Instituto de computação

Programa de Pós-Graduação
Instituto de Computação
Universidade Federal Fluminense

**Aula 17**

# Camada de Rede

## Roteamento intradomínio: vetor de distância e RIP

Igor Monteiro Moraes

Redes de Computadores

# ATENÇÃO!

Instituto de computação

- Este apresentação é baseada nas notas de aula do Prof. Luís Henrique M. K. Costa, disponíveis em http://www.gta.ufrj.br/ensino/CPE825/cpe825.html

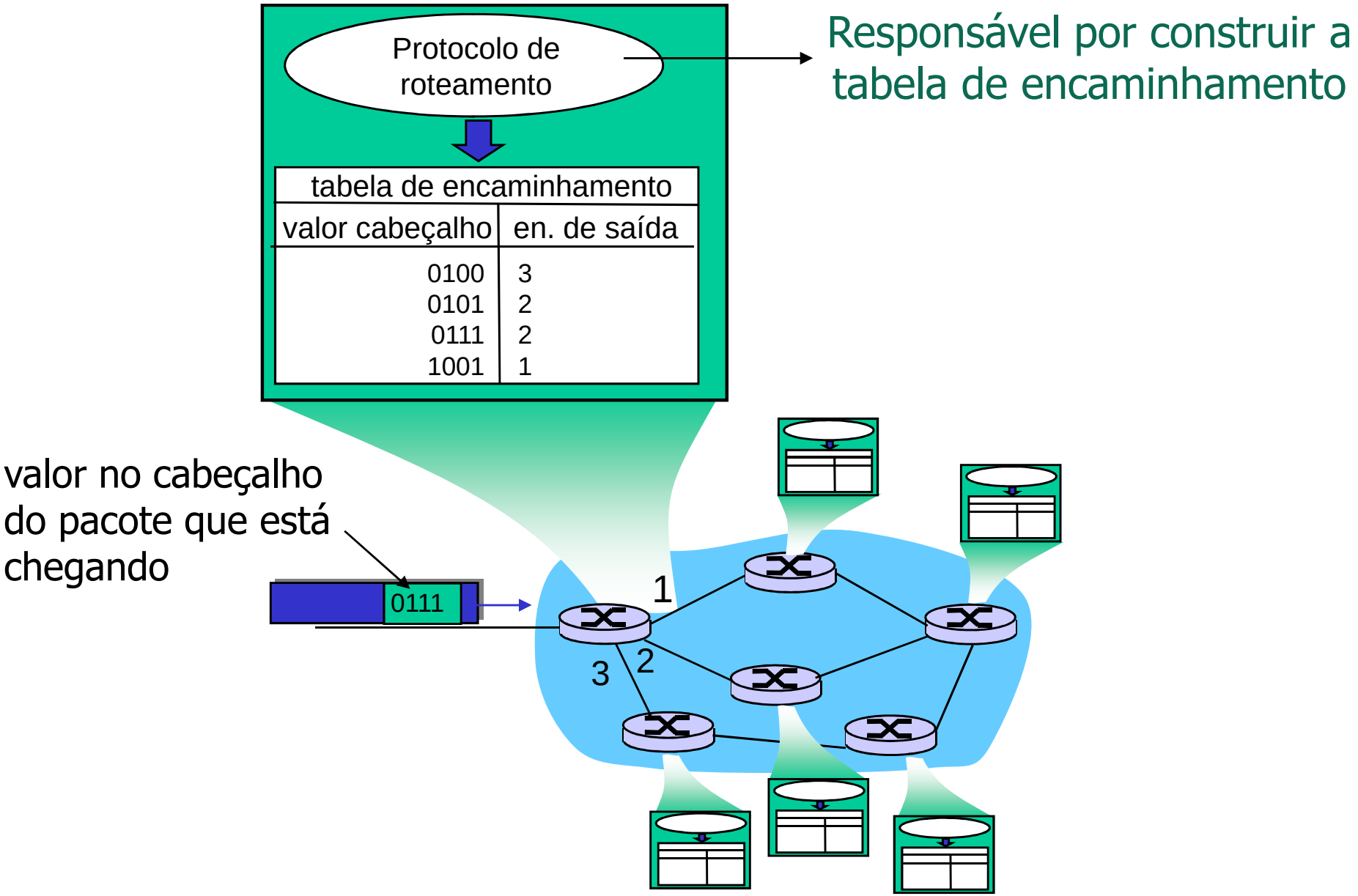
- Material complementar do livro Computer Networking: A Top Down Approach, 5th edition, Jim Kurose and Keith Ross, Addison-Wesley, abril de 2009

# Camada de Rede

Instituto de computação

- Responsável por
  - **Determinar o melhor caminho** para o envio dos pacotes
    - É função dos **protocolos de roteamento**
  - **Encaminhar** os pacotes até o destino
    - É função do protocolo IP
  - **Interconectar** redes de diferentes tecnologias
    - É função do protocolo IP

# Encaminhamento x Roteamento



# Importância do Roteamento

Instituto de Computação

- Uma rede conectada é suficiente para que dois nós se comuniquem?
  - Sim, a comunicação pode ser realizada por inundação
    - Todos os nós enviam mensagens para todos os vizinhos até que o destino seja alcançado
- Essa técnica é eficiente? É escalável?
  - NÃO!
    - O roteamento é responsável por encaminhar os dados até o destino de maneira eficiente
    - Os protocolos de roteamento encontram o caminho ótimo (ou quase-ótimo) pelo qual os dados são encaminhados

# Algoritmos de Roteamento

Instituto de computação

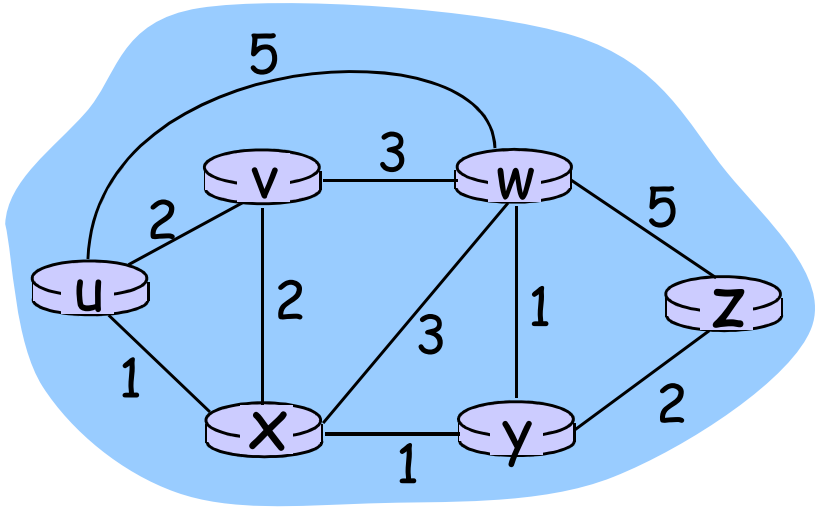
- Objetivo
  - Descobrir o caminho mais curto (*shortest path* – SP) entre qualquer par de nós da rede
- Tabela de roteamento
  - Cada entrada possui
    - Destino da rota
    - Próximo salto
    - Métrica

# Algoritmos de Roteamento

Instituto de Computação

- Protocolos
  - Vetores de distância (*Distance Vector* – DV)
    - Algoritmo de Bellman-Ford
  - Estado do enlace (*Link State* – LS)
    - Algoritmo de Dijkstra

