REDES DE COMPUTADORES

Nome: Maximilliam Alex de Araujo

DOCUMENTAÇÃO PARA TRABALHO PRÁTICO 1

PROBLEMA

Desenvolver uma jogo de Batalha Naval que funcione em cima de uma comunicação de cliente/ servidor, utilizando protocolo TCP e a biblioteca Socket.
O principal problema é entender e implementar uma comunicação socket TCP.
O segundo problema é entender a lógica de um jogo de batalha naval em que uma pessoa joga contra o computador. Isso envolve entender o jogo, desenvolver a inteligência da máquina e adaptar tudo em cima da comunicação do cliente/ servidor.

SOLUÇÃO IMPLEMENTADA

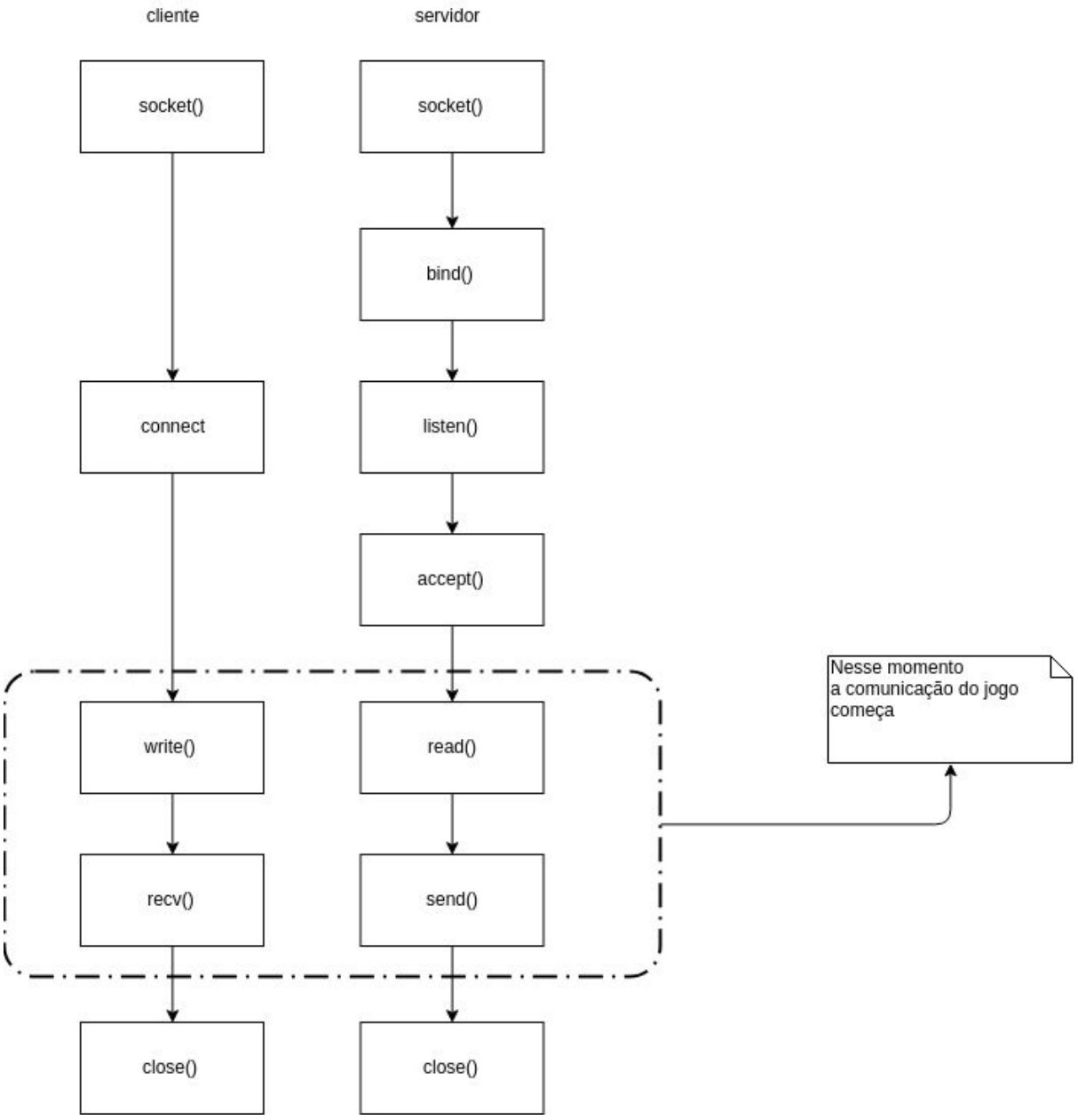
Através da API Socket utilizando a liguangem de programação C, foram desenvolvidos três programas básicos.

