MC833 - Projeto 1
Cliente e Servidor TCP

Luciano Zago - 182835
Vinicius Couto - 188115

Introdução

O projeto tem como objetivo criar uma comunicação TCP, através de sockets, entre um cliente e servidor. A aplicação consiste em um sistema para armazenar perfis de diversos usuários, com as seguintes informações: email (chave), nome, sobrenome, foto da pessoa, residência, formação acadêmica, habilidades e experiência profissional. O servidor deve ser capaz de receber as requisições do cliente, localizado em uma máquina diferente, e transmitir todas as informações disponíveis. O cliente pode visualizar todas as informações disponíveis e adicionar novas experiências em um perfil.

Sistema

1. Descrição Geral

O ambos o servidor eo cliente estão divididos em dois arquivos: ".h" e ".c". Para ambos, o ".h" define assinaturas de funções, inclui bibliotecas necessárias e define constantes essenciais para ambos os lados. Em relação às constantes, deve-se ressaltar a importância de "BUFFLEN" e "PORT". A primeira é responsável por padronizar o tamanho da mensagem, fazendo com que haja um mapeamento 1:1 entre um envio do servidor com sua recepção no cliente, e vice-versa. A segunda define a porta em que deve-se estabelecer a conexão. Ambas devem assumir os mesmo valores no servidor e no cliente. Outros componentes importantes nos *headers* são os *wrappers* para as funções *send* e *recv*. Os wrappers, além de capturar erros das *syscalls*, garantem que o tamanho da mensagem enviada/recebida seja sempre igual ao valor de "BUFFLEN", garantindo o mapeamento 1:1 citado anteriormente.

Os arquivos ".c" do servidor e do cliente são responsáveis por coordenar a troca de mensagens. O cliente executa essencialmente três operações: enviar mensagem, receber mensagem e receber arquivo. O servidor, analogamente, executa três operações: enviar mensagem, receber mensagem e enviar arquivo. Troca de mensagens refere-se a enviar/receber uma única *string* contendo um comando ou dado, já troca de arquivos trata-se de uma série de trocas de mensagens contadas: antes de começar o envio do arquivo, é enviada uma mensagem com o tamanho do arquivo para que o cliente saiba quantos bytes ele deve receber. Todo o fluxo de execução é controlado por duas estruturas *switch-case* que aguardam uma a outra para realizar a comunicação coordenamente. Nota-se que a conexão é mantida em aberto até que seja encerrado ou o processo cliente, ou o processo filho correspondente do servidor.

Os dados estão estruturados como arquivos. No servidor, a pasta "data/" armazena arquivos textos com os dados dos perfis e a pasta "imagem" que guarda as fotos dos

perfis. Os arquivos são nomeados de acordo com o email do perfil, e este email é utilizado como chave na busca de arquivos. Um arquivo “index.txt”, utilizado para acessar os arquivos quando a opção selecionada não especifica um email, armazena todos as chaves (emails) presentes em dado momento no servidor. O cliente não armazena dados dos perfis, apenas exibe-os no terminal. Excetua-se as imagens, já que não há como exibi-las no terminal, estas são salvas na pasta “data/” presente no cliente.

2. **Casos de Uso**

help	- Mostra os comandos disponiveis
1 <curso>	- Lista todas as pessoas formadas em um determinado curso
2 <cidade>	- Lista as habilidades dos residentes da cidade
3 <emai> <experiência>	- Adiciona determinada experiência em um perfil
4 <email>	- Lista todas as experiências de um perfil
5	- Lista todas as informações de todos os perfis
6 <email>	- Lista todas as informações de um perfil

Estrutura de dados e armazenamento

Os dados do servidor estão todos armazenados dentro do diretório “server/data”. As imagens de perfil estão armazenadas no diretório “server/data/images”.

A estrutura de dados consiste em um arquivo “index.txt” para registrar as chaves (emails) cadastradas no sistema, e arquivos “[email].txt” para armazenar as informações relativas ao perfil do usuário. As fotos de perfil são armazenadas em arquivos “images/[email].jpg”.

