Universidade Federal da Fronteira Sul

Curso de Ciência da Computação

Disciplina: Redes de Computadores – 2015.2

Prof. Marco Aurélio Spohn

Projeto: Programação *Socket* e Roteamento

Descrição geral	Neste projeto você irá praticar programação com *sockets* e simular/emular um protocolo de roteamento em redes. Você irá executar processos representando os roteadores (nós) da rede, os quais irão trocar pacotes de roteamento via *sockets* UDP. Na versão final do trabalho, os nós executarão o algoritmo **Bellman-Ford distribuído** para computar as suas tabelas de roteamento. **LEIA ATENTAMENTE A DESCRIÇÃO DO PROJETO. IGNORÂNCIA NÃO É DESCULPA!!!!**
Observações:	• **O projeto deve ser implementado em linguagem C.** • **O projeto deve ser desenvolvido SOMENTE na plataforma Linux!** • Utilizar somente *sockets* UDP! • **Cada NÓ deve executar como um processo individual!!!! (obs.: no entanto cada processo pode, caso desejável, ser *multithread*)** • Submeter 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.
Descrição:	Desenvolva um programa que simule os roteadores de uma rede. O programa deve obter as informações de configuração de arquivos. Cada roteador deve ser capaz de conversar com outros roteadores (*i.e.*, mesmo programa instanciado múltiplas vezes) através de *sockets* UDP. O projeto está dividido em duas etapas: a) **Etapa 1**: correspondente à componente avaliativa T1; b) **Etapa 2**: correspondente à componente avaliativa T2. **Informações Gerais** O programa lê pelo menos um parâmetro da linha de comando, sendo este o ID do roteador sendo instanciado. Informação sobre os enlaces existentes entre os roteadores são obtidas do arquivo de configuração (*enlaces.config*). Informações sobre em quais portas UDP cada roteador está se comunicando com os demais roteadores são obtidas do arquivo de configuração (*roteador.config*).

	Etapa 1 (Trabalho 1) Nessa etapa, assume-se que os roteadores conhecem a **topologia completa** da rede. Utilizando-se soluções da teoria dos grafos, cada nó computa o caminho completo para todos os destinos possíveis na rede. Nessa etapa, deve-se assumir que a topologia permanece constante (i.e., não sofre qualquer mudança) ao longo da vida da rede. Implementar uma aplicação de transmissão confiável de mensagens de texto, limitadas a 100 caracteres. A mensagem deve ser roteada da origem até o destino segundo o caminho computado por cada roteador. 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"). **Etapa 2 (Trabalho 2)** Inicialmente, cada roteador conhece apenas os seus vizinhos imediatos e a distância (i.e., custo dos enlaces) até os mesmos. Os roteadores trocam informações de roteamento utilizando o algoritmo **Bellman-Ford** distribuído. 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. A qualquer momento, os roteadores podem ser "ligados" ou "desligados" (criando ou matando os processos correspondentes). O roteamento deve se adaptar a estas situações. Você também deve tratar o problema de contagem ao infinito (*count to infinity*). **Observe que você não vai resolver o problema da contagem ao infinito (como exemplo de um protocolo vetor distância sem o problema da contagem ao infinito verifique o protocolo DUAL por J.J. Garcia-Luna-Aceves), apenas defina um valor finito (i.e., valor > 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.
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 (os programas que os representam) 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 acompanhado de documentação. Também deve incluir um arquivo com instruções **DETALHADAS** de como operar o programa.
O que e quando submeter	**O que**: para cada etapa, submeter (via *moodle*) um arquivo compactado (somente formato *tar*) com todos os arquivos necessários para a execução do sistema. **Quando**: • **Etapa 1: 12/10/2015** • **Etapa 2: 30/11/2015** **Após o prazo de entrega de cada trabalho, definir-se-á uma data para apresentação!!**