servidor.c: responsável por criar um socket da família ipv6 com parâmetros que permitem escutar um cliente com endereço ipv4 ou ipv6.

cliente.c: responsável por criar uma socket com base no endereço fornecido, classificando-o como ipv4 ou ipv6 para conectar no servidor.

tabuleiro.c: concentra a maior parte da lógica para que o jogo funcione.

FLUXO BÁSICO DO PROGRAMA



TRATAMENTO PARA IPV4 E IPV6

No servidor a função socket() recebeu como parâmetro:

AF_INET6: Utilizado para receber ipv6

SOCK_STREAM: Utilizado para TCP/IP

Dentro da estrutura do socke do servidort:

self.sin6_addr = in6addr_any (Trecho de código)

self: é a estrutura do socket do próprio servidor
sin6_addr: parâmetro para endereçamento ipv6
in6addr_any: escolhido para que o socket consiga escutar tanto ipv6 quanto ipv4.

No cliente foi utilizado a estrutura addrinfo para obter as informações para construção do socket com base no tipo de ip (v4 ou v6).

A função inet_pton () foi usada para converter a forma de texto do endereço para binário

Em seguida getaddrinfo() para obter informações sobre o servidor e com base nas informações retornadas o socket é criado e partir daqui é seguido o fluxo de conecção com o servidor.

FUNÇÕES

tabuleiro.c

inicializa_tabuleiro(): recebe uma matriz por referência e preenche com zeros.

orientacao(:) retorna 0 ou 1 aleatoriamente. Zero indica que a é para colocar uma peça na vertical e Um na horizontal.

casa_inicial(): Gera um número aleatório entre 0 e um máximo definido pela função que chamar.

verifica_disponibilidade(): Com base na orientação, na posição sorteada e no tamanho da peça, verifica se é possível colocar a peça.

preencheTNavio(): Coloca a peça com a orientação e posição na matriz.

preencher_navio(): Função responsável por verificar o tipo de peça, a quantidade de peças e chamar a funções de aleatoriedade para tentar colocar as peças no tabuleiro.

exibir_tabuleiro(): função imprime um tabuleira na tela

carregar_tabuleiro_de_um_arquivo(): função responsável por carregar um tabuleiro para uma matriz através de um arquivo.

inicializa_mapa_de_tiros(): Função gera uma matriz com simbolos para ajudar o jogador a mapear as coordenadas que já foram jogadas.

exibir_mapa(): Imprime um mapa na tela

traduzir_tiro(): função traduz uma coordenada literal para coordenadas de matriz em c.

c_letra(): Pega uma letra e retorna qual coordenada essa letra representa.

letra_c(): Pega um número e retorna qual letra este número representa.

LÓGICA DO JOGO

Todas as informações que o cliente eo servidor precisam são trafegadas em mensagens únicas.

Exemplo:
Mensagem enviada do servidor para cliente:
EA1
E: indica que o cliente errou o tiro
A1: coordenada do tiro do servidor

Cliente envia para servidor:
A10A
A10: coordenada do tiro do cliente
A: indica que o servidor acertou o último tiro

É definida uma quantidade de peças no tabuleiro e toda vez que o servidor ou o cliente acertam uma peça um contador é reduzido. Quando esse chega a zero, o jogo termina.

