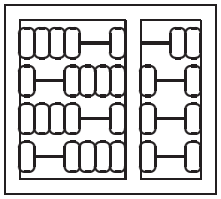
**UNIVERSIDADE ESTADUAL DE CAMPINAS**

INSTITUTO DE COMPUTAÇÃO



# SERVIDOR DE AGENDA BASEADO EM SOCKET TCP

*Relatório do primeiro laboratório de MC823*

**Aluno**: Marcelo Keith Matsumoto **RA**: 085937

**Aluno**: Tiago Chedraoui Silva **RA**: 082941

**Resumo**

O TCP (Transmission Control Protocol) é um protocolo do nível da camada de transporte [3]. Algumas de suas características são transferência garantida, controle de congestão, ser orientado à conexão e controle de fluxo.

Utilizando esse protocoloco desenvolveu-se uma aplicação distribuída que simulava uma agenda para o mês de abril no qual avaliou-se a interação entre servidor e cliente através de diversas funções (inserção de dados na agenda, recuperação de dados, remoção de dados) em termos de tempo de comunicação e tempo total.

## Sumário

# 1 Introdução

O TCP (Transmission Control Protocol) é um protocolo do nível da camada de transporte [3]. Dentre usas principais características temos:

**Orientado à conexão** Cliente e o servidor trocam pacotes de controle entre si antes de enviarem os pacotes de dados. Isto é chamado de procedimento de estabelecimento de conexão (handshaking).

**Transferência garantida** Dados trocados são livres de erro, o que é conseguido a partir de mensagens de reconhecimento e retransmissão de pacotes.

**Controle de fluxo** Assegura que nenhum dos lados da comunicação envie pacotes rápido demais, pois uma aplicação em um lado pode não conseguir processar a informação na velocidade que está recebendo.

**controle de congestão** Ajuda a prevenir congestionamentos na rede.

**Fim-a-fim** Conexão é sempre entre o host emissor e o host receptor.

Cada um dos segmentos da camada transporte tem em seu cabeçalho campos denominado número de portas que indicam a qual processo o mesmo deve ser entregue, existindo um número de porta do emissor e o número de porta do receptor.

Possuindo as duas portas, pode-se realizar uma conexão entre elas conhecida por socket. Com o socket é possível diferenciar os canais utilizados por cada processo de aplicação.

Os segmentos TCP, através do protocolo IP, são entregues a sistemas terminais, cada um identificado por seu endereço IP. E o protocolo TCP cuida de repassar os dados à cada processo de aplicação de acordo com porta especificada no cabeçalho do segmento.

Devido a importância do protocolo, este laboratório tem o objetivo de medir o tempo total e de comunicação de uma conexão TCP entre um cliente e um servidor.

# 2 Servidor de agenda

O sistema implementado, uma agenda distribuída, se baseia numa comunicação cliente-servidor. Nele o servidor possui todas as informações da agenda que estão armazenadas em um banco de dados, assim como as opções de interações com os dados que são apresentadas aos clientes em formas de um menu. O cliente só escolhe alguma opção de interação com os dados de acordo com menu. No menu inicial pode-se logar com um usuário ou criar um usuário. E cada usuário possui sua agenda. Assim, dentre as possibilidades de interações para um usuário logado tem-se:

- Inserção de um compromisso que possui um nome, dia, hora, e minuto.
- Remoção de um compromisso através de seu nome

- Pesquisa de compromisso por dia
- Pesquisa de compromisso por dia e hora
- Ver todos os compromisso de mês de abril

# 3 Ambiente de implementação

O sistema de agenda foi implementado e executado nos seguintes sistemas operacionais :

- FC14 - Fedora Laughlin Linux 2.6.35.11

O sistema de agenda foi implementado na linguagem C. Para o armazenamento dos dados, utilizou-se arquivos. Cada usuário possui um arquivo, a sua agenda, no qual armazena-se o nome do compromisso, o dia, a hora e o minuto do mesmo. O sistema lê esse arquivo quando o usuário loga e transfere-o à memória principal, e a cada alteração na agenda o servidor atualiza as informações dos arquivos.

O servidor aceita diversas conexões de clientes, funcionando perfeitamente para interações em diferentes agendas, pois cada cliente possui além de um processo único, que foi criado em um fork, possui um ponteiro para sua agenda. Assim, o servidor consegue alterar todas as agendas independemente.

O nosso sistema, além disso,apresenta transparência ao usuário. Os tipos de transparência a serem destacados são:

**Acesso:** Esconde as diferenças nas representações de dados e na invocação de funções ou métodos para facilitar a comunicação entre objetos da rede.

**Localização:** Esconde o local em que o objeto se encontra.

**Concorrência:** Esconde como as atividades são coordenadas entre os objetos para obter consistência em um nível mais alto.

Listaremos a seguir algumas funções implementadas de interação:

- Funções de interação com o banco de dados são:

```
1    /* Encontra usuario que ja esta cadastrado no servidor e verifica
2    senha*/
3    int findUser(char nome[], char pwd[]);
4
5    /* Insere novo usuario (nome e senha) no banco de dados*/
6    int newUser(char nome[], char senha[]);
7
8    /*Carrega agenda de usuario*/
9    int loadCal(User *user);
```

```
10
11      /*Salva agenda*/
12      int saveCal(User *user);
```