Cada tipo de informação do perfil é armazenada consistentemente em determinada linha do arquivo de texto, como no esquema a seguir:

Linha 1	Nome
Linha 2	Sobrenome
Linha 3	Cidade
Linha 4	Curso
Linha 5	Habilidades
Linha 5+i, i>=1	Experiência nº i

Implementação do servidor TCP

O servidor foi implementado visando o armazenamento de dados de forma persistente e paralelismo entre conexões. A fim de garantir o paralelismo, o servidor TCP possui, inicialmente, um único processo pai que atua como *dispatcher* para novas conexões. O *dispatcher* é associado a porta “PORT” com protocolo IPv4 (AF_INET) e, para que este *socket* seja associado a porta “PORT” de todas as interfaces de rede do computador, ele recebe o IP “INADDR_ANY”. Enquanto o dispatcher aguarda novas conexões com o uso da *syscall listen*, os clientes já conectados são distribuídos aos processos filhos que ficam responsáveis por atender as *requests* dos clientes associados a eles sem que o processo pai tenha que interromper a escuta por novas conexões [1]. Para fins de praticidade, supôs-se que todos os

comandos fornecidos pelo cliente eram válidos dentro do escopo de operações do servidor; também assumiu-se que as conexões são sempre estáveis. A finalidade das suposições é de evitar tratamento de erros, o que aumentaria a complexidade do servidor e divergiria do objetivo do projeto.

Resultados

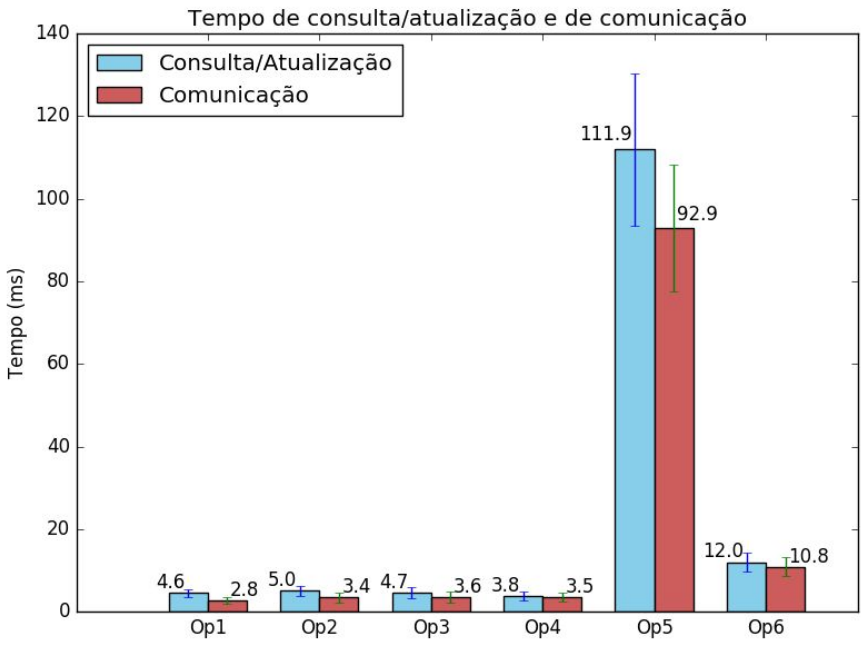
Os resultados correspondem à consultas realizadas em máquinas diferentes em uma rede local.

O cálculo do erro com intervalo de confiança de 95% foi realizado como $\sigma_M = 1.96 \times \frac{\sigma}{\sqrt{N}}$ sendo σ o desvio padrão e N o número total de iterações. O valor 1.96 é a constante que representa o intervalo de confiança de 95% [2].

