UNIVERSIDADE FEDERAL DE MINAS GERAIS

REDES DE COMPUTADOR -TRABALHO PRÁTICO

2

GABRIEL SOUZA DE OLIVEIRA

1. INTRODUÇÃO

O advento da tecnologia Smart Grid tem transformado radicalmente a indústria de energia elétrica, possibilitando a integração de fontes renováveis, otimização da distribuição e melhoria na eficiência energética. Este relatório documenta o desenvolvimento de um sistema cliente-servidor que simula as operações vitais de uma Smart Grid, permitindo o controle, monitoramento e interação com sensores em tempo real.

2. ARQUITETURA DO SERVIDOR (MTU)

O sistema é baseado em uma arquitetura servidor-cliente e servidor-servidor (p2p) , utilizando a comunicação via TCP/IP. O servidor (MTU) é responsável pelo processamento das solicitações recebidas do cliente (RTU) e pela gestão dos sensores. O cliente, por sua vez, interage com o usuário e envia comandos para o servidor.

- **Limitações do servidor**

1- O servidor só recebe o comando “kill” teclado.

```
////////////////////////////////////////////////////////////////////// recebi comando do teclado no server
if (!(FD_ISSET(s_p2p, &readfds_p2p)) && !(FD_ISSET(s, &readfds)) && FD_ISSET(STDIN_FILENO, &read_input))
{

    memset(palavra_entrada_tec, 0, BUFSZ);
    fgets(palavra_entrada_tec, BUFSZ, stdin);
    memset(mensagem, 0, BUFSZ);

    if (strcmp(palavra_entrada_tec, "kill\n") == 0)
    {
```

3- O servidor só suporta a conexão com 1 servidor.

```
if (limite_de_serv == 1)
{
    break;
}
```

2- O limite de clientes na fila é de 15.

```
////////////// listen() /////////// fica aguardando ate 15 clientes na fila
if (0 != listen(s, 15))
{
    logexit("listen");
}
```

- **Funcionalidades do servidor-cliente (comunicação com cliente)**

Após o recebimento da mensagem do “client” via comando recv() , como mostrado abaixo:

```
////////////// recv() ////////////////// server recebe dado do cliente
for (int i = 0; i < clientmax; i++)
{
    sd = client_socket[i];

    if (FD_ISSET(sd, &readfds))
    {

        char buf[BUFSZ];
        int atr_1 = 0;
        memset(buf, 0, BUFSZ);
        size_t count = recv(sd, buf, BUFSZ - 1, 0); // recebi uma mensagem do client
```

verifico qual comando eu recebi do cliente, podendo ser:

1- REQ_LS (show localmaxsensor)

```
if (strncmp("REQ_LS", buf, strlen("REQ_LS")) == 0)
{ ////////////// recebi REQ_LS do cliente
```

1.1-Retorna para o cliente.

```
show localmaxsensor
Local 4 sensor 5: 1073 (1154 93)
```

2- REQ_LP (show localpotency)

```
if (strncmp("REQ_LP", buf, strlen("REQ_LP")) == 0)
{ ////////////// recebi REQ_LP do cliente
```

2.1-Retorna para o cliente.

```
show localpotency
Local 4 potency: 3263
```

3- REQ_ES (show externalmaxsensor)

```
if (strncmp("REQ_ES", buf, strlen("REQ_ES")) == 0)
{ ////////////// recebi REQ_ES do cliente
```

3.1-Retorna para o cliente.

```
show externalmaxsensor
External 6 sensor 4: 956 (1275 75)
```

4- REQ_EP (show externalpotency)

```
if (strncmp("REQ_EP", buf, strlen("REQ_EP")) == 0)
{ ////////////// recebi REQ_EP do cliente
```

4.1-Retorna para o cliente.

```
show externalpotency
External 6 potency: 4040
```

5- REQ_MS (show globalmaxsensor)

```
if (strncmp("REQ_MS", buf, strlen("REQ_MS")) == 0)
{ ////////////// recebi REQ_MS do cliente
```

