preços

Insira o IP do servidor: 192.168.15.15
Insira a porta para conexão: 3333

Qual o tipo de mensagem? (Dados = D ou Pesquisa = P):
```

Inserimos todos os dados

```
C:\Users\luca-\Desktop\TrabalhoRedesI\sistema de precos>python client_udp.py

Sistema de Preços

Insira o IP do servidor: 192.168.15.15
Insira a porta para conexão: 3333

Qual o tipo de mensagem? (Dados = D ou Pesquisa = P): d

Qual o tipo de combustível? (0-diesel, 1-álcool, 2-gasolina): 0

Qual o preço do combustível? 50

Qual a latitude do posto de combustível? 5

Qual a longitude do posto de combustível? 5
Confirmado: 0

Qual o tipo de mensagem? (Dados = D ou Pesquisa = P):
```

Servidor confirma recebimento da mensagem

```
C:\Users\luca-\Desktop\TrabalhoRedesI\sistema de precos>python server_udp.py
Insira a porta para conexão: 3333
Servidor iniciado. Aguardando conexões...
Host: 192.168.15.15
Porta: 3333
192.168.15.15 conectado. Preparando...
Mensagem 0 recebida
```

Inserindo mais dados

```
C:\Users\luca-\Desktop\TrabalhoRedesI\sistema de precos>python client_udp.py

Sistema de Preços

Insira o IP do servidor: 192.168.15.15
Insira a porta para conexão: 3333

Qual o tipo de mensagem? (Dados = D ou Pesquisa = P): d
Qual o tipo de combustível? (0-diesel, 1-álcool, 2-gasolina): 0
Qual o preço do combustível? 50
Qual a latitude do posto de combustível? 5
Qual a longitude do posto de combustível? 5
Confirmado: 0

Qual o tipo de mensagem? (Dados = D ou Pesquisa = P): d
Qual o tipo de combustível? (0-diesel, 1-álcool, 2-gasolina): 0
Qual o preço do combustível? 40
Qual a latitude do posto de combustível? 9
Qual a longitude do posto de combustível? 9
Confirmado: 1

Qual o tipo de mensagem? (Dados = D ou Pesquisa = P): 0
Tipo inválido! (Digite D ou P)

Qual o tipo de mensagem? (Dados = D ou Pesquisa = P): d
Qual o tipo de combustível? (0-diesel, 1-álcool, 2-gasolina): 0
Qual o preço do combustível? 35
Qual a latitude do posto de combustível? 5
Qual a longitude do posto de combustível? 5
Confirmado: 2
```

Todos os dados foram recebidos e confirmados

```
C:\Users\luca-\Desktop\TrabalhoRedesI\sistema de precos>python server_udp.py
Insira a porta para conexão: 3333
Servidor iniciado. Aguardando conexões...
Host: 192.168.15.15
Porta: 3333
192.168.15.15 conectado. Preparando...
Mensagem 0 recebida
Mensagem 1 recebida
Mensagem 2 recebida
```

Pesquisa do menor valor

```
Qual o tipo de mensagem? (Dados = D ou Pesquisa = P): p
Qual o tipo de combustível? (0-diesel, 1-álcool, 2-gasolina): 0
Qual é o raio da busca? 100
Qual a latitude do centro da busca? 5
Qual a longitude do centro da busca? 5
Retransmissão...
Menor preço encontrado: 35
Qual o tipo de mensagem? (Dados = D ou Pesquisa = P):
```

3. Conclusão

Com a implementação dos problemas acima e com a utilização dos protocolos TCP e UDP podemos observar algumas das diferenças entre eles de forma prática juntamente com o que foi apresentado nas aulas teóricas.

O protocolo TCP se caracteriza principalmente por ter uma transferência confiável de dados garantindo a integridade dos dados entregues, na aplicação em TCP não observamos falhas na troca de dados.

O protocolo UDP se caracteriza por ter um transporte não confiável, segmentos UDP podem ser perdidos ou entregues fora de ordem. Na aplicação apresentada é possível observar em alguns casos uma falha no envio dos dados, necessitando com isso a confirmação do recebimento de mensagens.

4. Referências

Socket — Low-level networking interface <<https://docs.python.org/3/library/socket.html>>. Acessado em 05/2020

Find if a point lies inside a Circle <<https://www.geeksforgeeks.org/find-if-a-point-lies-inside-or-on-circle/>>. Acessado em 05/2020

Programação com Sockets em Python <<http://ww2.inf.ufg.br/ricardo/python/programacao-sockets.html>>. Acessado em 05/2020