# Abstração com grafos

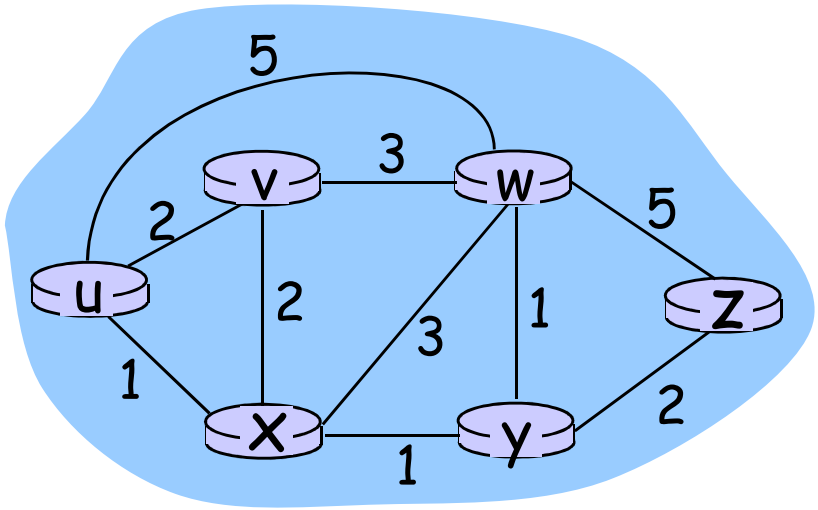



Grafo: G = (N,E)

N = conj. de roteadores = { u, v, w, x, y, z }

E = conj. de enlaces =
=={(u,v), (u,x), (u,w), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z)}

# Abstração com grafos: custos




- c(x,x') = custo do enlace (x,x')

  - p.e., c(w,z) = 5

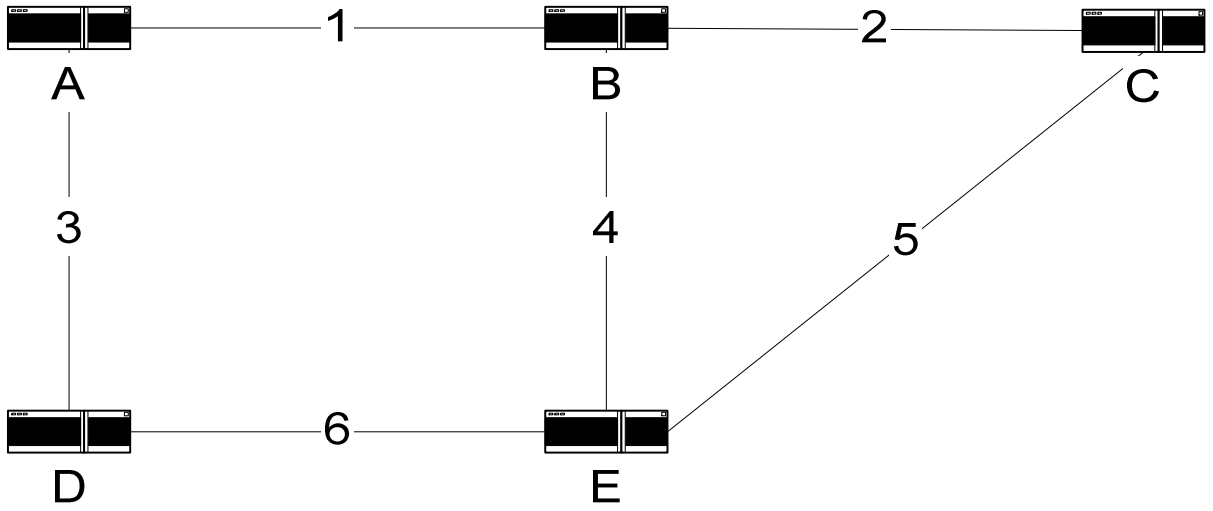
- custo poderia também ser 1, ou inversamente relacionado à banda, ou inversamente relacionado ao congestionamento

Custo do caminho $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

# Topologia

5 roteadores: de **A** a **E**

6 enlaces: de **1** a **6**




Suponha que todos os enlaces possuem custo igual a 1

# Algoritmo de Bellman-Ford

Instituto de computação

- Equação de Bellman-Ford

  $d_x(y)$ → custo do caminho de menor custo entre x e y

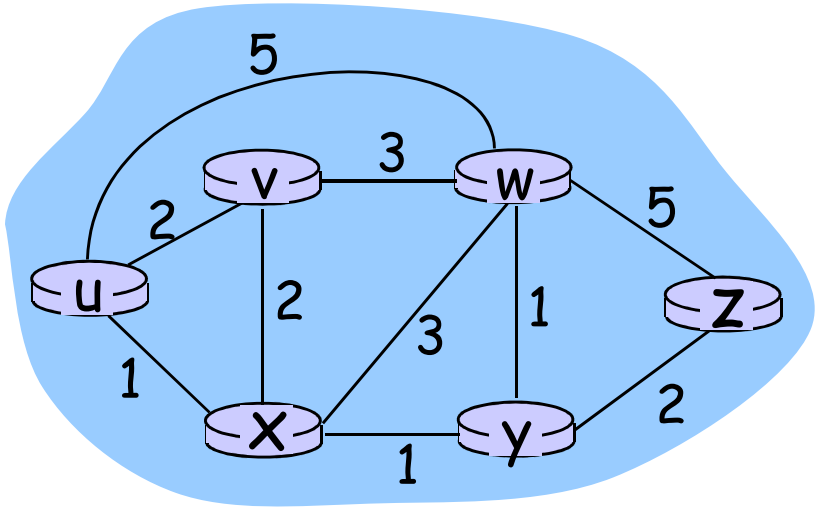
- Logo

$$d_x(y) = \min_v\{c(x,v) + d_v(y)\},$$

onde min é tomado entre todos os vizinhos v de x

# Exemplo com Bellman-Ford

U→Z?




$d_v(z) = 5,\ d_x(z) = 3,\ d_w(z) = 3$

A equação B-F diz:

$$d_u(z) = \min \{c(u,v) + d_v(z),\ c(u,x) + d_x(z),\ c(u,w) + d_w(z)\}$$
$$= \min \{2 + 5,\ 1 + 3,\ 5 + 3\} = 4$$

O nó que leva ao custo mínimo é o próximo passo
ao longo do caminho mais curto➜ tabela de encaminhamento

# Algoritmo de Bellman-Ford

Instituto de computação

- $D_x(y)$ = estimativa do menor custo entre x e y
- Nó x sabe o custo para cada vizinho v: $c(x,v)$
- Nó x mantém o vetor de distâncias $\mathbf{D}_x = [D_x(y): y \in N ]$
- Nó x mantém ainda os vetores de distâncias dos seus vizinhos
  - Para cada vizinho v, x mantém

    $\mathbf{D}_v = [D_v(y): y \in N ]$

# Algoritmo de Bellman-Ford

Instituto de computação

- Cada nó envia periodicamente o seu próprio vetor de distâncias estimadas para os vizinhos
- Quando um nó x recebe um novo DV com as estimativas de um vizinho, ele atualiza o seu DV usando a equação B-F

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{p/ cada nó } y \in N$$

- Sob condições mínimas, naturais, a estimativa $D_x(y)$ converge para o menor custo real $d_x(y)$