5.1-Retorna para o cliente.

```
show globalmaxsensor
global 4 sensor: 5: 1073 (1154 93)
```

6- REQ_DC (Comando inválido ou kill do cliente)

```
if (strncmp(buf, "REQ_DC(", 7) == 0)
{ ////////////// recebi kill do cliente
```

6.1- Fecha o terminal do cliente.

```
kill
Successful disconnect
gabriel89067@DESKTOP-7OHVILA:~/programas
$
```

7- REQ_MN (show globalmaxnetwork)

```
if (strncmp("REQ_MN", buf, strlen("REQ_MN")) == 0)
{ ////////////// recebi REQ_MN do cliente
```

7.1- Retorna para o cliente.

```
show globalmaxnetwork
global 6 potency: 4040
```

- **Funcionalidades do servidor-servidor (comunicação com o servidor_p2p)**

Após o recebimento da mensagem do “server” via comando recv() , como mostrado abaixo:

```
////////////// recv() /////////////// server recebe dado do p2p

sd_p2p = identificador_serv_externo_int;

if (FD_ISSET(sd_p2p, &readfds_p2p))
{
    char value_rece[BUFSZ];
    char buf_p2p[BUFSZ];
    memset(buf_p2p, 0, BUFSZ);
    memset(value_rece, 0, BUFSZ);
    size_t count_p2p = recv(sd_p2p, buf_p2p, BUFSZ - 1, 0); // recebi uma mensagem do server_p2p
```

verifico qual comando eu recebi do server_p2p, podendo ser:

1. REQ_LS (kill do server)

```
if (strncmp(buf_p2p, "REQ_DCPEER", 9) == 0)
{ // recebi "REQ_DCPEER" do server_p2p
```

1.1-Retorna para o server_p2p.

```
show localmaxsensor
Local 4 sensor 5: 1073 (1154 93)
```

2. REQ_ES (show externalmaxsensor)

```
if (strcmp(buf_p2p, "REQ_ES") == 0)
{ // recebi req_es do servidor p2p
```

2.1-Retorna para o server_p2p.

```
show localpotency
Local 4 potency: 3263
```

3. REQ_EP (show externalpotency)

```
if (strcmp(buf_p2p, "REQ_EP") == 0) ////////////////////////////
{                              // recebi REQ_EP do servidor p2p
```

3.1-Retorna para o server_p2p.

- **Dados armazenados no servidor**

Armazeno no servidor, uma tabela com 10 sensores e suas características geradas aleatórias como mostra a figura:

```c
struct equipamento
{
    int ID;
    int CORR;
    int TENS;
    int EFI;
    int POT;
    int POT_UTIL;
};
```

```c
struct equipamento equipamentos[100];

srand(time(NULL));
for (int i = 0; i < 10; i++)
{ // inicializa os sensores no servidor com valores aleatorios

    equipamentos[i].ID = i;
    equipamentos[i].EFI = rand() % 100;
    equipamentos[i].POT = rand() % 1500;
    equipamentos[i].POT_UTIL = (equipamentos[i].EFI * equipamentos[i].POT) / 100;
}
```

3. ARQUITETURA DO CLIENTE (RTU)

O sistema é baseado em uma arquitetura cliente-servidor, utilizando a comunicação via TCP/IP. O cliente (RTU) é responsável pelo envio das solicitações para o servidor (MTU) e pela gestão dos inputs do teclado.

- **Obrigações do cliente**

1- O cliente recebe comandos do teclado e os manipula para enviar para o servidor.

```
gabriel89067@DESKTOP-7OHVILA:~/programas
$ ./client 127.0.0.1 90903
New ID: 5
show localmaxsensor
Local 4 sensor 5: 1073 (1154 93)
show localpotency
Local 4 potency: 3263
show externalmaxsensor
External 6 sensor 4: 956 (1275 75)
```

- **Funcionalidades do cliente**

Após o recebimento da mensagem via teclado , como mostrado abaixo:

```c
memset(pala_in, 0, 50);
fgets(pala_in, 50, stdin);
pala_conhecid = 0;
```

verifico qual comando eu recebi do cliente, podendo ser:

