RELATÓRIO PROJETO 1
MC833 - LAB. DE REDES
1s2019

NOME: João Paulo Soubihe

RA: 151106

- ## Introdução

A proposta deste projeto foi a de entender o funcionamento de uma conexão TCP servidor - cliente, uma stream de dados, e da comunicação entre máquinas de uma rede de computadores. Desenvolvemos um sistema em linguagem C, em baixo nível de abstração, ou seja, tratamos cada operação dos dois lados da “conversa” de maneira detalhada, de forma a analisar com clareza as requisições e execuções necessárias para uma boa troca de dados e os eventuais problemas a serem tratados.

- ## Sistema

Nosso sistema consistia em uma pequena rede social de conexões profissionais. É possível executar algumas funcionalidades como:

1. **Login**

Server

```
server: waiting for connections...
server: got connection from 127.0.0.1
Login de ana@gmail.com.txt
```

Client

```
Bem-vindo!
Login:
ana@gmail.com
(0) Sair
(1) listar todas as pessoas formadas em um determinado curso
(2) listar as habilidades dos perfis que moram em uma determinada cidade
(3) acrescentar uma nova experiência em um perfil
(4) dado o email do perfil, retornar sua experiência
(5) listar todas as informações de todos os perfis
(6) dado o email de um perfil, retornar suas informações
```

2. **(1)Listar todas as pessoas com a mesma formação acadêmica**

Server

```
1
Retornando pessoas com a formação desejada...
Tempo de atualização: 0.035405s
```

Client

```
1
Engenharia
Pessoas com a formação desejada:
maria_silva@gmail.com
silva@gmail.com
ze@gmail.com
```

3. **(2)Listar as habilidades dos perfis de pessoas que residem em uma determinada cidade, talvez próxima ao seu empreendimento**

Server

```
2
Retornando habilidades...
Tempo de atualização: 0.000175s
```

Client

```
2
Campinas
As pessoas dessa cidade possuem as seguintes habilidades:
maria_silva@gmail.com
Análise de Dados, Internet das Coisas, Computação em Nuvem
silva@gmail.com
Análise de Dados, Internet das Coisas, Computação em Nuvem
```

4. **(3)Acrescentar experiências em seu próprio perfil**

Server

```
3
s_p = (3) Teste relatorio
Escrita realizada no arquivo!
Tempo de atualização: 0.000175s
```

Client

```
3
Teste relatorio
```

5. **(4)Conferir as experiências de um usuário em específico**

Server

```
4
Retornando experiencias...
Tempo de atualização: 0.000056s
```

Client

```
4
ze@gmail.com
 (1) Estágio de 1 ano na Empresa X, onde trabalhei como analista de dados
(2) Trabalhei com IoT e Computação em Nuvem por 5 anos na Empresa Y
```