# Algoritmo de Bellman-Ford

Instituto de computação

## Iterativo, assíncrono

Cada iteração local causada por

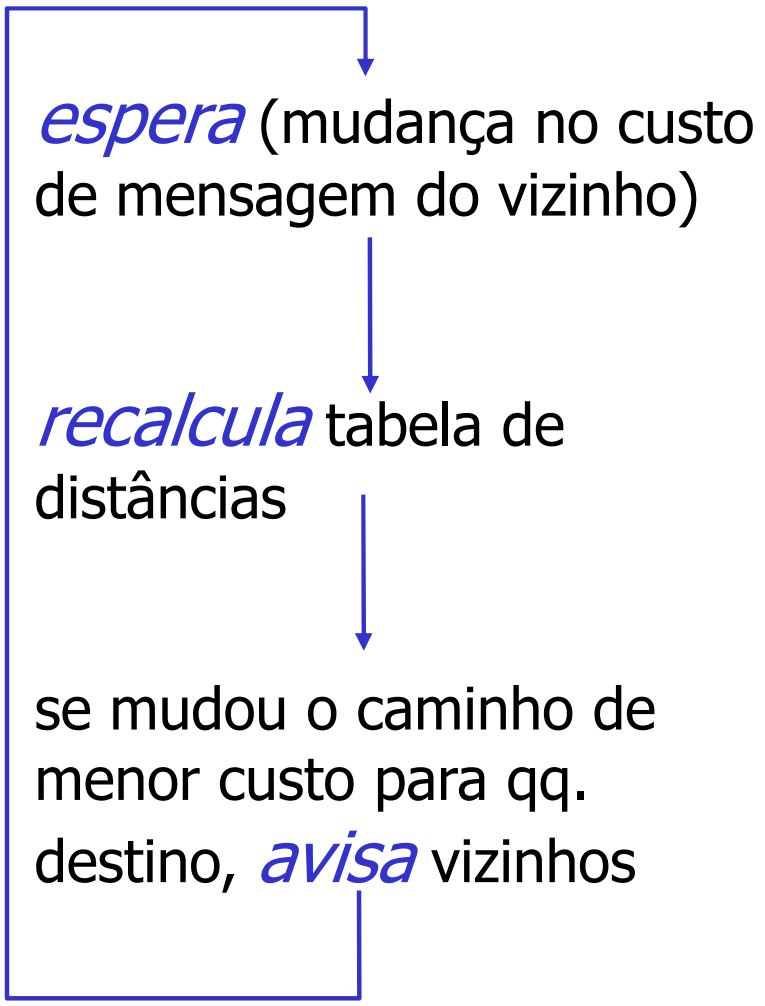
- Mudança do custo do enlace local
- Mensagem do vizinho: mudança de caminho de menor custo para algum destino

## Distribuído

- Cada nó avisa a seus vizinhos *apenas* quando muda seu caminho de menor custo para qualquer destino
  - Os vizinhos então avisam a seus vizinhos, se for necessário

## Complexidade: $O(N^2)$

## Cada nó:



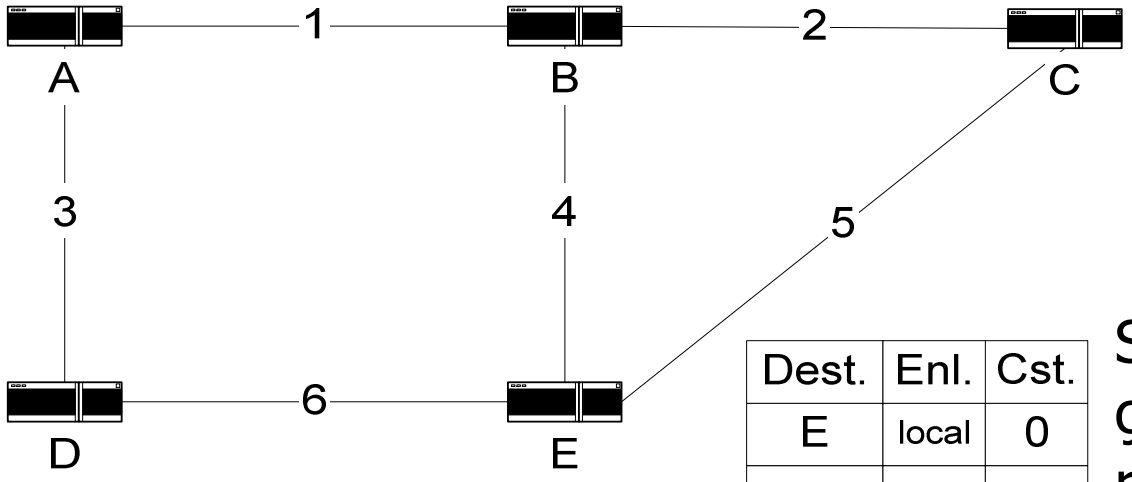
# Exemplo – DV

A rede é ligada

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| | | |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| | | |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| | | |
| | | |
| | | |
| | | |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| | | |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| | | |
| | | |
| | | |
| | | |

Só existe informação **local** em cada roteador

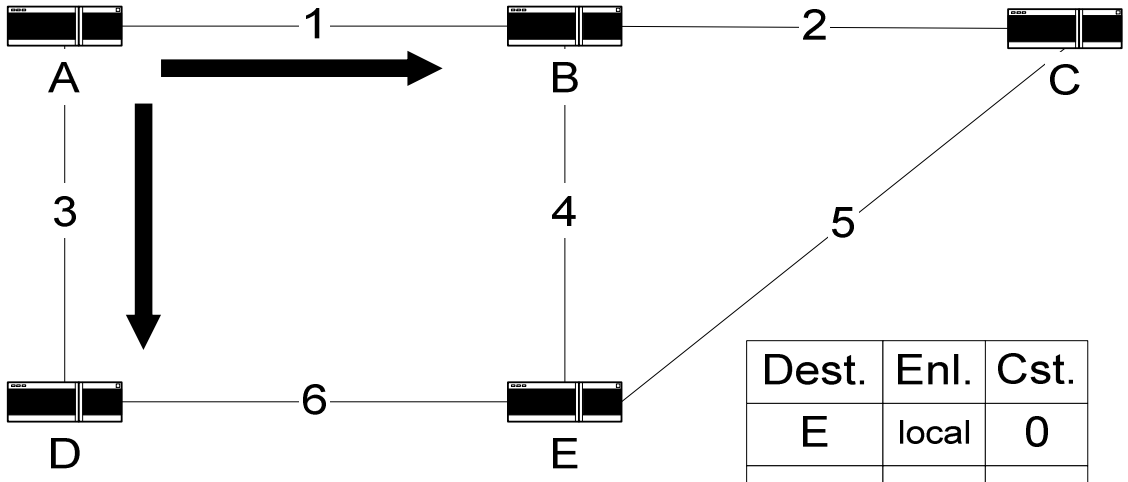
# Exemplo – DV

**A** envia um vetor de distância

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| | | |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| | | |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| | | |
| | | |
| | | |
| | | |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| | | |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| | | |
| | | |
| | | |
| | | |

**(A = 0)**

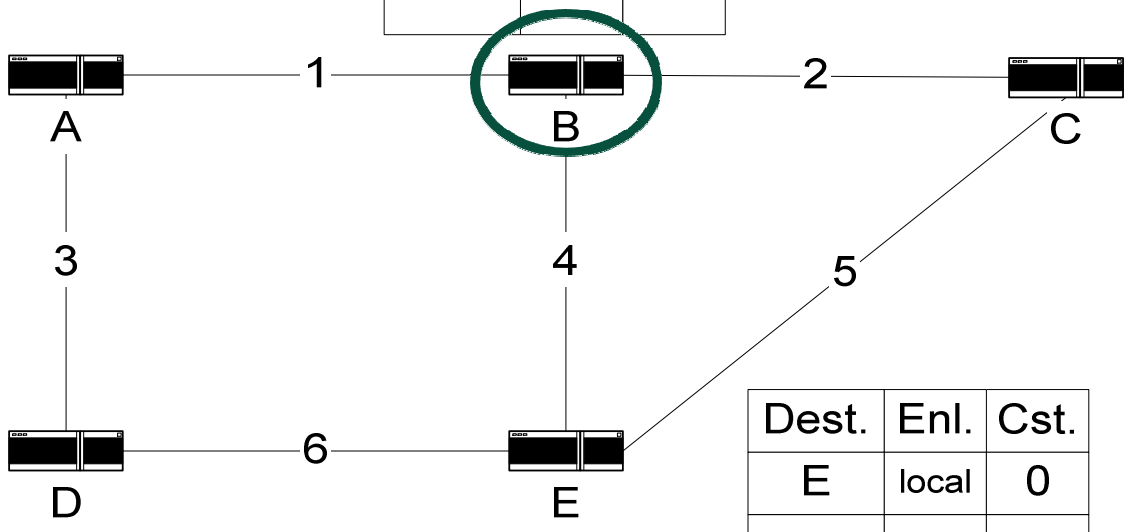
# Exemplo – DV

**B** recebe o vetor de distância de **A**.

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| | | |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | 1 |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| | | |
| | | |
| | | |
| | | |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| | | |
| | | |
| | | |
| | | |

