Algoritmo de Roteamento

Teleinformática e Redes 2 (TR-2)

Jacir Luiz Bordim
Departamento de Ciência da Computação (CIC)
Universidade de Brasília (UnB)

5 de setembro de 2016

– Algoritmos e Protocolos de Roteamento

Esta tarefa visa estudar e implementar os algoritmos e protocolos de roteamento considerados nos capítulos 04 e 02 dos livros do Kurose et al. [1] e do Medhi et al. [2], respectivamente. O trabalho pode ser feito em dupla ou individual. Para cada algoritmo implementado, deve-se observar os seguintes requisitos:

1. A linguagem de programação: C/C++ ou Python 2.7 (Linux de preferência). As funções (algoritmos de roteamento, devem ser implementados pelos grupos, ou seja, funções prontas não serão aceitas);

2. O programa deve receber como entrada (argumento de linha):

   (a) **Topologia:** a topologia da rede pode ser vista como um grafo $G(V, E)$, onde V representa o conjunto dos nós e E o conjunto das arestas. Cada aresta $(u, v) = p$, onde $p > 0$, representa o peso da aresta conectando o nó u ao nó v. Cada nó deve receber como entrada a relação dos seus vizinhos diretamente conectados bem como o seu custo.

   Exemplo:

   $$nodo; vizinho1[custo]; vizinho2[custo]; \cdots, vizinhoN[custo]$$

   $$1; 2[4]; 3[1];$$

   $$2; 1[4]; 3[2];$$

   $$3; 1[1]; 2[3]$$

3. O programa deve fornecer como saída:

   (a) **Lista de destinos e seu custo**: A saída do seu programa deve fornecer, para cada nó, a lista de destinos bem como o custo para cada nó;

   (b) **Convergência**: Tendo em vista que a sua implementação é distribuída [1], o seu programa deverá mostra o tempo de convergência e terminar a execução após convergir (isto é, todos os nós terem a mesma informação).

[1] Você terá que pensar em como fazer isso é claro! Threads pode ser uma alternativa, mas claro, há outras

(c) **Valiadação**: A sua implementação deve apresentar mecanismo para mostrar, pelo menos para um nó selecionado, as trocas de informações para fins de visualização/validação.

4. Exemplos: o trabalho deve acompanhar de pelo menos 3 topologias de exemplo. Cada topologia deve conter um número crescente de nós e enlaces (e seus respectivos custos/vizinhos). A topologia mínima deve conter pelo menos 6 nós, a segunda pelo menos 12, a terceira pelo menos 18. O diâmetro do seu grafo deve ser superior a 4, 5, e 7, respectivamente, para cada topologia.

5. Documentação:

   (a) Cada grupo deve informar as restrições (limitações) do programa. Toda a função, ou código que não seja declarações devem ser comentadas. Da mesma forma, cada programa deve conter informações de como compilar e executar.

   (b) A documentação pode ser fornecida via LEIA.TXT (como compilar, executar os exemplos e detalhar o formato da lista de adjacência, e demais parâmetros de entrada, restrições etc). As demais informações devem ser inseridas no código (comentários sobre a implementação do algoritmo).

6. Exemplos Prontos: Encontrar um exemplo da Internet, livro ou alguma outra fonte e adaptar é válido. Agora, você deverá informar a fonte que utilizou como exemplo (seja o livro que contém a descrição do algoritmo e/ou código utilizado como base). Esta informação é **obrigatória**. Da mesma forma, o custo computacional do seu algoritmo deve ser informado;

7. Cópia: Copiar do seu colega, adaptar do seu colega não será permitido. Como não saberei quem copiou de quem, todos serão penalizados, sem exceção.

8. Prazo de entrega: 1 semana (via aprender). O grupo deverá apresentar em sala o código e exemplos funcionando.

Lista dos algoritmos a serem implementados e os respectivos grupos

1. Path vector;
2. Link state;
3. Distance Vector;

Referências Bibliográficas

[1] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach (6th Edition)*. Pearson, 6th edition, 2012.

[2] Deepankar Medhi and Karthikeyan Ramasamy. *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.