

Universidade Federal da Fronteira Sul

Campus Chapecó, SC

Curso de Ciência da Computação

CCR: Redes de Computadores – 2022.1

Prof. Marco Aurélio Spohn

Programação Socket e Roteamento

| | |
|-----------------|--|
| Descrição geral | Nesse trabalho você praticará programação com <i>sockets</i> e simulação de um protocolo de roteamento em redes. Você executará processos representando os roteadores (nós) da rede, os quais trocarão pacotes de roteamento via <i>sockets</i> UDP. Na versão final do trabalho, os nós executarão o algoritmo Bellman-Ford distribuído para computar as suas tabelas de roteamento. |
| Observações: | <ul style="list-style-type: none">• O trabalho deve ser implementado em linguagem C• O trabalho deve ser desenvolvido na plataforma Linux• Utilizar somente sockets UDP• Cada NÓ deve executar como um processo (<i>multithread</i>) individual!!!!• A cada submissão, deve-se enviar todos os arquivos necessários para a compilação e execução do sistema. Isto inclui um arquivo do tipo LEIAME com instruções de como utilizar o seu sistema. |
| Modalidade: | <ul style="list-style-type: none">• Individual ou dupla |
| Descrição: | <p>Desenvolva um programa que simule os roteadores de uma rede. O programa deve obter as informações de configuração via arquivos. Cada roteador deve ser capaz de se comunicar com outros roteadores (<i>i.e.</i>, mesmo programa instanciado múltiplas vezes) através de <i>sockets</i> UDP.</p> <p style="text-align: center;"><u>Informações Gerais</u></p> <p>O programa lê pelo menos um parâmetro da linha de comando, sendo este o identificador (ID) do roteador instanciado. Informações sobre os enlaces existentes entre os roteadores são obtidas do arquivo de configuração “<i>enlaces.config</i>”. Informações sobre em quais portas UDP cada roteador está se comunicando com os demais roteadores são obtidas do arquivo de configuração “<i>roteador.config</i>”.</p> <p>Inicialmente, cada roteador conhece apenas os seus vizinhos imediatos e a distância (<i>i.e.</i>, custo dos enlaces) até os mesmos. Os roteadores trocam informações de roteamento utilizando o algoritmo Bellman-Ford distribuído.</p> <p>Os roteadores devem trocar informações de roteamento periodicamente para manter as rotas atualizadas. Assumindo uma rede conectada, as tabelas de roteamento convergem após um determinado tempo.</p> <p>A qualquer momento, os roteadores podem ser “ligados” ou “desligados” (<i>e.g.</i>, criando ou matando os processos correspondentes). O roteamento deve se adaptar a estas situações. Você também deve perceber o problema da contagem ao infinito</p> |

(*count to infinity*). **Observe que você não vai resolver o problema da contagem ao infinito: apenas defina um valor finito (i.e., valor maior que o diâmetro da rede) para parar a contagem, caso ela ocorra!**

Após cada atualização da tabela de rotas, o roteador deve retornar a tabela no console com o *timestamp* da mudança. Ele também deve apresentar uma mensagem quando recebe ou envia pacotes.

Cada roteador se comunica somente com os seus vizinhos imediatos (adjacentes), utilizando os seus respectivos endereços de *sockets* (informação obtida do arquivo de configuração ***roteador.config*** para enviar e receber mensagens.

--> **Incluir a opção de envio** de mensagens de texto, limitadas a 100 caracteres, para qualquer roteador da rede. A mensagem deve ser roteada da origem até o destino segundo a rota computada pelos roteadores. Da origem até o destino, qualquer roteador encaminhando o pacote deve apresentar uma mensagem na tela (e.g., **“Roteador X encaminhando mensagem com # sequência N para o destino Y”**).

Etapas

O projeto será executado em etapas ao longo do semestre letivo.

Etapas 01

Cada roteador será simulado via uma aplicação *multithread* que se comunica com outras instâncias da mesma aplicação (i.e., que representam outros roteadores).

Deve-se definir, pelo menos, as seguintes estruturas de dados:

- **Estrutura (*struct*)** para armazenar mensagens de controle (e.g., trocadas para fins do algoritmo de roteamento) e mensagens de dados. A estrutura deve conter, pelo menos, os seguintes campos:
 - **Tipo de mensagem:** controle ou dado;
 - **Endereços do roteador fonte e roteador destino;**
 - **Carga da mensagem (*payload*):** os dados transportados pela mensagem/pacote; ou seja, se for do algoritmo de roteamento, trata-se de um vetor distância sendo transportado. Caso contrário, será uma mensagem de dados da aplicação (e.g., uma simples *string* de caracteres).
- **Fila de entrada:** para armazenar, temporariamente, todas as mensagens recebidas dos roteadores vizinhos.
- **Fila de saída:** contém as mensagens a serem encaminhadas para os roteadores vizinhos.

LEMBRETE: sempre que houver compartilhamento de dados entre *threads* (e.g., acesso às filas), deve-se tratar o acesso concorrente corretamente (e.g., empregando *mutexes*).