**B** soma **1** (custo do enlace **1**) ao vetor de distância, obtendo **A=1**. Esta é uma nova entrada, então **B** atualiza sua tabela.
Idem para **D**

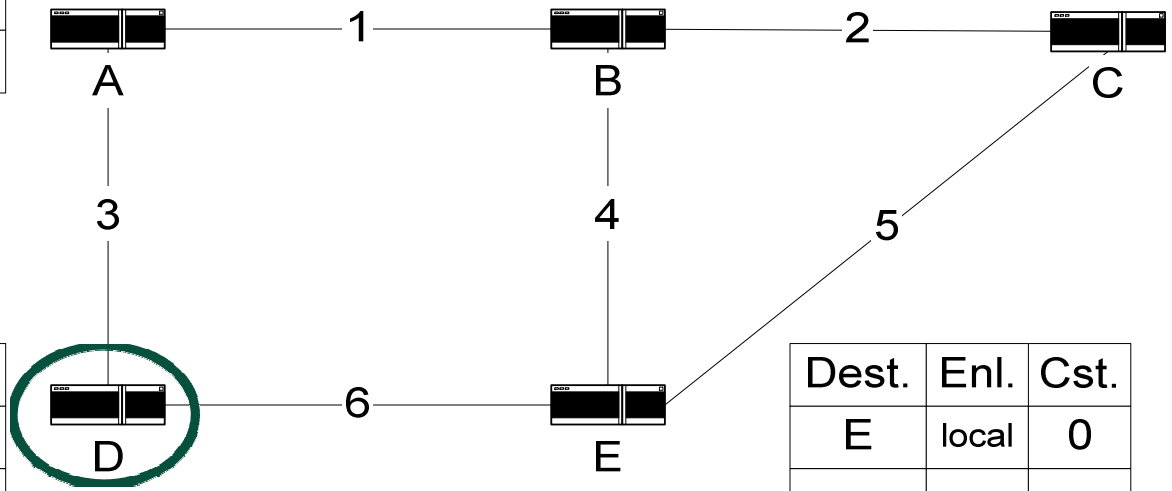
# Exemplo – DV

**D** recebe o vetor de distância de **A**

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| | | |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | 1 |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| | | |
| | | |
| | | |
| | | |




Idem para **D**

| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| | | |
| | | |
| | | |
| | | |

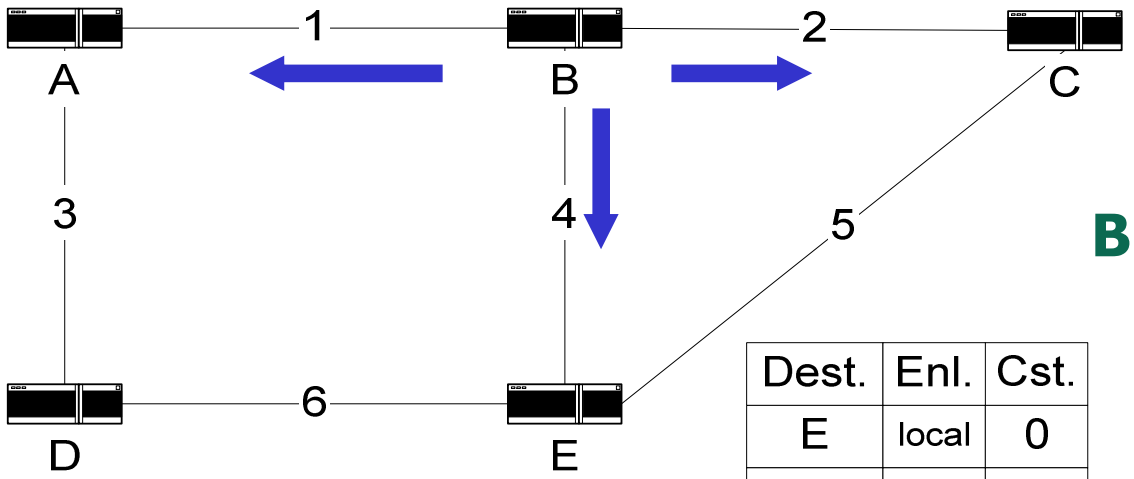
# Exemplo – DV

**B** e **D** preparam seus vetores de distância

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| | | |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | 1 |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| | | |
| | | |
| | | |
| | | |




**B** envia primeiro

**(B=0, A = 1)**

| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| | | |
| | | |
| | | |
| | | |

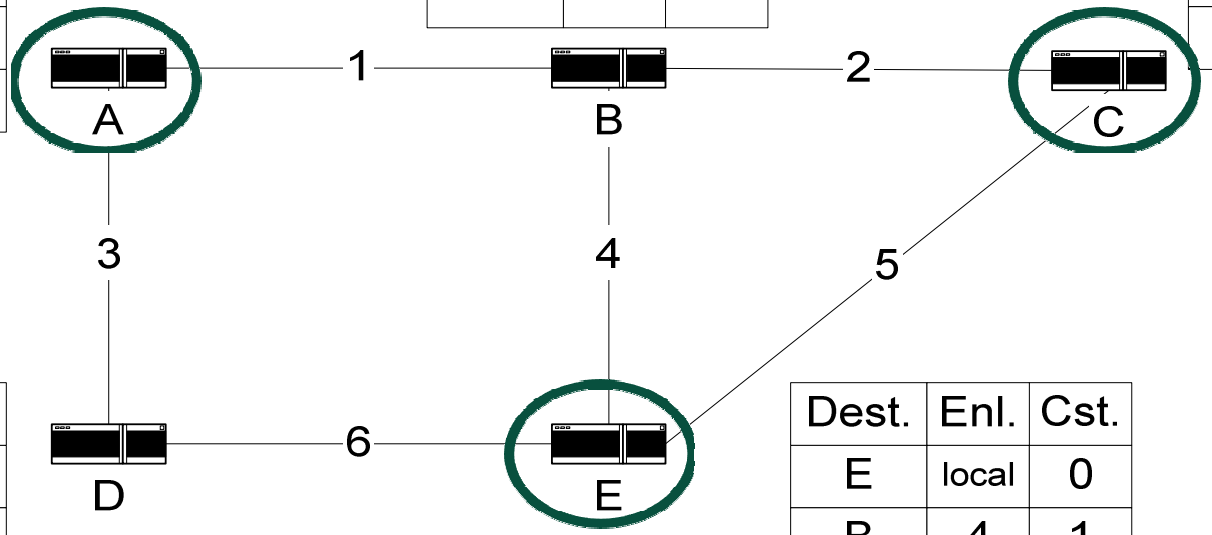
# Exemplo – DV

**A**, **C** e **E** atualizam suas tabelas

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | 1 |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | 1 |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| | | |
| | | |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| | | |
| | | |
| | | |