CONSIDERAÇÕES FINAIS

Não foi implementado uma lógica para que o servidor atire ao redor de um acerto.
Para fins de teste, o tabuleiro é preenchido com 28 peças, sendo 1: 55555, 2: 4444, 3: 333, 3:22. Isso porque a função de aleatoriedade utiliza um clock e um time(NULL) com semente para um rand() do C, com um módulo 10. Ainda sim a função demonstrou certos padrões de números gerados e para um tabuleiro totalmente preenchido era necessário uma grande quantidade de tempo para encontrar lugar para todas as peças.

TESTES

```
maxaraujo@maxaraujo-Inspiron-5423:~/Documentos/UFMG/REDES/TP/batalh
a-naval-socket-2.0/batalha-naval-socket$ ./servidor 9999
- 1 2 3 4 5 6 7 8 9 10
A 4 0 0 0 0 0 0 0 0 0
B 4 3 0 0 0 0 0 0 0 0
C 4 3 2 2 0 0 0 0 0 0
D 4 3 0 4 0 0 0 0 0 0
E 0 0 0 4 3 0 0 0 0 0
F 0 0 0 4 3 5 5 5 5 5
G 0 0 0 4 3 0 3 0 0 0
H 0 0 0 0 0 0 3 2 0 0
I 0 0 0 0 0 0 3 2 2 0
J 0 0 0 0 0 0 0 0 2 0
Cliente conectado.
 Aguardando por mensagens do cliente...
```

28 posições:

```
maxaraujo@maxaraujo-Inspiron-5423: ~/Docur
- 1 2 3 4 5 6 7 8 9 10
A * ~ ~ ~ ~ ~ ~ ~ ~ ~
B * * ~ ~ ~ ~ ~ ~ ~ ~
C * * * * ~ ~ ~ ~ ~ ~
D * * ~ * ~ ~ ~ ~ ~ ~
E ~ ~ ~ * ~ ~ ~ ~ ~ ~
F ~ ~ ~ * * * * * * *
G ~ ~ ~ * * ~ * ~ ~ ~
H ~ ~ ~ ~ ~ ~ * * ~ ~
I ~ ~ ~ ~ ~ ~ * * * ~
J ~ ~ ~ ~ ~ ~ ~ ~ * ~
Atire:
E5
Acertou!
Você venceu!
```

make e servidor rodando

obs.: Matriz gerada e exibida no servidor para fins de teste

```
maxaraujo@maxaraujo-Inspiron-5423:~/Documentos/UFMG/REDES/TP/batalh
a-naval-socket-2.0/batalha-naval-socket$ make
gcc     -c      servidor.c
servidor.c: In function ‘main’:
servidor.c:78:17: warning: implicit declaration of function ‘read’
[-Wimplicit-function-declaration]
                read(client_s, msg_read, MAXBUF);
                ^
servidor.c:107:9: warning: implicit declaration of function ‘close’
 [-Wimplicit-function-declaration]
         close(client_s);
         ^
gcc     servidor.o      tabuleiro.o     -o      servidor
maxaraujo@maxaraujo-Inspiron-5423:~/Documentos/UFMG/REDES/TP/batalh
a-naval-socket-2.0/batalha-naval-socket$ ./servidor 9999
- 1 2 3 4 5 6 7 8 9 10
A 4 4 4 4 0 0 0 0 0 0
B 0 3 0 0 0 0 0 0 0 0
C 0 3 2 0 0 0 0 0 0 0
D 0 3 2 2 0 0 0 0 0 0
E 0 0 0 2 5 0 0 0 0 0
F 0 0 0 0 5 4 4 4 4 0
G 0 0 0 0 5 0 3 0 0 0
H 0 0 0 0 5 0 3 3 0 0
I 0 0 0 0 5 0 3 3 2 0
J 0 0 0 0 0 0 0 3 2 0
Cliente conectado.
 Aguardando por mensagens do cliente...
```