1- show localmaxsensor

```c
if (strncmp(pala_1_1, pala_in, strlen(pala_1_1)) == 0)
{ // digitei show localmaxsensor
```

Comando que requisita ao servidor via código “INS_REQ” a instalação de um sensor por meio de parâmetros passados na tela .Sua sintaxe é da forma: `show localmaxsensor` .
Tratamento de 1 possível erro :

1.1- erro de comando ,envia ao servidor o código “kill” e fecha o terminal do cliente

```
show localmaxsenr
Successful disconnect
gabriel89067@DESKTOP-7OHVILA:~/programas
$
```

2- show localpotency

```
if (strncmp(pala_1_2, pala_in, strlen(pala_1_2)) == 0)
{ // digitei show localpotency
```

Comando que requisita ao servidor via código “INS_REQ” a instalação de um sensor por meio de um arquivo .txt .Sua sintaxe é da forma: `show localpotency` .
Tratamento de 1 possível erro :

2.1- erro de comando ,envia ao servidor o código “kill” e fecha o terminal do cliente

```
show localmaxsenr
Successful disconnect
gabriel89067@DESKTOP-7OHVILA:~/programas
$
```

3- show externalmaxsensor

```
if (strncmp(pala_2, pala_in, strlen(pala_2)) == 0)
{ // digitei show externalmaxsensor
```

Comando que requisita ao servidor via código “REM_REQ” a remoção de um sensor por meio de um parâmetro passado na tela.Sua sintaxe é da forma: `show externalmaxsensor` .
Tratamento de 1 possível erro :

3.1-erro de comando ,envia ao servidor o código “kill” e fecha o terminal do cliente

```
show localmaxsenr
Successful disconnect
gabriel89067@DESKTOP-7OHVILA:~/programas
$
```

4- show externalpotency

```
if (strncmp(pala_3_1, pala_in, strlen(pala_3_1)) == 0)
{ // digitei show externalpotency
```

Comando que requisita ao servidor via código “SEN_REQ” a visualização de um sensor por meio de um parâmetro passado na tela .Sua sintaxe é da forma: `show externalpotency` .
Tratamento de 1 possível erro :

4.1- erro de comando ,envia ao servidor o código “kill” e fecha o terminal do cliente

```
show localmaxsenr
Successful disconnect
gabriel89067@DESKTOP-7OHVILA:~/programas
$
```

5- show globalmaxsensor

```
if (strncmp(pala_3_2, pala_in, strlen(pala_3_2)) == 0)
{ // digitei show globalmaxsensor
```

Comando que requisita ao servidor via código “VAL_REQ” a visualização de todos os sensor por meio de um parâmetro passado na tela .Sua sintaxe é da forma: `show globalmaxsensor` .
Tratamento de 1 possível erro :

5.1-erro de comando ,envia ao servidor o código “kill” e fecha o terminal do cliente

```
show localmaxsenr
Successful disconnect
gabriel89067@DESKTOP-7OHVILA:~/programas
$
```

6- show globalmaxnetwork

```
if (strncmp(pala_4_1, pala_in, strlen(pala_4_1)) == 0)
{ // digitei show globalmaxnetwork
```

Comando que requisita ao servidor via código “CH_REQ” a modificação de um sensor por meio de parâmetros passados na tela .Sua sintaxe é da forma: `show globalmaxnetwork`

Tratamento de 1 possível erro :

6.1-erro de comando ,envia ao servidor o código “kill” e fecha o terminal do cliente

```
show localmaxsenr
Successful disconnect
gabriel189067@DESKTOP-7OHVILA:~/programas
$
```

7- kill

```
if (strncmp("kill", pala_in, 4) == 0)
```

Comando que requisita ao servidor via código “kill” a instalação de um sensor por meio de um parâmetro passado na tela .Sua sintaxe é da forma: `kill`