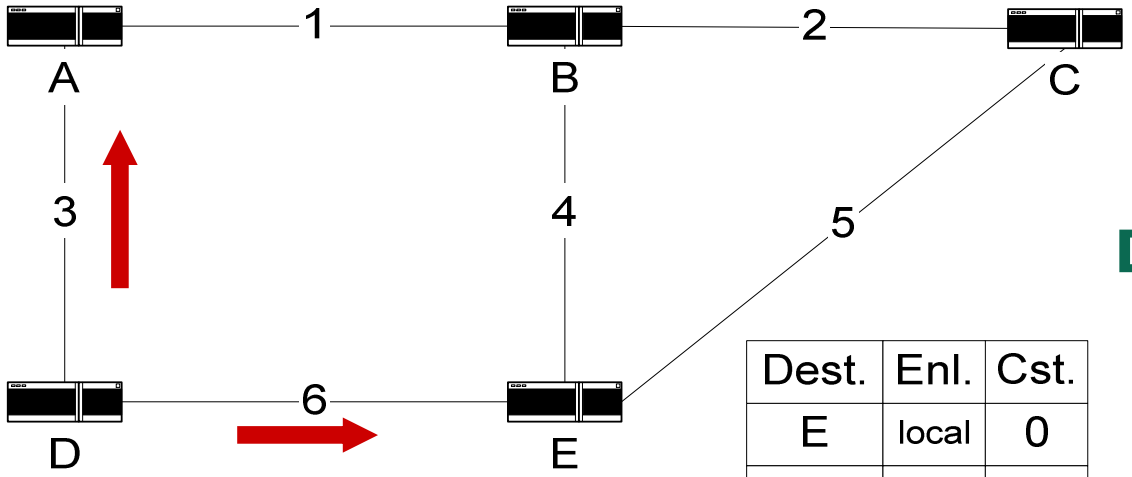
| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| | | |
| | | |

# Exemplo – DV

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | 1 | | | |
| | | | | | |
| | | | | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | 1 | | | |
| | | | | | |
| | | | | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| | | | | | |
| | | | | | |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 | | | |
| | | | | | |
| | | | | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | 2 | | | |
| | | | | | |
| | | | | | |

**D** envia seu vetor

**(D=0, A = 1)**

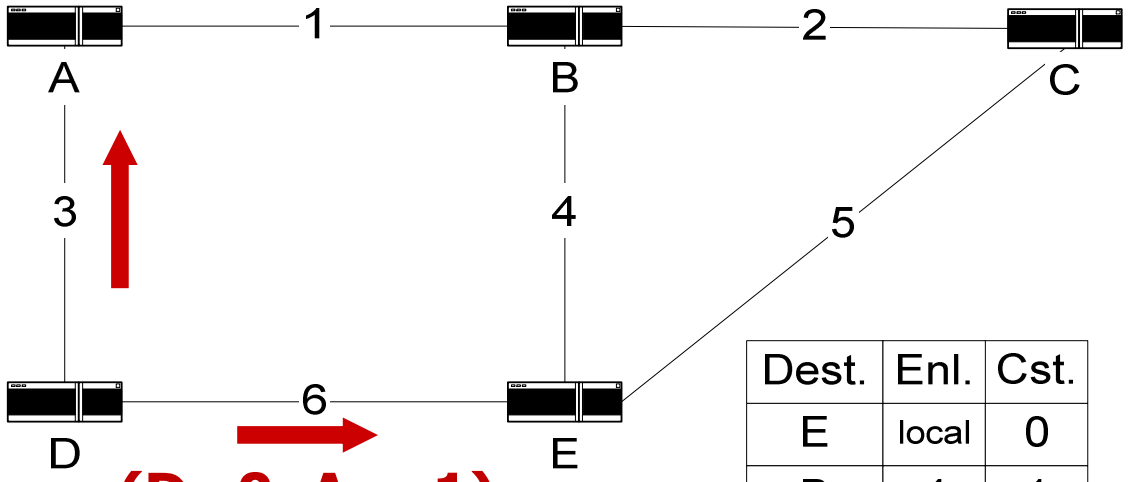
# Exemplo – DV

**A** atualiza sua tabela (**D=1**)

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | 1 |
| D | 3 | 1 | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | 1 |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| | | |
| | | |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| | | |

**E** atualiza sua tabela com **D=1**.
Além disso, também poderia atualizar **A=2** passando por **D**.

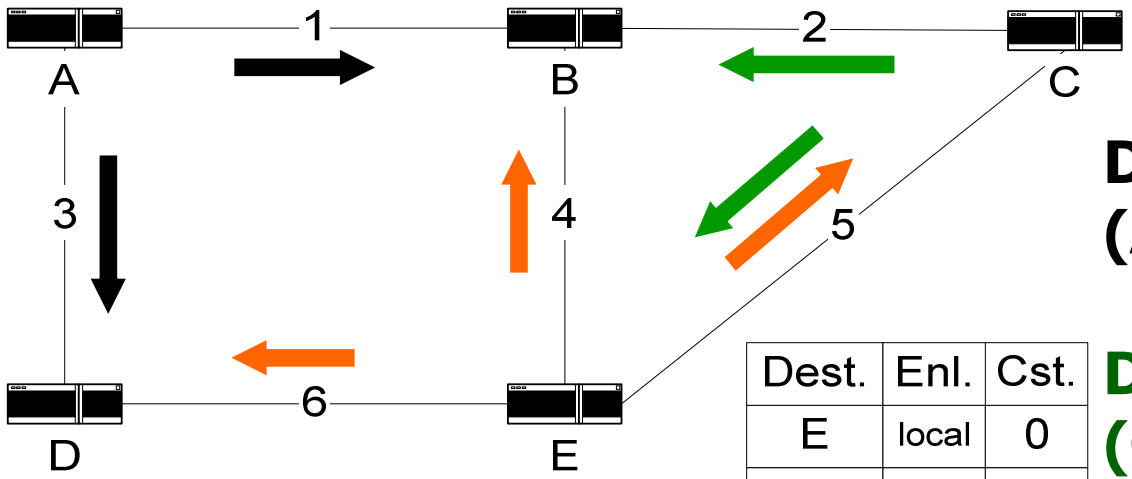
# Exemplo – DV

**A**, **C** e **E** preparam vetores, pois atualizaram suas tabelas

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | 1 |
| D | 3 | 1 |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | 1 |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| | | |
| | | |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| | | |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| | | |

**De A: → (A=0, B=1, D=1)**

**De C: → (C=0, B=1, A=2)**

**De E: → (E=0, B=1, A=2, D=1)**

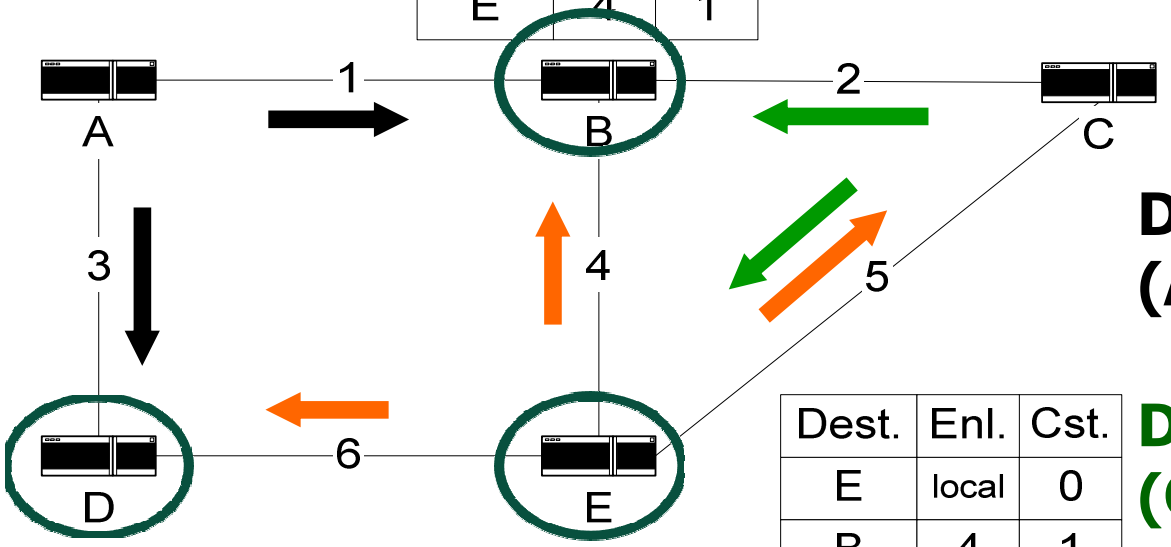
# Exemplo – DV

**B, D** e **E** atualizam suas tabelas

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | 1 |
| D | 3 | 1 |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | 1 |
| D | 1 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| | | |
| | | |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | 2 |
| E | 6 | 1 |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

