/* Diego Aron Poplade - GRR20101352
 * Gustavo Toshi Komura - GRR20102342
 * Prof. Elias P. Duarte Jr.
   Servidor TCP Iterativo   */

Observação: A codificação do texto está em UTF-8.

Introdução:

Este trabalho propõe a implementação de um sistema cliente-servidor que possui uma unidade intermediária(dispatcher) para controle de transmissão de mensagens. Este controle é feito apartir de uma lista de servidores conectados com o dispatcher que são selecionados em dois modos:

-aleatório: Os servidores são selecionados de forma aleatória pelo dispatcher

-Round-Robin: Os servidores são selecionados em ordem dentro da lista do dispatcher. Quando o último servidor é escolhido, o dispatcher selecionará o primeiro servidor da lista para continuar as transmissões.

Modo de Funcionamento:

O funcionamento do sistema ocorre de forma bem simples. Primeiro é inciado o dispatcher com os seguintes parâmetros:

DISPATCHER

<**porta**> - Porta com o qual será feita a comunicação com o dispatcher
<tipo-de-escolha-do-servidor(**aleatória/round-robin**)> - Escolhe qual vai ser a forma de seleção dos servidores

O dispatcher então inicializará uma lista com tamanho MAX_SERVER_NUMBER e entrará em modo servidor, esperando uma requisição JOIN dos servidores para estes serem incluidos na lista.

Agora deve ser feira a inicialização dos servidores, que segue com os seguintes parâmetros:

SERVIDORES

<**nome-dispatcher**> - Endereço do dispatcher
<**porta-dispatcher**> - Porta de comunicação com o dispatcher
<**porta-servidor**> - Porta com o qual o dispatcher fará comunicação com o servidor

No momento de inicialização(boot), o servidor enviará uma requisição **JOIN** para o dispatcher para entrar em sua lista. Se a entrada for confirmada o dispatcher responde com um **ACCEPTED** então o servidor

entrará em modo de espera para receber as requisiçoes dos clientes.
Se a lista do dispatcher estiver cheia, este então este envia uma
resposta **LIST_FULL** no qual encerrará o servidor já que ele não poderá
entrar na lista.

Vale destacar neste ponto, que ao se iniciar alguma comunicação entre
os servidores e os cliente, não conseguimos descobrir qual o motivo
pelo qual o dispatcher não estava aceitando mais nenhum servidor,
por mais que existisse espaço na lista de servidores.

Agora que os servidores e o dispatcher estão prontos, os clientes podem
enviar requisições.

Neste trabalho a implementação do sistema é de um simples date remoto e
mais algumas perguntas básicas para o servidor, como por exemplo, uma delas sugeridas pelo professor em aula, uma calculadora,
onde o cliente irá pedir qual é a data e hora local do servidor ou perguntar
algo ou pedir para solucionar algum tipo de conta.

Os formatos variam entre **DATA**, **PERGUNTA** e **FALHA**.

- **DATA** vai ser uma requisição de **DATA** para algum servidor.

- **PERGUNTA** vai ser uma requisição de alguma resposta,
podendo ser uma pergunta simples, como "qual é o seu nome ?"
como uma conta matemática.

-- As perguntas podem variar de "**qual o seu nome ?**"
"**qual sua idade ?**" até "**quanto é a soma de x + y ?**"

Para garantir que o usuário não viesse a causar danos
a comunicação, ele é restringido a escolher as opções de pergunta
através de números, ao invés de poder realmente perguntar algo.
Acabamos optando por esta implementação, pois tratar todos os
possiveis casos de strings que poderiam ser digitados era imensa.
Chegamos até a pensar em algumas tipos de soluções, como por exemplo,
utilizar expressões regulares, mas já que isso iria ocupar muito tempo,
acabamos deixando de lado.

- **FALHA** vai ser quando alguma inconsistência ocorrer no cliente,
como por exemplo escolher uma opção não existente,
porém como o cliente não pode gerar nenhuma resposta, ele pergunta
para o servidor o que responder.

No dispatcher foram feitos alguns trabalhos braçais, que envolveram
o tratamento de strings, não necessáriamente para passar a mensagem para
os servidores, mais sim para mostrar nos logs.

Para o cliente, ele deve passar os seguintes parâmetros:

CLIENTE

<**nome-servidor**> - Endereço do dispatcher
<**porta**> - Porta de comunicação do dispatcher

O cliente irá mandar uma requisição **DATA**, **PERGUNTA** ou **FALHA** para o dispatcher. Este irá escolher um servidor em um dos modos aleatório ou Round-Robin, e enviará uma requisição. O servidor então irá processar a requisição e responderá para o dispatcher com um resposta adequada e este irá repassar a mensagem para o cliente.

Estruturas de Dados:

```
typedef struct
{
        struct sockaddr_in ip; // Ip do servidor
        int port; // Porta do servidor
}server; // Estrutura para a lista de servidores
```

Esta é a estrutura utilizada na implementação da lista dos servidores.

Compilar:

Para compilar os programas, basta executar o make.