4. EXEMPLO DE EXECUÇÃO

```
gabriel189067@DESKTOP-7OHVILA:~/programas
$ ./server 127.0.0.1 90900 90901
No peer found, starting to listen..
Peer 5 connected
New Peer ID: 4
REQ_ADD
Client 6 added
REQ_LS
REQ_ES
RES_ES 4 7: 1290 (1317 98)
REQ_LP
REQ_EP
REQ_EP
RES_EP 4 5875
REQ_MS
RES_ES 5 9: 1075 (1182 91)
RES_ES 4 7: 1290 (1317 98)
REQ_EP
RES_EP 4 5875
REQ_DC(6)
Client 6 removed
REQ_ADD
Client 6 added
Peer 5 disconnected
kill
gabriel189067@DESKTOP-7OHVILA:~/programas
$
```

```
gabriel189067@DESKTOP-7OHVILA:~/programas
$ ./server 127.0.0.1 90900 90902
New Peer ID: 5
Peer 4 connected
REQ_ADD
Client 5 added
REQ_LS
RES_ES 5 9: 1075 (1182 91)
REQ_ES
REQ_LP
REQ_EP
RES_EP 5 3532
REQ_EP
RES_ES 5 9: 1075 (1182 91)
REQ_MS
RES_ES 4 7: 1290 (1317 98)
REQ_EP
RES_EP 5 3532
REQ_DC(5)
Client 5 removed
REQ_ADD
Client 5 added
kill
Successful disconnect
Peer 5 disconnected
gabriel189067@DESKTOP-7OHVILA:~/programas
$
```

```
gabriel189067@DESKTOP-7OHVILA:~/programas
$ ./client 127.0.0.1 90901
New ID: 6
show localmaxsensor
Local 4 sensor 7: 1290 (1317 98)
show externalmaxsensor
External 5 sensor 9: 1075 (1182 91)
show localpotency
Local 4 potency: 5875
show externalpotency
External 5 potency: 3532
show globalmaxsensor
global 4 sensor: 7: 1290 (1317 98)
show globalmaxnetwork
global 4 potency: 5875
kill
Successful disconnect
gabriel189067@DESKTOP-7OHVILA:~/programas
$ ./client 127.0.0.1 90901
New ID: 6
Successful disconnect
gabriel189067@DESKTOP-7OHVILA:~/programas
$
```

```
gabriel189067@DESKTOP-7OHVILA:~/programa
s$ ./client 127.0.0.1 90902
New ID: 5
show localmaxsensor
Local 5 sensor 9: 1075 (1182 91)
show externalmaxsensor
External 4 sensor 7: 1290 (1317 98)
show localpotency
Local 5 potency: 3532
show externalpotency
External 4 potency: 5875
show globalmaxsensor
global 4 sensor: 7: 1290 (1317 98)
show globalmaxnetwork
global 4 potency: 5875
kill
Successful disconnect
gabriel189067@DESKTOP-7OHVILA:~/programa
s$ ./client 127.0.0.1 90902
New ID: 5
Successful disconnect
gabriel189067@DESKTOP-7OHVILA:~/programa
s$
```

