TRABALHO PRÁTICO #03:

ROTEAMENTO
POR
VETOR DE DISTÂNCIA

Trabalho Prático #03

- Conforme explicações e exemplos vistos em aula, implementar o algoritmo de **roteamento por vetor de distância**

- Relembrando as principais características do algoritmo
  - ✓ Cada roteador mantém uma tabela de roteamento indexada por cada roteador da sub-rede, com entrada para cada um destes roteadores
  - ✓ A tabela é composta de duas partes:
    - A linha de saída preferencial a ser utilizada para determinado destino
    - Métrica para alcançar o destino (será usado retardo de tempo em ms)
    - Para saber qual é o retardo até cada um de seus vizinhos, o host utiliza um PING que simplesmente irá ler as informações contidas no arquivo **backbone.txt**
    - As tabelas são atualizadas através de trocas de informações com os vizinhos **a cada T ms**
  - ✓ Iniciado seu programa, os nós deverão trocar informações **com seus vizinhos** (a cada T ms) e assim sucessivamente
    - A fase de preenchimento das tabelas acaba quando TODAS as tabelas forem preenchidas COMPLETAMENTE
    - A quantidade de interações necessárias variará de acordo com a complexidade do *backbone*

Trabalho Prático #03

- Relembrando as principais características do algoritmo (CONTINUA)
  - ✓ Quando as tabelas forem totalmente preenchidas, os pacote poderão circular pela sub-rede
  - ✓ Neste ponto seu programa deverá:
    - Permitir ao usuário escolher GRAFICAMENTE uma **ORIGEM** e um **DESTINO** na GUI
    - Mostrar **GRAFICAMENTE** na GUI qual será a rota percorrida
      - Esta informação obviamente virá a partir dos dados calculados nas tabelas de roteamento
    - Também deverá ser possível ao usuário **GRAFICAMENTE** desabilitar um *link*/aresta
    - Neste momento seu programa deverá voltar ao ponto inicial, ou seja:
      - Indicar na GUI tal situação E redesenhar a sub-rede
      - Recalcular as tabelas de roteamento para refletir tal situação
      - Permitir ao usuário novas interações com a aplicação
  - ✓ Aproveite seu simulador para testar o problema da contagem até o infinito!

Trabalho Prático #03

- Conforme explicações e exemplos vistos em aula, implementar o algoritmo de **roteamento por vetor de distância**

- Considerações
  - ✓ Utilizar GUI
  - ✓ Permitir a escolha na GUI do TRANSMISSOR e do RECEPTOR
  - ✓ Ao final do algoritmo indicar **GRAFICAMENTE** na GUI qual foi a rota escolhida
  - ✓ A representação da sub-rede será **FLEXÍVEL**
    - As informações da estrutura da sub-rede estarão em um arquivo texto
    - Nome do arquivo será **backbone.txt** e estará na raiz da pasta principal
    - Formato do arquivo no próximo slide

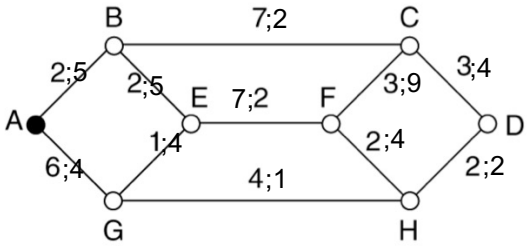
Trabalho Prático #03

➪ Formato do arquivo

- ✓ 1a linha.....: contém o número de nós da sub-rede
- ✓ 2a linha até a última: contém a conexão entre os nós e tempo de retardo de ida e volta, separados por ponto-vírgula
  - Ex: **NO1;NO2;VALOR1;VALOR2**
    significa que o NO1 está conectado ao NO2 com a métrica VALOR1 (IDA) e o NO2 está conectado com o NO1 com métrica VALOR2 (VOLTA)
  - Um exemplo pode ser observado abaixo

IDA = *echo request*
VOLTA = *echo reply*
} PING

Grafo da sub-rede



ping 1->2=2ms
ping 2->1=5ms
ping 2->3=7ms
ping 3->2=2ms
ping 3->4=3ms
ping 4->3=6ms
E asim sucessivamente.

backbone.txt

```
8;
1;2;2;5
1;7;6;4
2;3;7;2
2;5;2;5
3;6;3;9
3;4;3;4
4;8;2;2
5;6;7;2
5;7;1;4
6;8;2;4
7;8;4;1
```

OBSERVAÇÕES:

- As ligações entre os nós são representadas apenas uma vez no arquivo texto. Assim **a linha 1;2;2;5** indica que há um caminho entre o nó 1 e o nó 2 com tempo de retardo igual a 2 no sentido 1->2, e TAMBÉM que a há um caminho entre o nó 2 e o nó 1 com tempo de retardo igual a 5 no sentido 2->1
- Apesar dos nós estarem representados numericamente no arquivo texto, a representação do *label* correspondente em sua GUI pode seguir outro padrão (alfa)numérico. Exemplo: A, B, C, etc ou No1, No2, No3, etc
- O final do arquivo indica o fim da estrutura da sub-rede