Conectando com ipv4

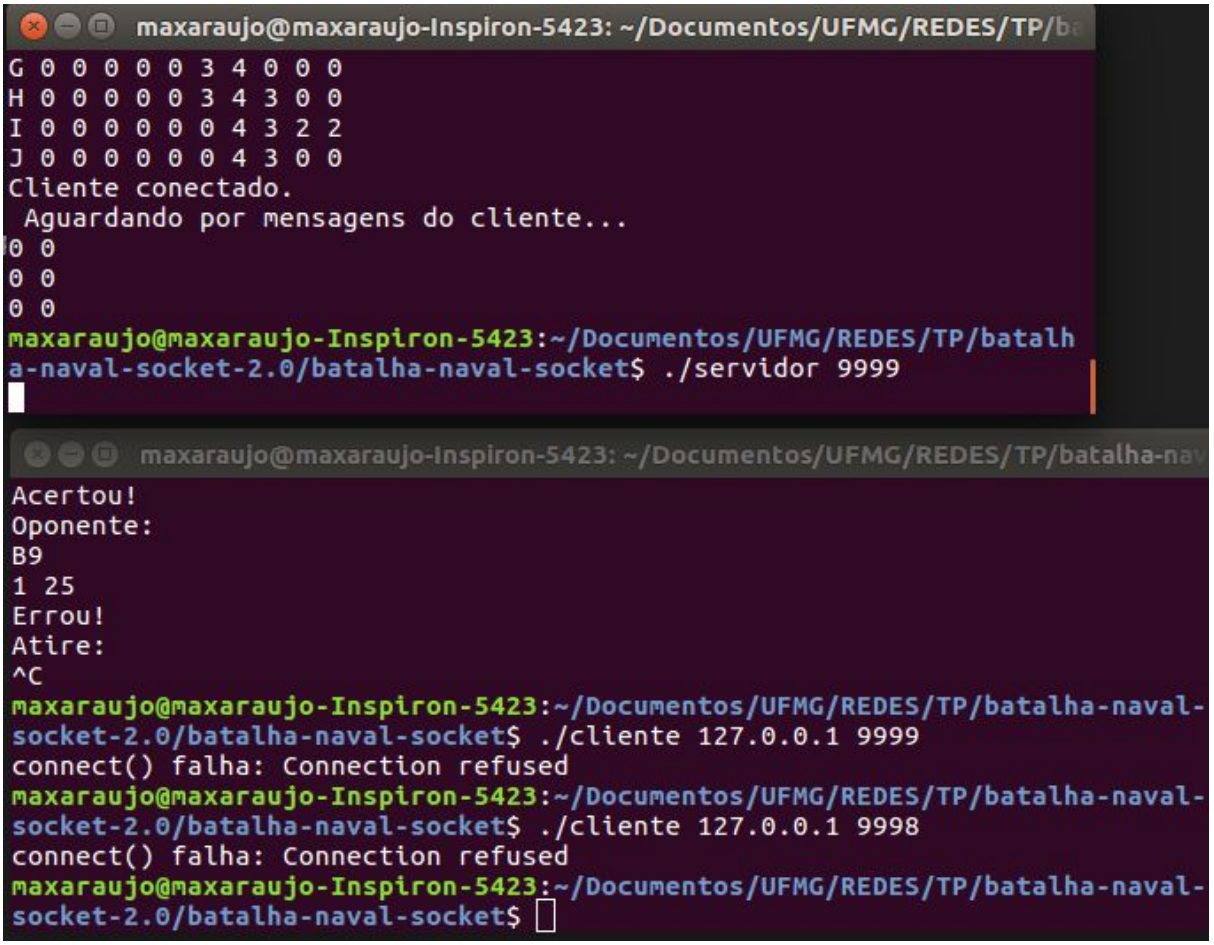
```
maxaraujo@maxaraujo-Inspiron-5423:~/Documentos/UFMG/REDES/TP/batalha-naval-
socket-2.0/batalha-naval-socket$ ./cliente 127.0.0.1 9999
Conectado


Atire:
A1
Acertou!
Oponente:
B9
1 25
Errou!
Atire:
```