**De A: → (A=0, B=1, D=1)**

**De C: → (C=0, B=1, A=2)**

**De E: → (E=0, B=1, A=2, D=1)**

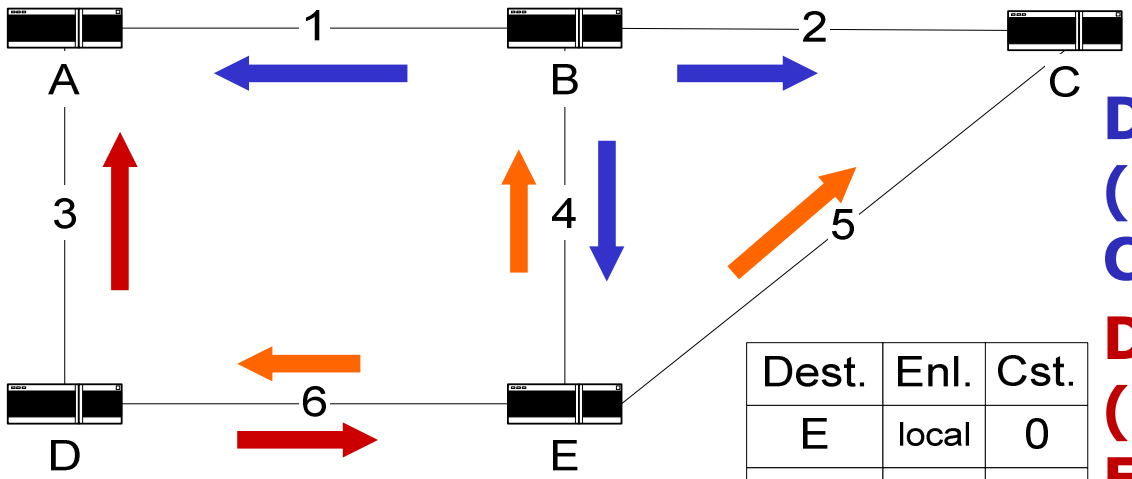
# Exemplo – DV

**B, D** e **E** atualizaram suas tabelas e enviam vetores

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | 1 |
| D | 3 | 1 |
| | | |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | 1 |
| D | 1 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| | | |
| | | |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | 2 |
| E | 6 | 1 |
| | | |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

**De B: → (B=0, A=1, D=2, C=1, E=1)**

**De D: → (D=0, A=1, B=2, E=1)**

**De E: → (E=0, B=1, A=2, D=1, C=1)**

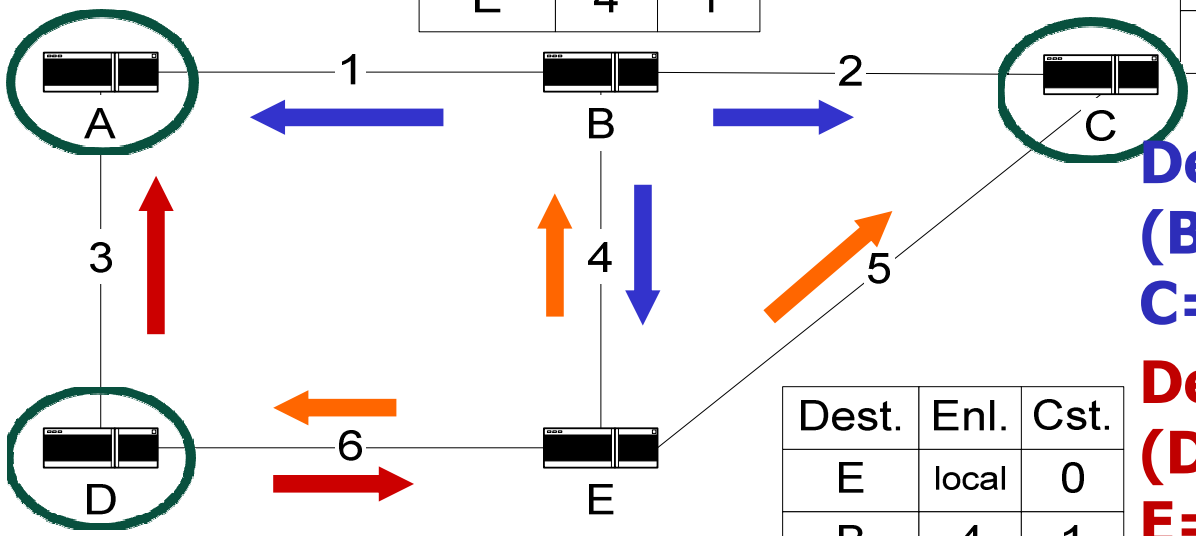
# Exemplo – DV

**A**, **C** e **D** atualizam suas tabelas

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | 1 |
| D | 3 | 1 |
| C | 1 | 2 |
| E | 1 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | 1 |
| D | 1 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| E | 5 | 1 |
| D | 5 | 2 |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

**De B: → (B=0, A=1, D=2, C=1, E=1)**

**De D: → (D=0, A=1, B=2, E=1)**

**De E: → (E=0, B=1, A=2, D=1, C=1)**

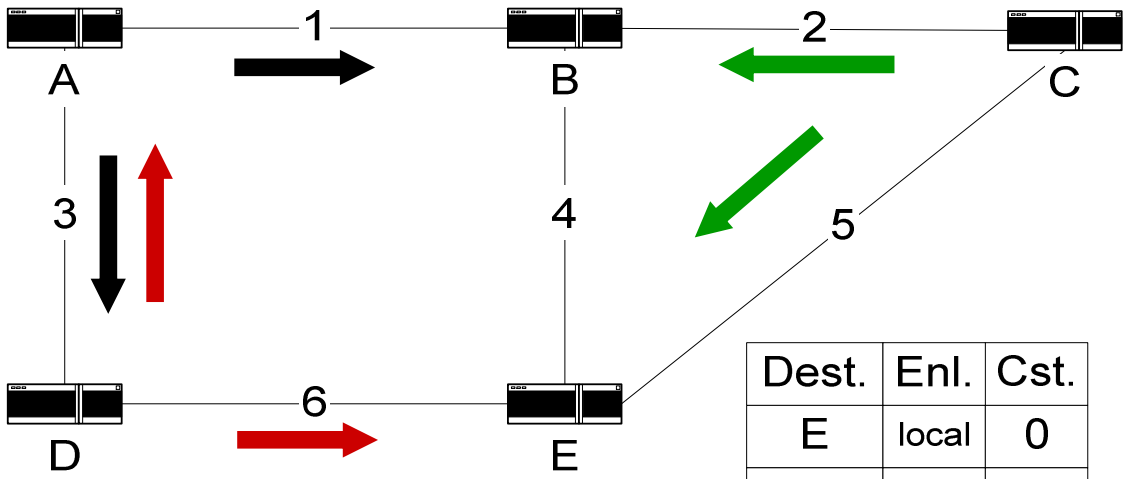
# Exemplo – DV

**A**, **C** e **D** atualizaram suas tabelas e enviam vetores

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | 1 |
| D | 3 | 1 |
| C | 1 | 2 |
| E | 1 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | 1 |
| D | 1 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| E | 5 | 1 |
| D | 5 | 2 |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