Desenvolver uma aplicação *multithread* que tenha, no mínimo, quatro *threads* (observação: **nem todas os serviços previstos para cada *thread* precisarão ser desenvolvidas nessa etapa**):

- **receiver**: será responsável por processar as mensagens recebidas de outros roteadores vizinhos (i.e., outras instâncias da mesma aplicação representando outros roteadores).
- **sender**: será responsável por enviar mensagens para outros roteadores vizinhos (i.e., outras instâncias da mesma aplicação representando outros roteadores).
- **packet_handler**: responsável por processar as mensagens localmente. Mensagens de controle (roteamento) devem ser processadas pelo algoritmo Bellman-Ford distribuído. Mensagens de dados, devem ser processadas adequadamente. Isto é, caso o roteador atual seja o destino da mensagem, trata-se a mensagem localmente; caso contrário, a mesma deve ser encaminhada para o roteador vizinho segundo informações da tabela de roteamento.
- **terminal**: dedicada ao terminal utilizado pelo usuário da aplicação (deve haver um menu de operações para interação do usuário com a aplicação).

Funcionalidades/Serviços:

- Configuração inicial do roteador a partir dos arquivos de configuração (**roteador.config** e **enlaces.config**).
 - Isto também envolve criar e inicializar o *socket* (UDP) de cada roteador.
- Via terminal, implementar a opção de envio de uma mensagem (pode ser uma simples string) para um roteador **vizinho**. A recepção da mensagem e o seu conteúdo devem ser apresentados no terminal do roteador.

Etapa 02

- Definir as estruturas de dados correspondentes ao **vetor distância**:
 - Os vetores distância recebidos de cada vizinho devem ser mantidos na memória enquanto o vizinho estiver ativo/alcançável (*reachable*).
- Definir a estrutura da tabela de roteamento.

Funcionalidades/Serviços:

- Envio do vetor distância aos roteadores vizinhos:
 - O roteador deve enviar periodicamente o vetor distância a cada um de seus vizinhos (esse tempo deve ser configurável pelo usuário, podendo também ser definido no próprio código fonte).
 - Ao enviar e receber um vetor distância, apresentar seu conteúdo na tela do roteador (também pode ser implementado como uma opção no menu principal, onde o usuário seleciona a opção de visualizar os últimos vetores distância recebidos dos vizinhos).

Etapa 03

Funcionalidades/Serviços:

- Processamento do vetor distância com o protocolo BellMan-Ford distribuído (estudado em aula).
 - Além de enviar periodicamente o vetor distância, toda vez que houver qualquer alteração no vetor distância local, deve-se enviar um novo vetor atualizado para cada um dos roteadores vizinhos.
- Envio de mensagem de dados para qualquer roteador destino, apresentando-se no terminal de todos os roteadores envolvidos (no roteamento) o processo de encaminhamento da mensagem até o seu destino.

OBS.: As três etapas completam o trabalho. A entrega denominada “final” é, de fato, uma extensão de prazo para todos os cenários onde houveram entregas nos prazos das etapas ou nenhuma entrega parcial! INCLUSIVE, é uma oportunidade para entregar uma nova versão com eventuais correções e/ou melhorias.

Dicas

Vocês podem discutir com os demais colegas, mas ***em hipótese alguma compartilhem código!!!***

Teste o seu programa com cenários diferentes. Desative e ative roteadores para observar como a topologia da rede muda. Você pode tentar verificar o resultado primeiro no papel para comparar com os resultados obtidos com o seu programa em execução.

Não espere até o último minuto para começar a trabalhar no seu projeto (mesmo que você seja um programador experiente!).

Arquivos exemplo

roteador.config

O formato do arquivo é (por linha): identificador do roteador (inteiro), número da porta e número IP (**espaçamento livre entre parâmetros**).

O programa recebe na entrada o identificador do roteador em questão, basta então ler do arquivo de entrada o seu IP e qual a porta do seu *socket*.

Exemplo:

```
1 25001 127.0.0.1
2 25002 127.0.0.1
3 25003 127.0.0.1
4 25004 127.0.0.1
5 25005 127.0.0.1
6 25006 127.0.0.1
```

enlaces.config

O formato de cada linha do arquivo é:

ID ID custo (**espaçamento livre entre parâmetros**).

Ou seja, o identificador dos dois roteadores conectados e o custo do enlace. Assume-se que os enlaces são simétricos (bidirecionais).

Exemplo:

| | | |
|---|---|----|
| 1 | 2 | 10 |
| 1 | 3 | 15 |
| 2 | 4 | 2 |
| 2 | 5 | 5 |
| 3 | 4 | 2 |
| 4 | 6 | 10 |
| 5 | 6 | 5 |

Documentação

O código fonte deve ser devidamente comentado. Também deve incluir um arquivo com instruções **DETALHADAS** de como operar o programa.

O que e quando submeter

O que:

- **Em cada etapa:** submeter (via *moodle*) um arquivo comprimido com todos os arquivos necessários para a execução da versão atual incremental (i.e., incluindo todas as *features* desenvolvidas até a etapa atual). **OBRIGATORIAMENTE** incluir um arquivo “README” com instruções para compilação e uso do sistema.
- **Versão final:** ao final do semestre letivo, submeter (via *moodle*) um arquivo compactado com todos os arquivos necessários para a execução do sistema final. **OBRIGATORIAMENTE** incluir um arquivo “README” com instruções para compilação e uso do sistema.

Quando:

- **Etapa 1:** 12/06/2022
- **Etapa 2:** 03/07/2022
- **Etapa 3:** 14/08/2022
- **Versão final:** 21/08/2022
- **Apresentação final:** 22 e 29/08/2022