6. **(5)Analisar todos os perfis da rede**

Server

```
5
Retornando .txt dos perfis...
Tempo de atualização: 0.000195s
```

Client

```
5
Email: ana@gmail.com
Nome: Ana Sobrenome: Rocha
Residencia: Valinhos
Formacao Academica: Engenharia de Alimentos
Habilidades: Análise de Dados, Internet das Coisas, Computação em Nuvem
Experiência: (1) Estágio de 1 ano na Empresa X, onde trabalhei como analista de dados
(2) Trabalhei com IoT e Computação em Nuvem por 5 anos na Empresa Y
(3) Teste relatorio
Email: jp.soubihe@gmail.com
Nome: Joao Sobrenome: Paulo
Residencia: Jundiai
Formacao Academica: Ciencia da Computacao
Habilidades: Análise de Dados, Internet das Coisas, Computação em Nuvem
Experiência: (1) Estágio de 1 ano na Empresa X, onde trabalhei como analista de dados
(2) Trabalhei com IoT e Computação em Nuvem por 5 anos na Empresa Y
Email: julia@gmail.com
Nome: Julia Sobrenome: Galvão
Residencia: Rio de Janeiro
Formacao Academica: Letras
Habilidades: Análise de Dados, Internet das Coisas, Computação em Nuvem
Experiência: (1) Estágio de 1 ano na Empresa X, onde trabalhei como analista de dados
(2) Trabalhei com IoT e Computação em Nuvem por 5 anos na Empresa
Email: maria_silva@gmail.com
Nome: Maria Sobrenome: Silva
Residencia: Campinas
Formacao Academica: Engenharia
Habilidades: Análise de Dados, Internet das Coisas, Computação em Nuvem
Experiência: (1) Estágio de 1 ano na Empresa X, onde trabalhei como analista de dados
(2) Trabalhei com IoT e Computação em Nuvem por 5 anos na Empresa Y
Email: silva@gmail.com
Nome: Maria Sobrenome: Silva
Residencia: Campinas
Formacao Academica: Engenharia
Habilidades: Análise de Dados, Internet das Coisas, Computação em Nuvem
Experiência: (1) Estágio de 1 ano na Empresa X, onde trabalhei como analista de dados
(2) Trabalhei com IoT e Computação em Nuvem por 5 anos na Empresa Y
Email: ze@gmail.com
Nome: Jose Sobrenome: Cardoso
Residencia: São Paulo
Formacao Academica: Engenharia
Habilidades: Análise de Dados, Internet das Coisas, Computação em Nuvem
Experiência: (1) Estágio de 1 ano na Empresa X, onde trabalhei como analista de dados
(2) Trabalhei com IoT e Computação em Nuvem por 5 anos na Empresa Y
```

7. **(6)Analisar um perfil em específico**

Server

```
6
Retornando perfil de julia@gmail.com.txt
Tempo de atualização: 0.000090s
```

Client

```
6
julia@gmail.com
Email: julia@gmail.com
Nome: Julia Sobrenome: Galvão
Residencia: Rio de Janeiro
Formacao Academica: Letras
Habilidades: Análise de Dados, Internet das Coisas, Computação em Nuvem
Experiência: (1) Estágio de 1 ano na Empresa X, onde trabalhei como analista de dados
(2) Trabalhei com IoT e Computação em Nuvem por 5 anos na Empresa
```

A ideia do nosso projeto se dá em concentrar todos os perfis e informações sobre eles em um único servidor que proverá dados aos clientes que se conectarem a ele, por meio de uma conexão TCP.

- **<u>Armazenamento</u>**

Esses perfis se encontrarão no servidor, na forma de arquivos .txt e uma imagem de perfil em .jpg.

Tais arquivos estarão organizados da seguinte forma:

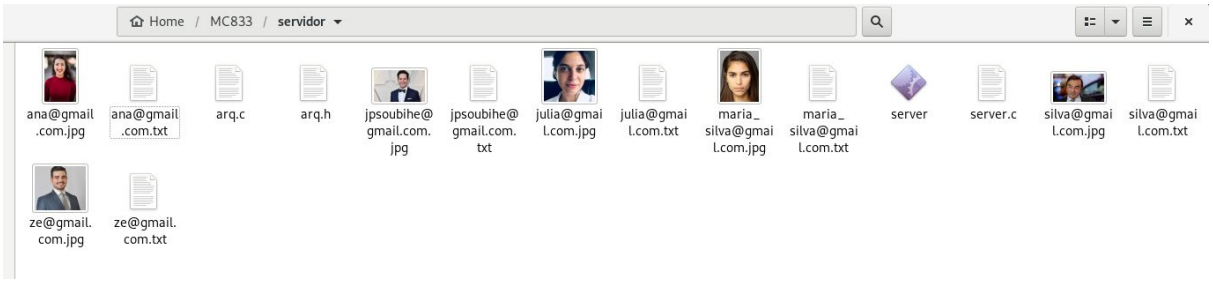
➔ Arquivos .txt

Estarão reunidos na mesma pasta do programa servidor e terão nomes correspondentes ao seu email como EMAIL.txt organizados como no exemplo abaixo

```
Email: ana@gmail.com
Nome: Ana Sobrenome: Rocha
Residencia: Valinhos
Formacao Academica: Engenharia de Alimentos
Habilidades: Análise de Dados, Internet das Coisas, Computação em Nuvem
Experiência: (1) Estágio de 1 ano na Empresa X, onde trabalhei como analista de dados
(2) Trabalhei com IoT e Computação em Nuvem por 5 anos na Empresa Y
(3) Curso de Office avançado
```