No entanto, estas mensagens não modificam mais as tabelas. O algoritmo **convergiu!**

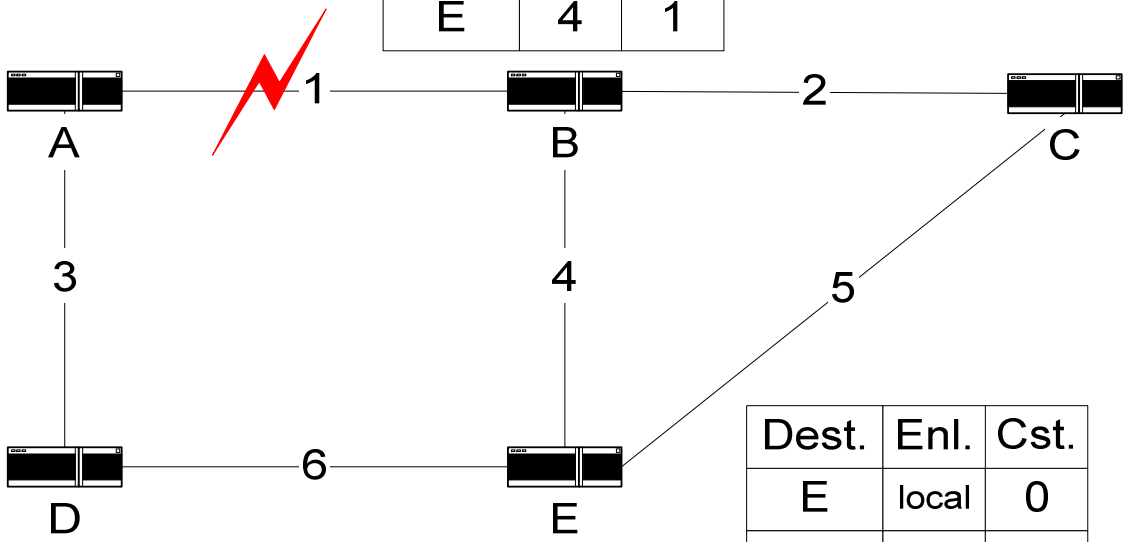
# Falha de um Enlace

Enlace **1** falha. **A** e **B** detectam a falha e atualizam suas tabelas

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | 1 |
| D | 3 | 1 |
| C | 1 | 2 |
| E | 1 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | 1 |
| D | 1 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| E | 5 | 1 |
| D | 5 | 2 |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

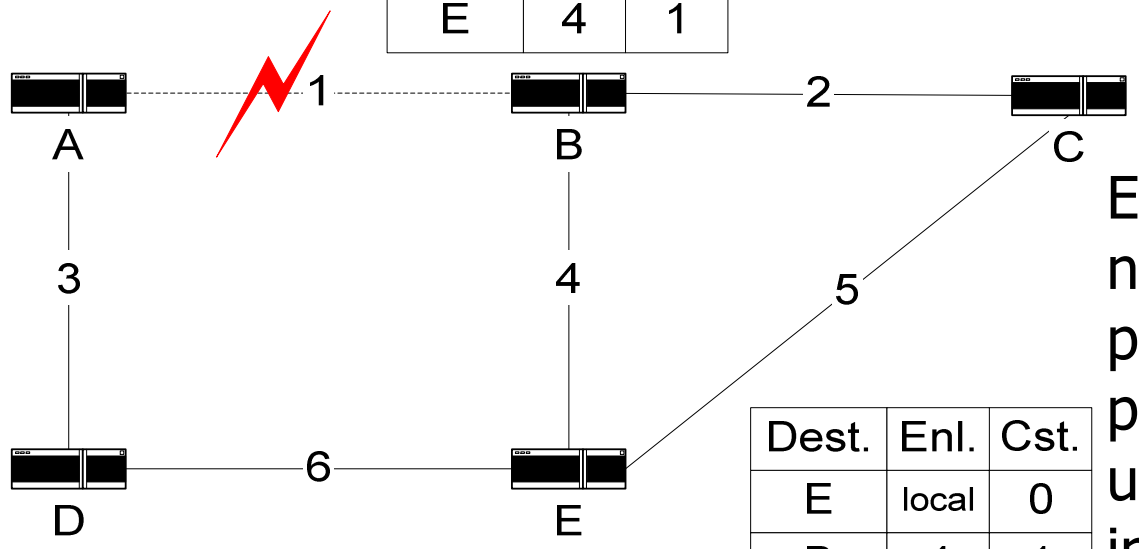
# Falha de um Enlace

**A** e **B** detectam a falha e atualizam suas tabelas

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | inf. |
| D | 3 | 1 |
| C | 1 | inf. |
| E | 1 | inf. |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | inf. |
| D | 1 | inf. |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| E | 5 | 1 |
| D | 5 | 2 |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

Em **A** e **B**, todos os nós alcançáveis pelo enlace **1** passam a estar a uma distância infinita

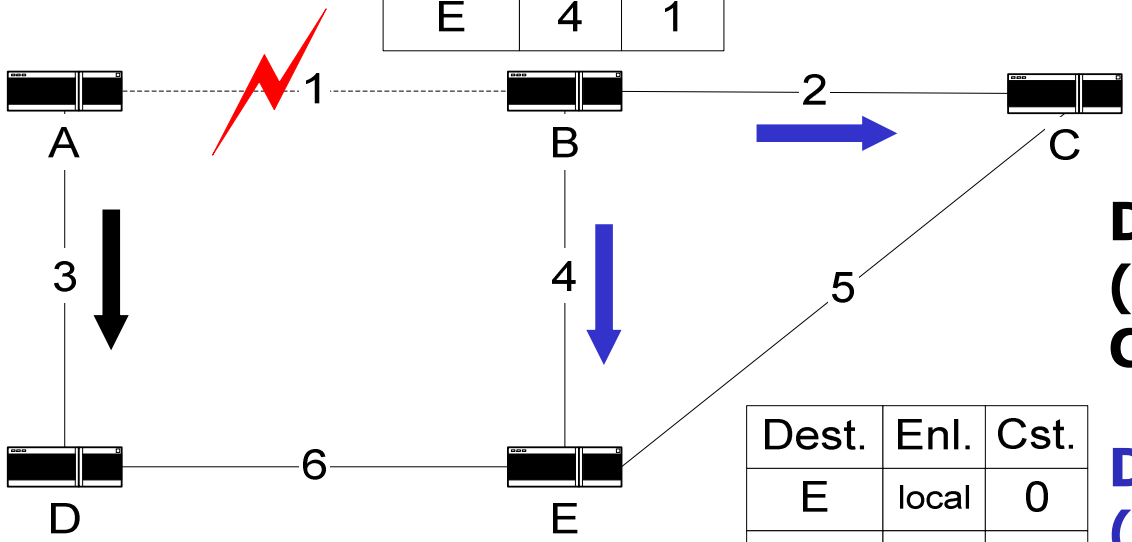
# Falha de um Enlace

**A** e **B** enviam vetores de distância

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | inf. |
| D | 3 | 1 |
| C | 1 | inf. |
| E | 1 | inf. |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | inf. |
| D | 1 | inf. |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| E | 5 | 1 |
| D | 5 | 2 |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

