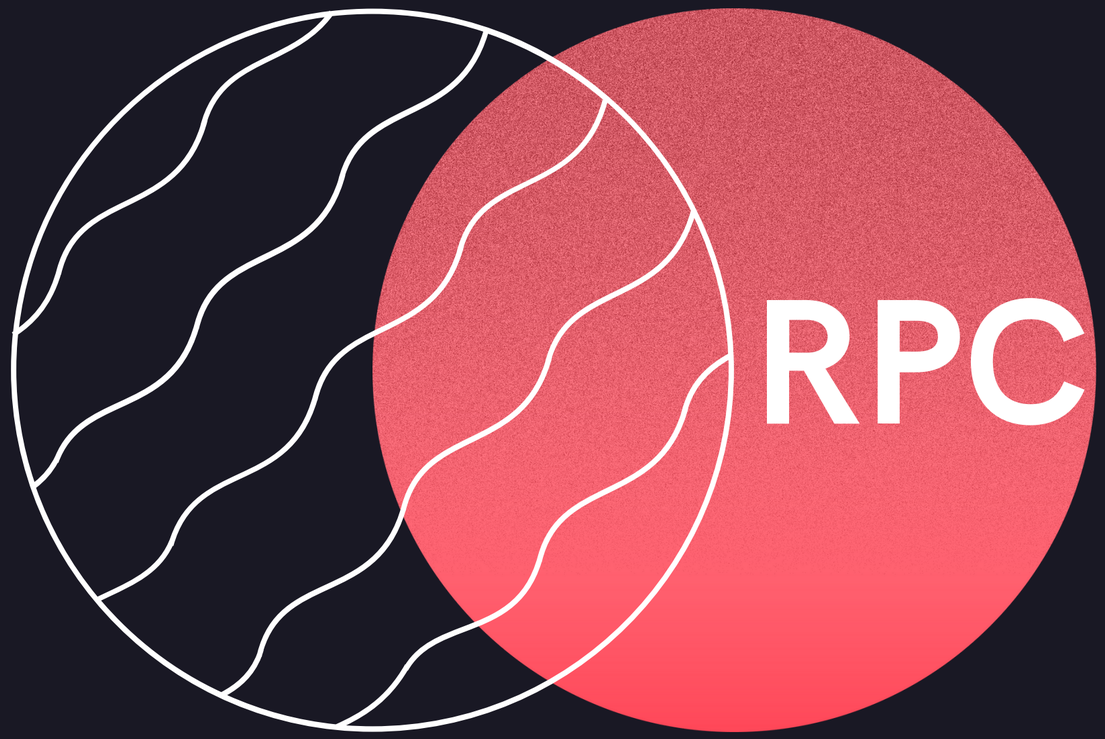
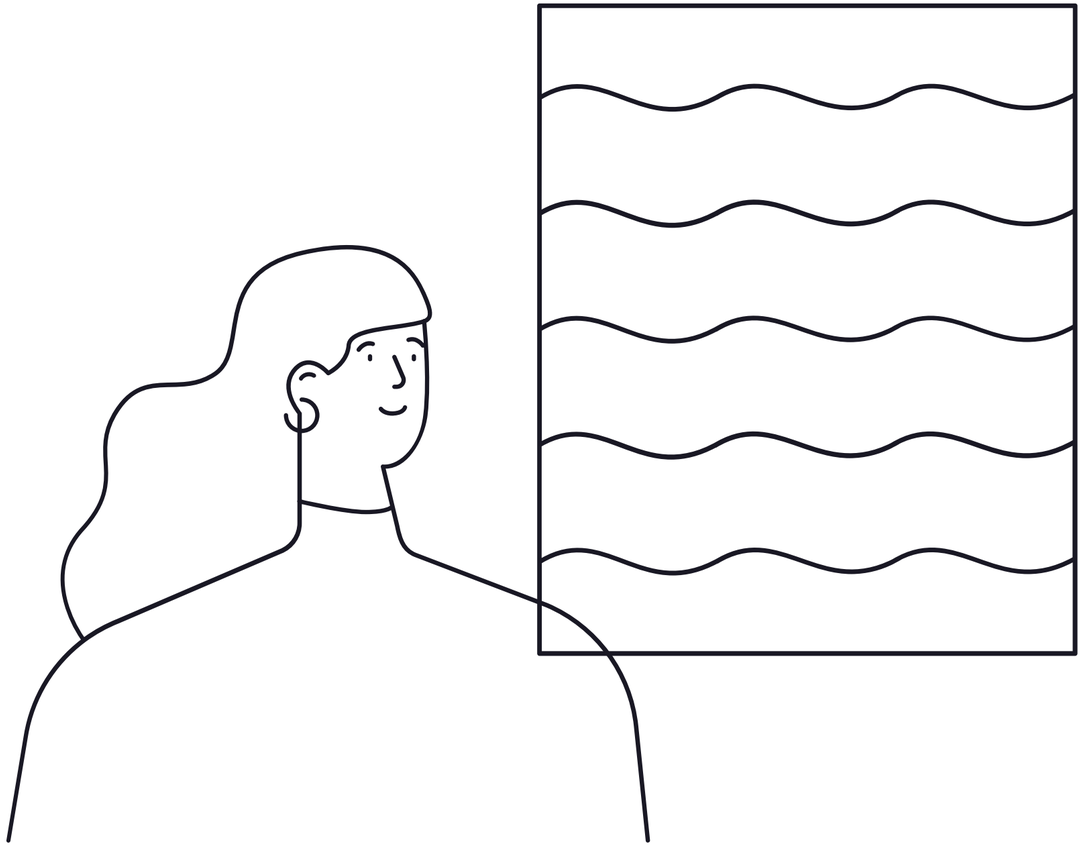