➔ Arquivos .jpg

Também serão armazenados na pasta do programa servidor, com a identificação EMAIL.jpg



- **<u>Desenvolvimento</u>**

Primeiramente, trabalhamos no lado do servidor, preparamos todo a infraestrutura necessária para o estabelecimento de uma conexão TCP. Determinamos valores constantes PORT, MAXDATASIZE e BACKLOG representando a porta em que os clientes irão enviar as suas requisições, o máximo tamanho de um pacote de dados tanto recebido quanto enviado e o número máximo de conexões simultâneas que nosso servidor suportará.

Outra escolha feita foi o modo da coleta de dados. Optamos por recorrer ao acesso aos arquivos e captar os dados através da sua leitura. Assim, após qualquer alteração no perfil de um usuário nosso servidor continua consistente e preparado para o envio de informações. Importante ressaltar que, apesar dessa consistência, perdemos eficiência, a execução fica um pouco mais lenta devido às recorrentes manipulações nos arquivos completos. Para melhor compreensão e organização, criamos um arquivo arq.h com o protótipo de funções que fizemos para a manipulação de arquivos, implementado em arq.c. Ficamos sempre muito atentos para o modo de abertura do arquivo, para que não ocorra sobrescrita enquanto um outro cliente requer a leitura do arquivo.

Ao começar os trabalhos na main() declaramos um vetor do tipo struct FileInfo, tal estrutura armazenará o nome do arquivo, para possibilitar o acesso posterior e um descritor do arquivo, para a leitura e eventual manipulação do conteúdo do arquivo.
Através da função getaddrinfo() setamos as informações de endereçamento do nosso servidor, como a porta e o tipo de protocolo para interação. Criamos o socket com as rotinas socket() e setsockopt() e o associamos a porta determinada através da função bind().
Com nosso ambiente preparado, ordenamos o server a “ouvir” possíveis requisições de conexão (listen()). Quando isso ocorre, a função accept() aceita o pedido, replica o processo através de um fork() e inicia a execução de nosso programa enviando uma requisição de login ao cliente, aceito com o envio de um email válido como resposta.
A partir desse momento daremos maior enfoque ao processo filho resultante do fork(), mas é importante salientar que o processo pai continua a ser executado, paralelamente, executando a função listen() e aguardando novas conexões.
É agora que a troca de informações cliente-servidor se intensificam. Foi tomado um cuidado para que as funções de envio e recebimento fossem executadas o menor número de vezes possível durante a conexão, de forma a diminuir o congestionamento da rede e, por consequência, a ocorrência de erros, mesmo considerando a simplicidade do projeto e das informações requisitadas.

Utilizamos 3 funções diferentes, mas com funções equivalentes, para o envio dos pacotes, variando de acordo com o tamanho em bytes do conteúdo a ser enviado:

1. send()
2. sendall()
3. write()

E 2 funções para a leitura:

1. recv()
2. read()

