Relatório

O código fornecido é uma implementação básica de um protocolo de roteamento que usa a abordagem de Vetor de Distância para descobrir e atualizar rotas entre roteadores em uma rede. Vou descrever o funcionamento do código e apresentar um cenário de prova de conceito (POC) para demonstrar a adição e remoção de nós durante o processo de descoberta e roteamento.

Funcionamento do Código

1. Inicialização dos Roteadores:

- Cada roteador é representado pela classe Router, que estende EventEmitter para permitir a comunicação entre roteadores.

- O construtor da classe Router recebe um id (identificador do roteador) e uma lista de vizinhos (neighbors), que são instâncias de outros roteadores.

2. Inicialização da Tabela de Roteamento:

- initializeRoutingTable: Inicializa a tabela de roteamento para cada roteador.

    - Cada roteador define sua própria rota direta para seus vizinhos e a rota para si mesmo.

    - Envia a tabela de roteamento inicial para todos os vizinhos através da função broadcastRoutingTable.

3. Recepção e Atualização da Tabela de Roteamento:

- receiveRoutingTable: Recebe a tabela de roteamento de um vizinho.

    - Atualiza a tabela de roteamento se encontrar um caminho mais curto.

    - Se a tabela for atualizada, retransmite a tabela atualizada para todos os vizinhos.

4. Broadcasting:

- broadcastRoutingTable: Envia a tabela de roteamento atual para todos os vizinhos para que eles possam atualizar suas tabelas de roteamento.

Cenário de Prova de Conceito (POC)

Configuração Inicial

Considere a seguinte topologia inicial de rede:

- Roteador A tem vizinhos B e C.
- Roteador B tem vizinhos A, C e D.
- Roteador C tem vizinhos A, B e D.
- Roteador D tem vizinhos B e C.

```
const routerA = new Router('A', []);
const routerB = new Router('B', [routerA]);
const routerC = new Router('C', [routerA, routerB]);
const routerD = new Router('D', [routerB, routerC]);

routerA.neighbors = [routerB, routerC];
routerB.neighbors = [routerA, routerC, routerD];
```

```
routerC.neighbors = [routerA, routerB, routerD];
routerD.neighbors = [routerB, routerC];

routerA.initializeRoutingTable();
routerB.initializeRoutingTable();
routerC.initializeRoutingTable();
routerD.initializeRoutingTable();
```

Teste 1: Adição de um Novo Nó

1. Adição do Roteador E:

- Adicionamos um novo roteador E conectado a C e D com uma distância específica.

```
const routerE = new Router('E', [routerC, routerD]);
routerC.neighbors.push(routerE);
routerD.neighbors.push(routerE);

// Inicializa a tabela de roteamento para o novo roteador
routerE.initializeRoutingTable();
```

2. Atualização das Tabelas de Roteamento:

- Todos os roteadores existentes (A, B, C, D) precisam atualizar suas tabelas de roteamento com a informação do novo roteador E.

- Observe como as rotas para o novo roteador E e as rotas através de E são atualizadas e propagadas.

3. Observação:

- Verifique como os roteadores ajustam suas tabelas de roteamento para incluir o novo

roteador E e como isso afeta as rotas entre todos os roteadores.

Teste 2: Remoção de um Nó

1. Remoção do Roteador B:

- Removemos o roteador B da rede e atualizamos as conexões.

```
routerA.neighbors = routerA.neighbors.filter(n => n.id !== 'B');
routerC.neighbors = routerC.neighbors.filter(n => n.id !== 'B');
routerD.neighbors = routerD.neighbors.filter(n => n.id !== 'B');

delete routerB;
```

2. Atualização das Tabelas de Roteamento:

- Os roteadores A, C, e D precisam atualizar suas tabelas de roteamento para refletir a ausência do roteador B.

- Observe como a remoção do roteador B afeta as rotas e como isso é propagado pelos roteadores restantes.

3. Observação:

- Verifique como as tabelas de roteamento dos roteadores restantes são ajustadas para refletir a remoção do roteador B e como as rotas são recalculadas.

Conclusão

Este código é uma implementação básica de um protocolo de roteamento baseado em Vetor de Distância. Ele demonstra como roteadores descobrem e atualizam rotas em uma rede dinâmica, considerando a adição e remoção de nós. A prova de conceito ilustra a capacidade do sistema de adaptar-se às mudanças na topologia da rede e manter tabelas de roteamento atualizadas e consistentes.