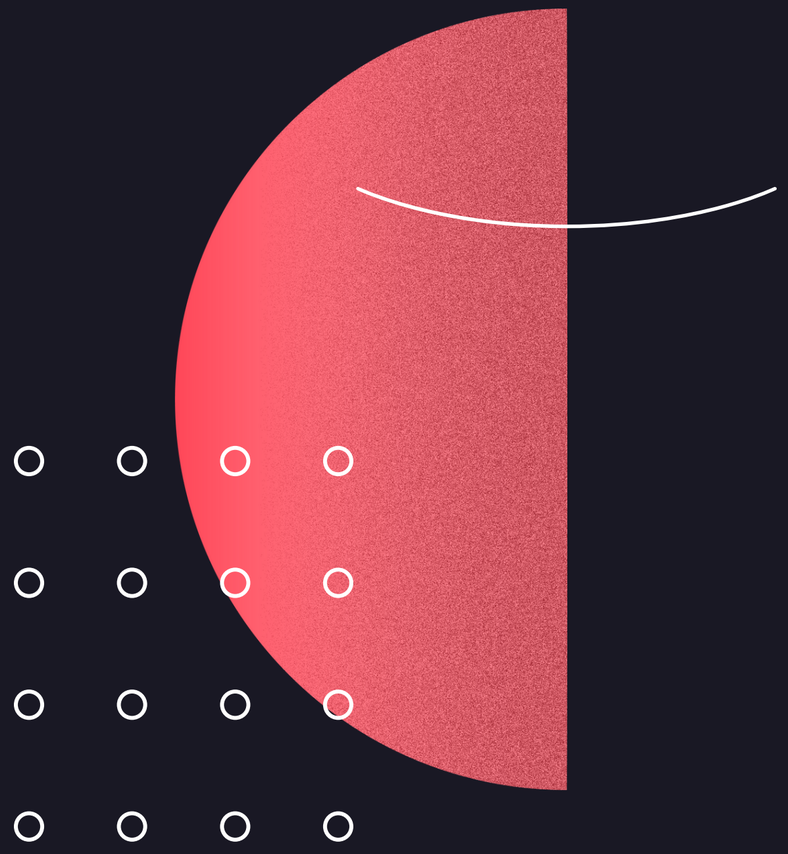
RPC - Laboratório 1

João Pedro Moura Oliveira - 190030879

O que é a arquitetura RPC ?

- Programas que chamam procedimentos em outras máquinas
- Stubs de servidor e cliente
- São construídos através de uma linguagem de definição de interfaces





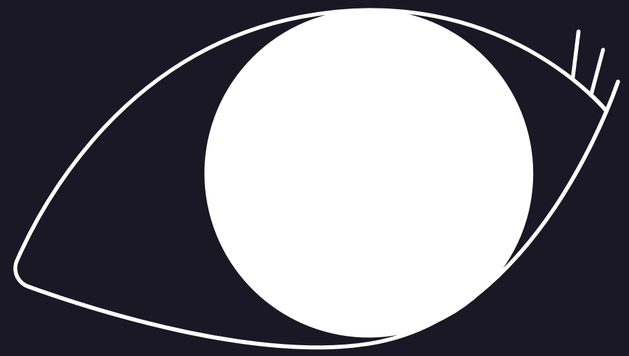
Um pouco mais sobre o problema

```
v[i] = (i - tamanho_do_vetor/2)
```