Tempos de consulta/atualização do client						
Iterações	**Opções**					
-	**1**	**2**	**3**	**4**	**5**	**6**
1	0.004794	0.003919	0.002509	0.001589	0.221225	0.012490
2	0.007689	0.003065	0.008735	0.001537	0.134130	0.019630
3	0.002781	0.010434	0.009400	0.001992	0.058004	0.012356
4	0.006732	0.002593	0.001734	0.006122	0.132866	0.005631
5	0.009084	0.006894	0.005084	0.001806	0.134607	0.010563
6	0.002819	0.002858	0.001801	0.006181	0.059027	0.005422
7	0.002996	0.010462	0.001596	0.006709	0.143028	0.005518
8	0.002842	0.004602	0.001520	0.001462	0.124570	0.018680
9	0.003810	0.002857	0.008123	0.001301	0.067282	0.012661
10	0.002869	0.004574	0.009190	0.003714	0.077694	0.011924
11	0.003771	0.003283	0.001668	0.003749	0.131294	0.010087
12	0.004161	0.002416	0.004339	0.001521	0.130589	0.010135
13	0.005968	0.006822	0.009446	0.007507	0.058454	0.014661
14	0.002914	0.009027	0.001669	0.002127	0.128978	0.005602
15	0.002868	0.005429	0.006345	0.001476	0.125042	0.014367
16	0.002632	0.002389	0.009018	0.002354	0.061815	0.021045
17	0.002836	0.008559	0.002040	0.003829	0.130240	0.005791
18	0.007198	0.005819	0.001660	0.005413	0.131553	0.008406
19	0.006994	0.002259	0.003192	0.008710	0.059055	0.020794
20	0.005329	0.002357	0.004551	0.007007	0.129339	0.013668
Média (s)	**0.004554**	**0.005031**	**0.004681**	**0.003805**	**0.111940**	**0.011972**
Erro (95%)	**0.000891**	**0.001220**	**0.001397**	**0.001077**	**0.018476**	**0.002261**

Tempos de operação do servidor						
Iterações	**Opções**					
-	**1**	**2**	**3**	**4**	**5**	**6**
1	0.001977	0.002880	0.00191	0.000344	0.186221	0.001653
2	0.001850	0.001890	0.000915	0.000340	0.028249	0.001089
3	0.001678	0.001591	0.001647	0.000276	0.008397	0.001279
4	0.001767	0.001742	0.000912	0.000366	0.024919	0.001271
5	0.001913	0.001689	0.000810	0.000352	0.006030	0.001169
6	0.001562	0.001704	0.000836	0.000347	0.008940	0.001217
7	0.001688	0.001605	0.000806	0.000211	0.009018	0.001271
8	0.001694	0.001439	0.000823	0.000203	0.008587	0.000514
9	0.001499	0.001620	0.000828	0.000151	0.008561	0.001278
10	0.001763	0.001194	0.001366	0.000350	0.008726	0.001126
11	0.001715	0.001496	0.000837	0.000292	0.008184	0.001186
12	0.001572	0.001418	0.000858	0.000281	0.008231	0.001181
13	0.003371	0.001527	0.001659	0.000351	0.008163	0.001136
14	0.001771	0.001313	0.000854	0.000354	0.009661	0.000539
15	0.001790	0.001421	0.000721	0.000293	0.008591	0.001203
16	0.001617	0.001270	0.002531	0.000235	0.007548	0.001032
17	0.001733	0.001417	0.001297	0.000338	0.008140	0.001275
18	0.001484	0.001495	0.000865	0.000350	0.008094	0.001159
19	0.001722	0.001413	0.000863	0.000389	0.008520	0.001170
20	0.001316	0.001571	0.001057	0.000287	0.008299	0.001140
Média (s)	**0.001774**	**0.001585**	**0.001120**	**0.000306**	**0.019054**	**0.001144**
Erro (95%)	**0.000178**	**0.000153**	**0.000209**	**0.000028**	**0.017423**	**0.000107**