A função send() exige como parâmetros um inteiro como socket descriptor, uma referência ao endereço onde o conteúdo a ser transmitido está armazenado, o tamanho da mensagem e as flags que indicam o tipo de transmissão (que para esse projeto setamos sempre como 0, indicando transmissão da dados “normal”).
Na implementação, sempre procuramos juntar o máximo de conteúdo possível em um único pacote, a fim de evitar o acúmulo de pacotes e aumentar o congestionamento na conexão, aumentando a chance de erros ou perda de informação. Determinamos um tamanho máximo para o pacote MAXDATASIZE, como citado acima.
A função sendall() impõe que toda a informação armazenada no buffer para envio seja enviada. Caso a informação fosse maior que o valor máximo, a função sendall() dividia a mensagem e enviava pacotes até que o conteúdo fosse entregue integralmente ao cliente.
No projeto, enviamos sempre uma string, que varia dependendo do que o cliente pede ao servidor. Tal conteúdo será armazenado pelo cliente, que salvará em arquivos de nome específico, em seu diretório, o texto, ou a imagem, entregue.
Para que o programa do cliente mantenha a execução até o momento em que ocorra um erro ou o mesmo faça uma requisição para encerrar a conexão, inserimos toda a lógica de nossas operações dentro de um loop, mantido por uma flag cuja funcionalidade é de apenas manter a execução do programa no lado do cliente.
No início de cada iteração desse loop imprimimos o menu e o cliente pode escolher que operação gostaria de executar. Essas operações serão representadas por números inteiros, que depois de convertidos para ‘long’ são enviados ao servidor. Dependendo da operação, o cliente enviará ou não novas mensagens seguindo com a interação cliente-servidor.
Para cessar a conexão, o cliente opta pela função 0. Tal função envia uma mensagem ao servidor, sinalizando o fim da comunicação, e fecha o socket dedicado à conexão. O lado servidor recebe essa mensagem e também fecha o socket que tinha reservado, abrindo espaço para outros clientes.

- ## **Análises**

Para efetuar as análises de tempo de operação, utilizamos a função gettimeofday() para medir o tempo TS de consulta no servidor e o tempo TT de consulta no cliente (Como mostrado na referência [2]). A partir destes dois tempos, determinamos o tempo de comunicação como sendo TT - TS.
Medimos 20 execuções de cada operação, com exceção da operação 3 por não haver resposta do servidor para o cliente na nossa implementação, todas

em uma mesma execução do servidor e também do cliente. Usamos dois computadores em uma mesma rede local (Rede do IC 3.5) durante o horário de almoço.
Importante ressaltar que a média foi calculada pelo somatório de todos os tempos de comunicação dividido por 20, o desvio padrão, o erro padrão e a margem de erro, pelas fórmulas descritas em (*),(**),(***),(****). Seguimos então os passos indicados por [3].

(*) média = $\sum Xi/n$, onde Xi é a amostra e n o número de amostras no total
(**) desvio padrão = $[\sum(Xi - media)^2/n]^{1/2}$
(***) erro padrão = desvio padrão / $n^{1/2}$
(****) margem de erro = erro padrão * W(Z/2), onde Z representa o nível de confiança exigido e W() mostra que o valor foi retirado de [4]

Sendo assim, sabemos que os resultados poderão oscilar de Tempo de comunicação ± margem de erro, determinada pelo nível de confiança de 95%.

Os resultados obtidos estão abaixo:

A. Operação 1

Tempo de consulta (cliente)	op	Tempo de consulta (servidor)	Tempo de comunicação(ms)
1,233	1	0,834	0,399
0,848	1	0,46	0,388
1,169	1	0,799	0,37
1,319	1	0,943	0,376
0,981	1	0,629	0,352
1,023	1	0,593	0,43
1,274	1	0,956	0,318
1,432	1	1,076	0,356
0,913	1	0,533	0,38
0,951	1	0,624	0,327
1,029	1	0,7	0,329
1,436	1	1,069	0,367
0,783	1	0,434	0,349
0,964	1	0,599	0,365
1,108	1	0,797	0,311
1,014	1	0,603	0,411
1,11	1	0,72	0,39
1,122	1	0,763	0,359
1,302	1	0,975	0,327
1,045	1	0,721	0,324
Media	0,3614		
Desvio padrao	0,031858122983001		
Erro padrao	0,007123692862554		
Margem de erro	0,013962438010606		

Tabela 1 - Tempos de consulta e comunicação da op. 1 (em ms)