COMO A SOLUÇÃO FOI PROJETADA ?

```
typedef float vetor<>;
program PROG {
    version VERSAO {
        float MENOR(vetor v) = 1;
        float MAIOR(vetor v) = 2;
    } = 10;
} = 100;
```

```
typedef struct {
        u_int vetor_len;
        float *vetor_val;
} vetor;
```



AS DIVERSAS SOLUÇÕES

UM *WORKER* SEM *THREADS*

```
prog_10 (clnt, (u_int) tam);
clnt_destroy(clnt);
exit (0);
```

```
void
prog_10 (CLIENT *clnt, int tam)
{
        float  *result_1;
        float  *result_2;
        vetor  vetor_arg;

        generate_vetor (&vetor_arg, tam);

        result_1 = menor_10 (&vetor_arg, clnt);
        if (result_1 == (float *) NULL)
                clnt_perror (clnt, "Erro na chamada do RPC\n");
        printf("O menor valor e: %f\n", *result_1);

        result_2 = maior_10 (&vetor_arg, clnt);
        if (result_2 == (float *) NULL)
                clnt_perror (clnt, "Erro na chamada do RPC\n");
        printf("O maior valor e: %f\n", *result_2);
}
```

PROBLEMAS ENCONTRADOS

- Os limites da conexão UDP
- Velocidade de resolução
- *Timeouts*

"DOIS" *WORKERS* SEM *THREADS*

```
for (int i = 0; i < qtdClnts; i++) {
        CLIENT *clnt;
        clnt = create_server (argv[i + 2], "tcp");
        int inicio = i * tam_por_server;
        int final;

        if (i + 1 < qtdClnts) final = inicio + tam_por_server;
        else final = inicio + tam_por_server + (tam % qtdClnts);

        resps resposta = prog_10 (clnt, tam, inicio, final);
        respostas[i] = resposta;
        clnt_destroy(clnt);
}
```

VÁRIOS *WORKER* COM *THREADS*

```c
typedef struct b {
        CLIENT *clnt;
        int tam;
        int inicio;
        int final;
        struct a resposta;
        pthread_t tid;
} param_thread;
```

```c
for (int i = 0; i < qtdClnts; i++) {
        CLIENT *clnt;
        clnt = create_server (argv[i + 2], "tcp");
        int inicio = i * tam_por_server;

        resps[i].clnt = clnt, resps[i].tam = tam;
        resps[i].inicio = inicio;

        if (i + 1 < qtdClnts) resps[i].final = inicio + tam_por_server;
        else resps[i].final = inicio + tam_por_server + (tam % qtdClnts);
        pthread_create (&(resps[i].tid), NULL, prepare_serve, &resps[i]);
}

printf("[CLIENTE]: Esperando finalizacao de %d Thread(s)...\n", qtdClnts);
for (int i = 0; i < qtdClnts; i++)
        pthread_join(resps[i].tid, NULL);

printf("[CLIENTE]: Calculando o menor e o maior valor dos vetores...\n");
menor = resps[0].resposta.result_1;
maior = resps[0].resposta.result_2;
for (int i = 1; i < qtdClnts; i++) {
        if (resps[i].resposta.result_1 < menor) menor = resps[i].resposta.result_1;
        else if (resps[i].resposta.result_2 > maior) maior = resps[i].resposta.result_2;
}
free(resps);
```

COMPARATIVO ENTRE AS SOLUÇÕES

```
./lab1_a_client 127.0.0.1 1000000000  11,97s user 4,80s system 69% cpu 24,235 total (killed)
./lab1_b_client 2 127.0.0.1 172.23.0.2 1000000000 10,82s user 2,09s system 65% cpu 19,791 total
./lab1_b_client 3 127.0.0.1 172.23.0.2 172.24.0.2 1000000000 10,77s user 2,01s system 63% cpu 20,248 total
./lab1_b_client 3 127.0.0.1 172.23.0.2 172.24.0.2 1000000000 17,73s user 5,14s system 197% cpu 11,566 total
./lab1_b_client 4 127.0.0.1 172.23.0.2 172.24.0.2 172.25.0.2 1000000000 19,61s user 6,59s system 229% cpu 11,390 total
```