Relatório

Esse código é uma implementação básica do Algoritmo de Vetor de Distância, que é usado para encontrar o caminho mais curto entre os nós em uma rede. Vou descrever como o código funciona e, em seguida, fornecer um cenário para uma prova de conceito (POC) que ilustra a adição e remoção de nós na rede.

Funcionamento do Código

1. Inicialização:

- O nó (roteador) é inicializado com um identificador (`nodeId`) e uma lista de vizinhos (`neighbors`), que são passados como argumentos na linha de comando.

- O código define o número total de nós (`N`) e a matriz de custos (`c`) entre todos os pares de nós.

2. Inicialização do Vetor de Distâncias:

- O vetor de distâncias (`D`) é inicializado com `Infinity` para todos os nós, exceto para o próprio nó, que tem distância 0.

3. Envio do Vetor de Distâncias:

- A função `sendVector` cria uma mensagem contendo o identificador do nó e o vetor de distâncias atual.

- A mensagem é enviada para todos os vizinhos do nó usando o protocolo UDP.

4. Processamento de Mensagens:

- Quando o nó recebe uma mensagem de outro nó, ele processa a mensagem para atualizar seu vetor de distâncias.

- Se o vetor de distâncias do nó muda, ele reenvia o vetor atualizado para seus vizinhos.

5. Servidor UDP:

- O servidor UDP escuta na porta 41234 e chama a função `processMessage` quando recebe uma mensagem.

- Inicialmente, o nó envia seu vetor de distâncias para os vizinhos quando começa a escutar.

Cenário de Prova de Conceito (POC)

Configuração Inicial

Suponha que temos 5 nós com a seguinte configuração inicial:

- Nó 0: vizinhos [1, 4]

- Nó 1: vizinhos [0, 2, 3]

- Nó 2: vizinhos [1, 3, 4]

- Nó 3: vizinhos [1, 2, 4]

- Nó 4: vizinhos [0, 2, 3]

- O código será executado em cada nó com parâmetros diferentes:

- `nodeId` e `neighbors` serão passados como argumentos ao executar o script para cada nó.

Teste 1: Adição de Nós

1. Início da Simulação:

- Todos os nós são iniciados com a configuração inicial.

2. Adição de um Novo Nó (Nó 5):

- O Nó 5 é adicionado à rede com vizinhos [2, 3].

- Atualize a matriz de custos (`c`) para refletir as novas distâncias.

- O Nó 5 se comunica com o Nó 2 e o Nó 3, e ambos os nós atualizam seus vetores de distâncias e enviam as atualizações para seus vizinhos.

3. Observação:

- Observe como o nó 5 começa a receber e enviar vetores de distâncias e como os vetores de distâncias dos nós existentes são atualizados para refletir o novo nó.

Teste 2: Remoção de Nós

1. Remoção de um Nó (Nó 3):

- O Nó 3 é removido da rede.

- Atualize a matriz de custos (`c`) para refletir a remoção do Nó 3 e ajuste os vizinhos dos outros nós para remover referências ao Nó 3.

2. Atualização da Rede:

- Os nós vizinhos do Nó 3 (Nó 1, Nó 2, e Nó 4) detectam a remoção e atualizam seus vetores de distâncias.

- Os vetores de distâncias são recalculados e propagados pela rede.

3. Observação:

- Observe como a rede se ajusta para refletir a remoção do Nó 3 e como os vetores de distâncias são atualizados para garantir a consistência das rotas.

Conclusão

Este código é uma implementação básica e pode ser expandida para incluir mais recursos, como a detecção de falhas e a atualização dinâmica de vizinhos. A prova de conceito demonstra como o algoritmo de vetor de distância lida com a adição e remoção de nós e como o sistema converge para um estado consistente de rotas mais curtas.