Tempos de comunicação						
Iterações	**Opções**					
-	**1**	**2**	**3**	**4**	**5**	**6**
1	0.002817	0.001039	0.000599	0.001245	0.035004	0.010837
2	0.005839	0.001175	0.007820	0.001197	0.105881	0.018541
3	0.001103	0.008843	0.007753	0.001716	0.049607	0.011077
4	0.004965	0.000851	0.000822	0.005756	0.107947	0.004360
5	0.007171	0.005205	0.004274	0.001454	0.128577	0.009394
6	0.001257	0.001154	0.000965	0.005834	0.050087	0.004205
7	0.001308	0.008857	0.000790	0.006498	0.134010	0.004247
8	0.001148	0.003163	0.000697	0.001259	0.115983	0.018166
9	0.002311	0.001237	0.007295	0.001150	0.058721	0.011383
10	0.001106	0.003380	0.007824	0.003364	0.068968	0.010798
11	0.002056	0.001787	0.000831	0.003457	0.123110	0.008901
12	0.002589	0.000998	0.003481	0.001240	0.122358	0.008954
13	0.002597	0.005295	0.007787	0.007156	0.050291	0.013525
14	0.001143	0.007714	0.000815	0.001773	0.119317	0.005063
15	0.001078	0.004008	0.005624	0.001183	0.116451	0.013164
16	0.001015	0.001119	0.006487	0.002119	0.054267	0.020013
17	0.001103	0.007142	0.000743	0.003491	0.122100	0.004516
18	0.005714	0.004324	0.000795	0.005063	0.123459	0.007247
19	0.005272	0.000846	0.002329	0.008321	0.050535	0.019624
20	0.004013	0.000786	0.003494	0.006720	0.121040	0.012528
Média (s)	**0.002780**	**0.003446**	**0.003561**	**0.003500**	**0.092886**	**0.010827**
Erro (95%)	**0.000870**	**0.001247**	**0.001311**	**0.001067**	**0.015382**	**0.002282**



Analisando os resultados, nota-se que o tempo de comunicação corresponde a uma parte significativa do tempo de consulta/atualização, sendo responsável por mais de noventa por cento do tempo na maioria dos casos. Como as tarefas executadas pela máquina são simples, é esperado que o tempo de processamento do servidor (representado pela diferença entre o tempo de consulta/atualização e o tempo de comunicação) fosse mínimo. Um fator que pode ter colaborado com o aumento do tempo de relativo de comunicação é o tamanho do buffer que fora utilizado, uma vez que são transmitidos apenas duzentos e cinquenta e seis bytes por *syscall*, tanto para *recv* quanto para *send*.

Os maiores tempos de processamento no servidor em relação ao tempo de consulta/atualização do cliente, são das opções um, dois e cinco. Justifica-se essa observação pelo fato que estas operações iteram por todos os arquivos do servidor com a finalidade de buscar os resultados desejados, aumentando a relação do tempo de processamento pelo de comunicação. O processamento necessário para abrir, ler e enviar um perfil, como é feito no caso seis, é menor do que os casos citados, uma vez que a tal opção recebe o email (chave) como parâmetro, encontrando o os arquivos desejados facilmente, e so requer a abertura e leitura de um único perfil.

As operações cinco e seis são as que apresentam os maiores tempos totais, uma vez que ambas operações requerem o envio de todas as informações do perfil. O tempo de comunicação nestes casos é fortemente ampliado pela quantidade de mensagens necessárias para enviar os arquivos de imagem, algo que não ocorre nos outros casos. A operação cinco, em particular, apresenta os maiores tempos medidos: consulta/atualização, processamento e comunicação. Tal resultado é coerente, pois esta é a única opção que envolve abrir, ler e enviar absolutamente todos os dados contidos no servidor e, além disso, fazer a busca de perfis pelo arquivo “index.txt”. Devido a escala da opção cinco, os tempos de operação e comunicação ficam muito acima de todas as outras operações.

Conclusão

O projeto criou uma comunicação TCP entre um cliente e servidor, localizados em máquinas deferentes em uma rede local, para atender os requisitos das seis operações citadas e as condições de paralelismo e persistência.

Analisando-se os resultados, nota-se que os tempos de comunicação foram sempre menores do que os tempos de consulta/atualização, o que é consistente com o esperado, pois o tempo de comunicação está contido no tempo total de consulta. As operações um, dois, três e quatro apresentaram tempos menores do que as operações cinco e seis, pois as últimas requerem o envio de todas as informações do perfil.

Dessa forma, o projeto atendeu os requisitos para um servidor TCP concorrente e apresentou uma performance coerente com o esperado. A persistência e confiança na transmissão de arquivos foi garantida em ambos servidor e cliente pelo protocolo TCP e a interação adequada dos programas com o protocolo através de wrappers.

Referências

1. Beej's Guide to Network Programming (http://beej.us/guide/bgnet/html/single/bgnet.html)
2. Confidence Interval on the Mean (http://onlinestatbook.com/2/estimation/mean.html)