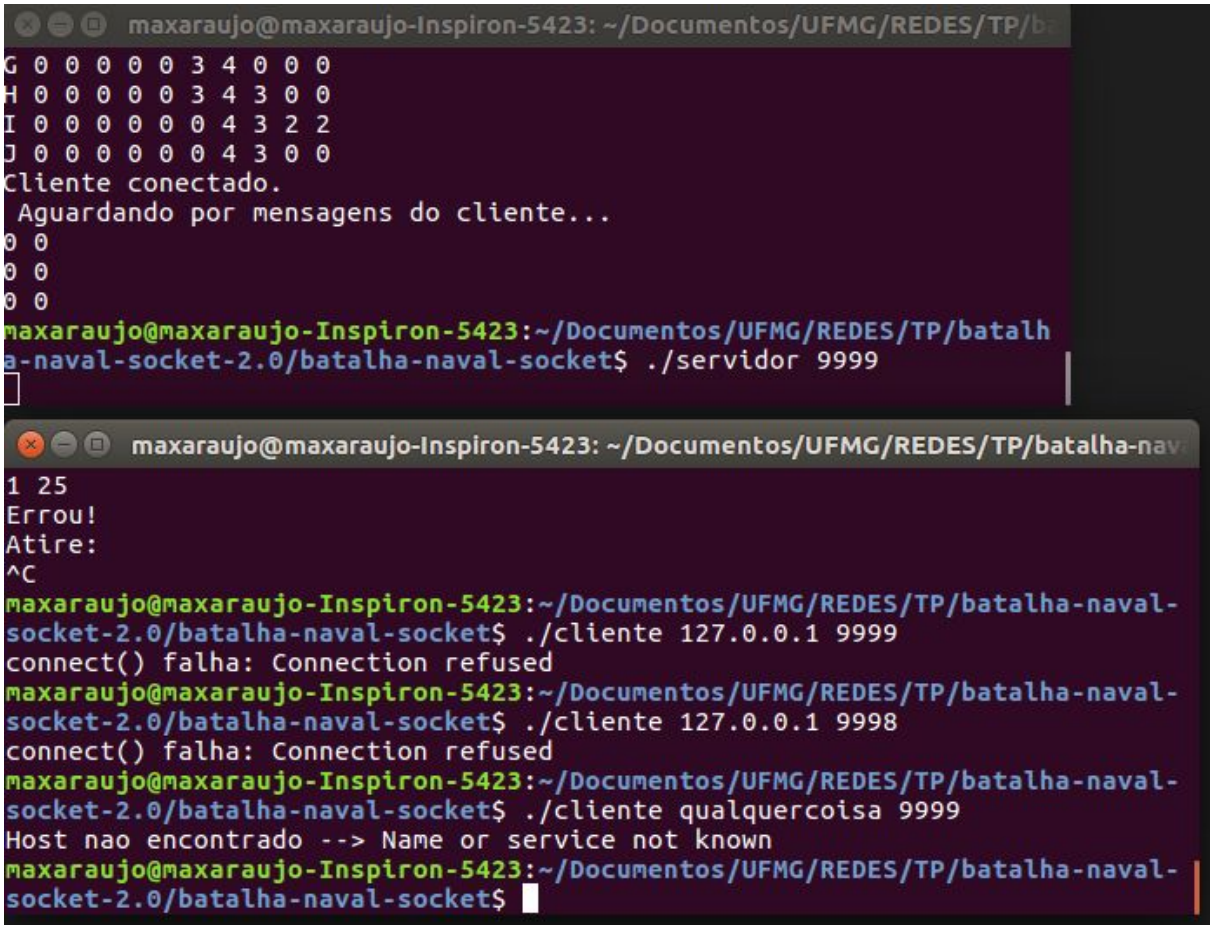
Servidor parado

```
maxaraujo@maxaraujo-Inspiron-5423:~/Documentos/UFMG/REDES/TP/batalha-naval-
socket-2.0/batalha-naval-socket$ ./cliente 127.0.0.1 9999
connect() falha: Connection refused
maxaraujo@maxaraujo-Inspiron-5423:~/Documentos/UFMG/REDES/TP/batalha-naval-
socket-2.0/batalha-naval-socket$
```

Porta errada



Host errado



BIBLIOGRAFIA

https://tools.ietf.org/html/rfc3493

https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_71/rzab6/ip6scen.htm

https://www.systutorials.com/docs/linux/man/3-inet_pton/