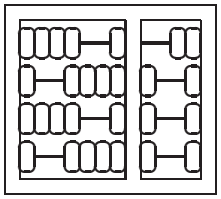
UNIVERSIDADE ESTADUAL DE CAMPINAS

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

Sumário

1 Introdução **1**

2 Servidor de agenda **1**

3 Ambiente de implementação **2**

4 Tempos de comunicação e total **3**

5 Conclusão **4**

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. Na qual 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. Os dados da agenda foram armazenados em arquivos, onde o servidor lê quando um usuário loga no sistema de agenda e os armazena em memória. A cada alteração na agenda o servidor atualiza as informações dos arquivos.

Listaremos a seguir algumas funções implementadas:

- 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

O round-trip time (RTT) é o tempo que leva-se para um sinal ser enviado mais o tempo que se leva para receber um acknowledgment que o sinal foi recebido. A ferramenta administrativa para as redes de computadores denominada “Ping” [2] é usada para testar se um host é alcançável e para medir o RTT para mensagens enviadas do host remetente para o destinatário. Utilizado o ping nas máquinas nos quais servidor e cliente foram usadas obtivemos um valor de : PREENCHER AQUI!!!!!!

Posteriormente, aplicamos o cálculo de tempo ao programa principal de forma a obtermos o tempo total e tempo de comunicação. 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.

O resultado obtido para 100 valores foi:

Para certificarmos do tempo, fizemos o teste com duas situações, uma mais rápida, uma mais lenta. E obtivemos valores muito próximos da média de tempo de ida e volta de um pacote(RTT).

Tabela I: Teste 1: conexão e fechamento de conexão com servidor

Valor	Tempo
Max	0.876 ms
Min	0.467 ms
Média	0.701 ms
Desvio	0.007 ms

Tabela II: Teste 2: conexão, login na conta, ver agenda do mês e fechamento de conexão com servidor

Valor	Tempo
Max	4.959 ms
Min	0.190 ms
Média	0.638 ms
Desvio	0.013 ms

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.