B. Operação 2

Tempo de consulta (cliente)	op	Tempo de consulta (servidor)	Tempo de comunicação(ms)
4,39	2	3,951	0,439000000000001
1,486	2	1,03	0,456
0,976	2	0,623	0,353
1,043	2	0,633	0,41
1,374	2	0,866	0,508
1,5	2	1,063	0,437
1,102	2	0,727	0,375
1,088	2	0,686	0,402
1,317	2	0,939	0,378
1,493	2	1,036	0,457
0,84	2	0,421	0,419
0,974	2	0,652	0,322
1,182	2	0,751	0,431
1,285	2	0,903	0,382
1,041	2	0,623	0,418
1,121	2	0,618	0,503
1,096	2	0,676	0,42
1,256	2	0,866	0,39
1,394	2	0,976	0,418
1,369	2	0,934	0,435
Media	0,41765		
Desvio padrao	0,04406957567302		
Erro padrao	0,009854256694444		
Margem de erro	0,019314343121111		

Tabela 2 - Tempos de consulta e comunicação da op. 2 (em ms)

C. Operação 3

A Operação 3 não foi medida por não haver uma resposta do servidor ao cliente nessa situação.

D. Operação 4

Tempo de consulta (cliente)	op	Tempo de consulta (servidor)	Tempo de comunicação(ms)
0,729	4	0,361	0,368
0,492	4	0,121	0,371
0,519	4	0,141	0,378
0,444	4	0,125	0,319
0,849	4	0,503	0,346
0,504	4	0,131	0,373
0,523	4	0,142	0,381
0,786	4	0,456	0,33
0,511	4	0,141	0,37
0,567	4	0,177	0,39
0,476	4	0,121	0,355
0,828	4	0,432	0,396
0,521	4	0,177	0,344
0,55	4	0,217	0,333
0,875	4	0,468	0,407
0,551	4	0,215	0,336
0,511	4	0,142	0,369
0,517	4	0,125	0,392
0,917	4	0,518	0,399
0,548	4	0,137	0,411
Media	0,3684		
Desvio padrao	0,026270515792424		
Erro padrao	0,005874265911584		
Margem de erro	0,011513561186705		

Tabela 3 - Tempos de consulta e comunicação da op. 4 (em ms)

E. Operação 5

Tempo de consulta (cliente)	op	Tempo de consulta (servidor)	Tempo de comunicação(ms)
4,599	5	4,218	0,381
0,828	5	0,437	0,391
0,984	5	0,616	0,368
1,049	5	0,655	0,394
1,535	5	1,38	0,155
1,323	5	0,983	0,34
1,072	5	0,725	0,347
1,252	5	0,904	0,348
0,836	5	0,455	0,381
1,16	5	0,87	0,29
0,93	5	0,537	0,393
1,36	5	0,958	0,402
1,254	5	0,872	0,382
1,434	5	1,095	0,339
1,051	5	0,669	0,382
1,062	5	0,647	0,415
1,507	5	1,06	0,447
1,202	5	0,842	0,36
0,963	5	0,624	0,339
0,934	5	0,532	0,402
Media	0,3628		
Desvio padrao	0,058754796612702		
Erro padrao	0,013137971923018		
Margem de erro	0,025750424969115		

Tabela 4 - Tempos de consulta e comunicação da op. 5 (em ms)

F. Operação 6

Tempo de consulta (cliente)	op	Tempo de consulta (servidor)	Tempo de comunicação(ms)
0,403	6	0,11	0,293
0,479	6	0,105	0,374
0,501	6	0,104	0,397
0,495	6	0,104	0,391
0,512	6	0,105	0,407
0,505	6	0,107	0,398
0,479	6	0,102	0,377
0,457	6	0,098	0,359
0,405	6	0,089	0,316
0,419	6	0,095	0,324
0,499	6	0,109	0,39
0,932	6	0,133	0,799
0,572	6	0,12	0,452
0,441	6	0,101	0,34
0,46	6	0,105	0,355
0,442	6	0,101	0,341
0,432	6	0,104	0,328
0,52	6	0,11	0,41
0,453	6	0,106	0,347
0,431	6	0,105	0,326
Media	0,3862		
Desvio padrao	0,102024800906446		
Erro padrao	0,02281343902177		
Margem de erro	0,044714340482669		

Tabela 6 - Tempos de consulta e comunicação da op. 6 (em ms)