5. ERROS NÃO RESOLVIDOS

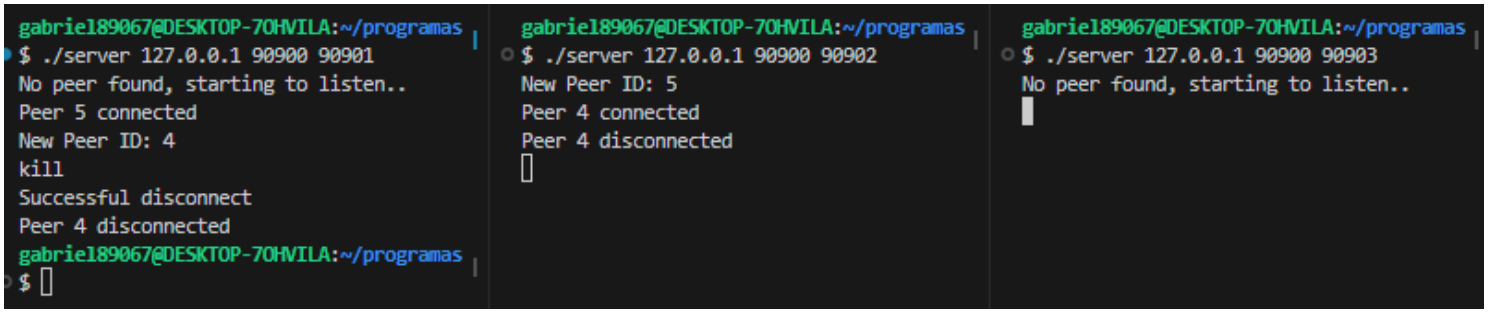
1- Após conectar e desconectar os server via ‘’kill’’ algumas vezes, aparece o erro no connect_p2p , sendo que eu dou close nos sockets e eles não fecham conexão entre si (Mas é resolvido caso se espere algum tempo “a conexão se encerra sozinha , pois naão tem conexão”).

```
gabriel189067@DESKTOP-7OHVILA:~/programas
$ ./server 127.0.0.1 90900 90901
connect_p2p: Connection refused
gabriel189067@DESKTOP-7OHVILA:~/programas
$
```

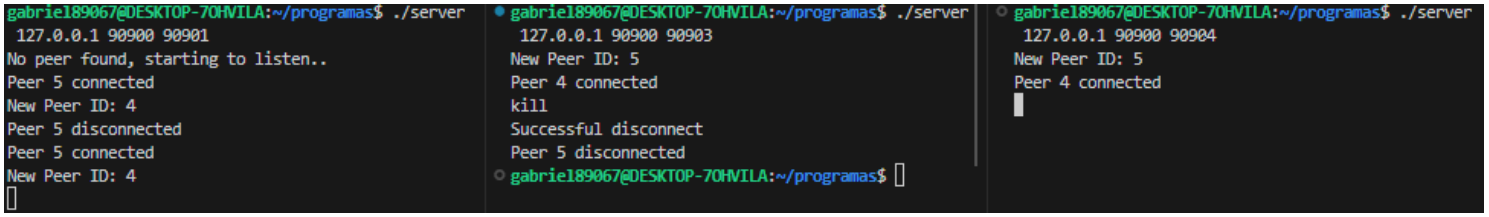
```
gabriel189067@DESKTOP-7OHVILA:~/programas
$ ./server 127.0.0.1 90900 90902
connect_p2p: Connection refused
gabriel189067@DESKTOP-7OHVILA:~/programas
$
```

```
    }
    close(s);
    close(s_p2p);
    break;
```

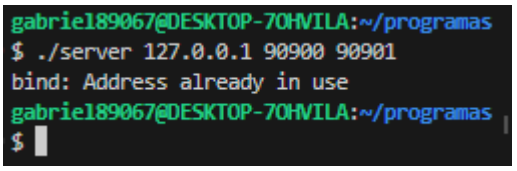
2-Quando eu dou kill no servidor primário (o primeiro a ser aberto) eu não consigo me conectar com o servidor secundário via novo terminal de ./server como mostra a figura (ele cria outro primário).



Entretanto se eu dou kill no server-secundário primeiro a execução continua normal.



3- Mesmo eu fechando os terminais e dando close nos socket para clientes , dá erro no bind do client, ou seja o socket ainda continua aberto (Mas é resolvido caso se espere algum tempo “a conexão se encerra sozinha , pois naão tem conexão”).




```
for (int i = 0; i < clientmax; i++)
{
    sd = client_socket[i];
    size_t count_p2p = send(sd, mensagem, strlen(mensagem) + 1, 0);
    close(sd);
    if (count_p2p == 0)
    {
    }
}

close(s);
close(identificador_serv_externo_int);
identificador_serv_externo_int = 0;
break; ////// esse da pau
```

Como mostrado na segunda figura , quando recebo o kill no server eu mando para todos o clientes conectados a mensagem kill e fecho a conexao csock = sd = cliente_socket , e em seguida dou um close no socket responsável pela conexão com o cliente.