- Funções de interação com a agenda são:

```
1       /*Cria agenda para usuario*/
2       User * agenda_init(char nome[]);
3
4       /*Apaga agenda da memoria principal do servidor */
5       void user_destroy(User *u);
6
7       /*Insere compromisso na agenda*/
8       int set_task(int dia,int hora, int min,char task[], User *u);
9
10      /*Cria compromisso */
11      Agenda * task_init(int dia,int hora, int min,char task[]);
12
13      /*Retorna compromissos do mes de abril*/
14      int verMes(int new_fd, User *u);
15
16      /*Retorna compromissos do mes de abril em determinado dia*/
17      int verDia(int new_fd, User *u, int dia);
18
19      /*Retorna compromissos do mes de abril em determinado dia e
20      determinada hora*/
21      int verHora(int new_fd, User *u, int dia, int hora);
22
23      /*Remove compromisso da agenda pelo nome*/
24      int delTask( User *u, char nome[]);
25
26      /*Comapara data de compromissos*/
27      int compData(Agenda *newTasks,Agenda *tasks);
```

- Funções de interação servidor-cliente criadas foram:

```
1
2       /*Envia mensagem ao cliente*/
3       void sendStr(int sockfd, char str[]);
4
5       /*Le mensagem do cliente*/
6       int leOpcao(struct sockaddr_storage their_addr, int sockfd);
7
8       /*Apresenta opcoes de login ou criacao novo usuario*/
```

```
9     void menu(int new_fd, struct sockaddr_storage their_addr);
10
11    /*Apresenta opcoes de interacao com agenda*/
12    void menu2(int new_fd, struct sockaddr_storage their_addr, User *user);
13
14    /*Envia mensagem para servidor*/
15    void envia_pct( int sockfd, char s[], int size){
```

# 4 Tempos de comunicação e total

Aplicamos o cálculo de tempo ao programa principal de forma a obtermos o tempo total, tempo de comunicação e os tempos da execução de cada função. Para o tempo total, no cliente pega-se o tempo antes do primeiro send e após o último recv. Para o tempo de comunicação, pega-se o tempo total e subtrai-se o tempo de processamento do servidor, que é depois do primeiro recv e antes do último send.

Para o tempo total das funções obteu-se o tempo de inserir um compromisso, remover o compromisso, ver a agenda do mês, ver a agenda de um dia, ver a agenda de uma hora. Os dados e os testes estão exemplificados nas tabelas I, II, III, IV e V.

O resultado obtido para 100 valores foi:

| Valor | Tempo |
|---|---|
| Max | 185.351 ms |
| Min | 45.577 ms |
| Média | 55.090 ms |
| Desvio | 0.743 ms |

(a) Tempo total

| Valor | Tempo |
|---|---|
| Max | 1.483 ms |
| Min | 0.500 ms |
| Média | 0.725 ms |
| Desvio | 0.076 ms |

(b) Tempo de comunidação

Tabela I: Conexão e fechamento de conexão com servidor

| Valor | Tempo |
|---|---|
| Max | 114.572 ms |
| Min | 32.248 ms |
| Média | 36.782 ms |
| Desvio | 0.069 ms |

(a) Tempo total

| Valor | Tempo |
|---|---|
| Max | 0.837 ms |
| Min | 0.386 ms |
| Média | 0.705 ms |
| Desvio | 0.001 ms |

(b) Tempo de comunidação

Tabela II: Conexão, login na conta, inserção de compromisso e fechamento de conexão com servidor

Note que os tempos das tabelas IIIa, IVa e Va estão, respectivamente, em ordem decrescente de tamanho. Isso se deve ao fato de que a operação 3 o programa busca pelo dia, hora e minuto do compromisso, fazendo com que o número de comparações feitas seja maior que as operaçãoes IVa e Va, tornando aquela operação mais lenta. O mesmo pode afirmar entre as operações 4 e 5.

| Valor | Tempo |
|---|---|
| Max | 214.583 ms |
| Min | 37.724 ms |
| Média | 47.599 ms |
| Desvio | 0.468 ms |

(a) Tempo total

| Valor | Tempo |
|---|---|
| Max | 0.851 ms |
| Min | 0.515 ms |
| Média | 0.715 ms |
| Desvio | 0.004 ms |

(b) Tempo de comunidação

Tabela III: Conexão, login na conta, remoção de compromissos e fechamento de conexão com servidor

| Valor | Tempo |
|---|---|
| Max | 168.404 ms |
| Min | 32.763 ms |
| Média | 38.612 ms |
| Desvio | 0.412 ms |

(a) Tempo total

| Valor | Tempo |
|---|---|
| Max | 1.472 ms |
| Min | 0.523 ms |
| Média | 0.727 ms |
| Desvio | 0.002 ms |

(b) Tempo de comunicação

Tabela IV: Conexão, login na conta, ver compromissos do determinado dia e hora e fechamento de conexão com servidor

| Valor | Tempo |
|---|---|
| Max | 76.481 ms |
| Min | 28.966 ms |
| Média | 32.334 ms |
| Desvio | 0.215 ms |

(a) Tempo de total

| Valor | Tempo |
|---|---|
| Max | 0.905 ms |
| Min | 0.504 ms |
| Média | 0.716 ms |
| Desvio | 0.000 ms |

(b) Tempo de comunidação

Tabela V: Conexão, login na conta, ver todos os compromissos do mês e fechamento de conexão com servidor

# 5 Conclusão

# Referências


[1] Brian "Beej Jorgensen"Hall Beej's Guide to Network Programming - Using Internet Sockets . Disponível em *http://beej.us/guide/bgnet/*, [Último acesso: 07/04/2011].

[2] Mike Muuss Packet Internet Grouper (Groper) . Disponível em *http://linux.die.net/man/8/ping*, [Último acesso: 10/04/2011].

[3] J. Kurose e K. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Pearson Addison Wesley, 3 ed., 2005.