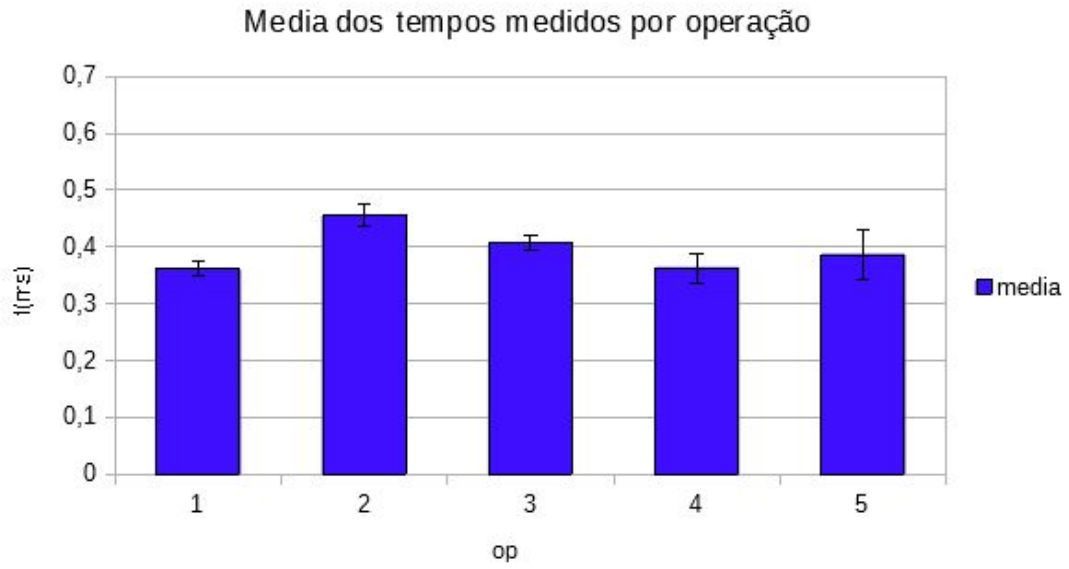


Gráfico 1 - Médias de tempo coletados por operação (em ms)
obs: 3 = op4,4 = op5,5 = op6

Através desses dados vemos que há uma variação praticamente imperceptível para nós, que independe da operação a ser feita no servidor. Muito disso se apoia no fato de que os testes foram feitos em LANs, em uma rede compartilhada pelas duas máquinas em questão, o que facilita a

comunicação e diminui o atraso. Muito provavelmente testes em máquinas de subredes muito distantes entre as máquinas possa trazer resultados mais relevantes no que diz respeito à variação do atraso. Importante ressaltar que não concluímos as transferências de imagens nas operações 5 e 6 (4 e 5 no gráfico), muito provavelmente o seu envio aumentaria algumas variáveis contabilizadas e assim gastariam um tempo maior e estariam mais suscetíveis a mudanças no tráfego da rede.
Outro ponto que pode justificar a pequena margem de erro é o fato de que como a medição foi realizada em um único período do dia as condições da rede foram muito parecidas em cada caso, não provocando um índice relevante.

- **Conclusão**

  Após o desenvolvimento deste projeto pudemos analisar que a linguagem C nos provém uma abordagem em baixo nível, detalhada sobre a comunicação em redes de computadores através do protocolo TCP. Ela requer que tomemos cuidado com muitos pontos da comunicação, tornando o código extenso e algumas vezes uma tarefa simples se torna complexa pelo nível de detalhamento que a implementação em C requer.
  Apesar disso, pode vir a ser uma implementação confiável, tanto no quesito de segurança, disponibilizando a implementação de medidas que visam melhorar esse aspecto, quanto a erros, pensando no protocolo TCP e a vantagem de se estabelecer uma conexão através do handshake.

- **Referências**

  [1] http://beej.us/guide/bgnet/
  [2] www.ic.unicamp.br/~edmundo/MC833/mc833/computar_tempo_trab.pdf
  [3] https://pt.wikihow.com/Calcular-o-Intervalo-de-Confian%C3%A7a
  [4] https://www.statisticshowto.datasciencecentral.com/tables/z-table/