**De A: ⟶**
**(A=0, B=inf, D=1, C=inf, E=inf)**

**De B: ⟶**
**(B=0, A=inf, D=inf, C=1, E=1)**

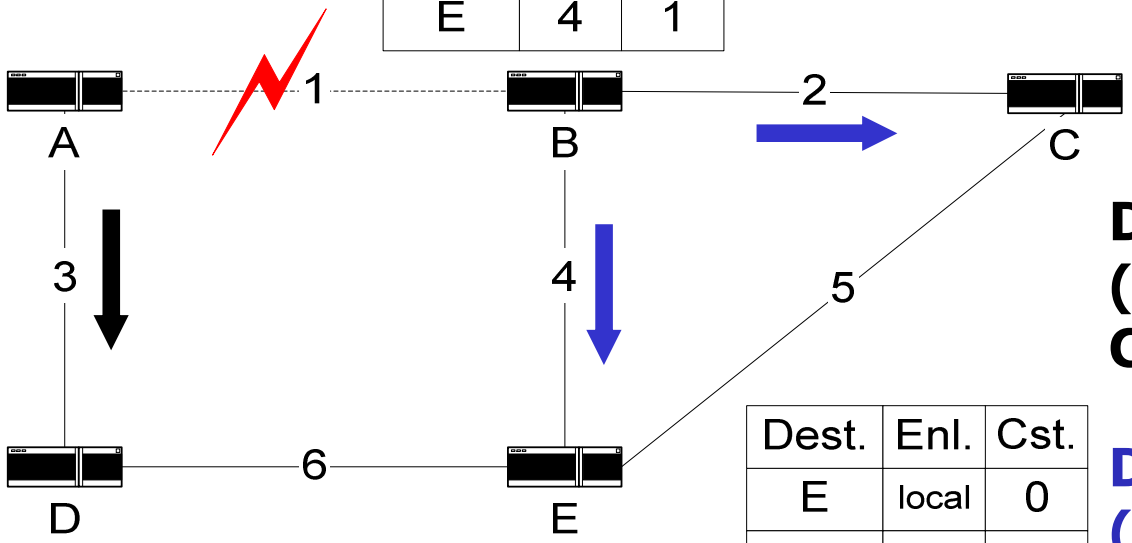
# Falha de um Enlace

**D** atualiza sua tabela. Só a rota p/ **B** é modificada (usa o enlace **3**)

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | inf. |
| D | 3 | 1 |
| C | 1 | inf. |
| E | 1 | inf. |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | inf. |
| D | 1 | inf. |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| E | 5 | 1 |
| D | 5 | 2 |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | inf. |
| E | 6 | 1 |
| C | 6 | 2 |

A partir do vetor de **A:**
**(A=1, B=inf, D=2, C=inf, E=inf)**

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

**De A: →**
**(A=0, B=inf, D=1, C=inf, E=inf)**

**De B: →**
**(B=0, A=inf, D=inf, C=1, E=1)**

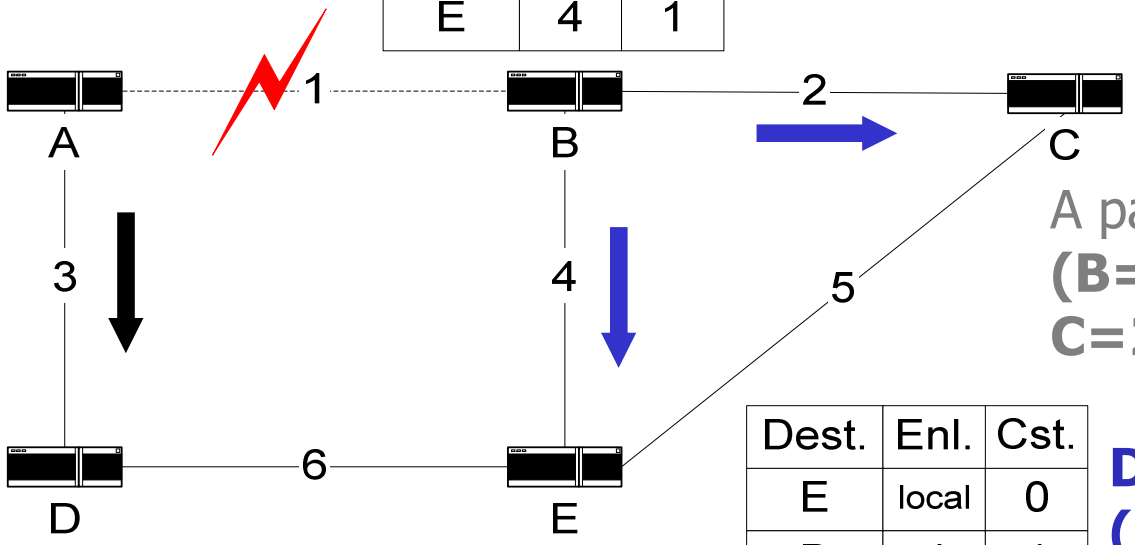
# Falha de um Enlace

**C** e **E** atualizam suas tabelas

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | inf. |
| D | 3 | 1 |
| C | 1 | inf. |
| E | 1 | inf. |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | inf. |
| D | 1 | inf. |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | inf. |
| E | 5 | 1 |
| D | 5 | 2 |




A partir do vetor de **B:** **(B=1, A=inf, D=inf, C=2, E=2)**

| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | inf. |
| E | 6 | 1 |
| C | 6 | 2 |

A partir do vetor de **B:** **(B=1, A=inf, D=inf, C=2, E=2)**

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | inf. |
| D | 6 | 1 |
| C | 5 | 1 |

**De B:** → **(B=0, A=inf, D=inf, C=1, E=1)**

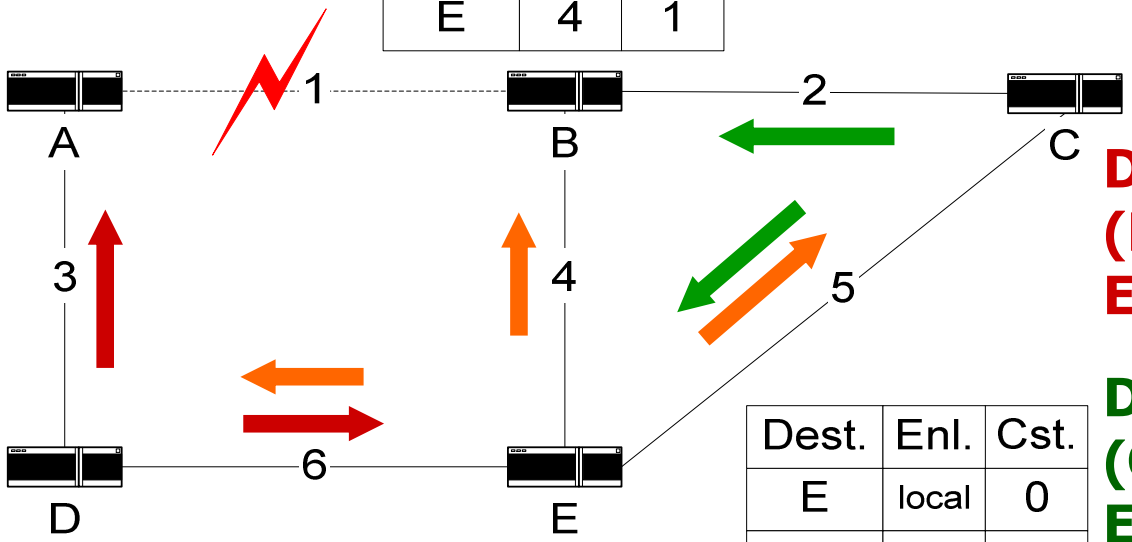
# Falha de um Enlace

**D, C** e **E** enviam vetores de distância

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | inf. |
| D | 3 | 1 |
| C | 1 | inf. |
| E | 1 | inf. |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | inf. |
| D | 1 | inf. |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | inf. |
| E | 5 | 1 |
| D | 5 | 2 |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | inf. |
| E | 6 | 1 |
| C | 6 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 4 | inf. |
| D | 6 | 1 |
| C | 5 | 1 |

**De D: →**
**(D=0, A=1, B=inf, E=1, C=2)**

**De C: →**
**(C=0, B=1, A=inf, E=1, D=2)**

**De E: →**
**(E=0, B=1, A=inf, D=1, C=1)**

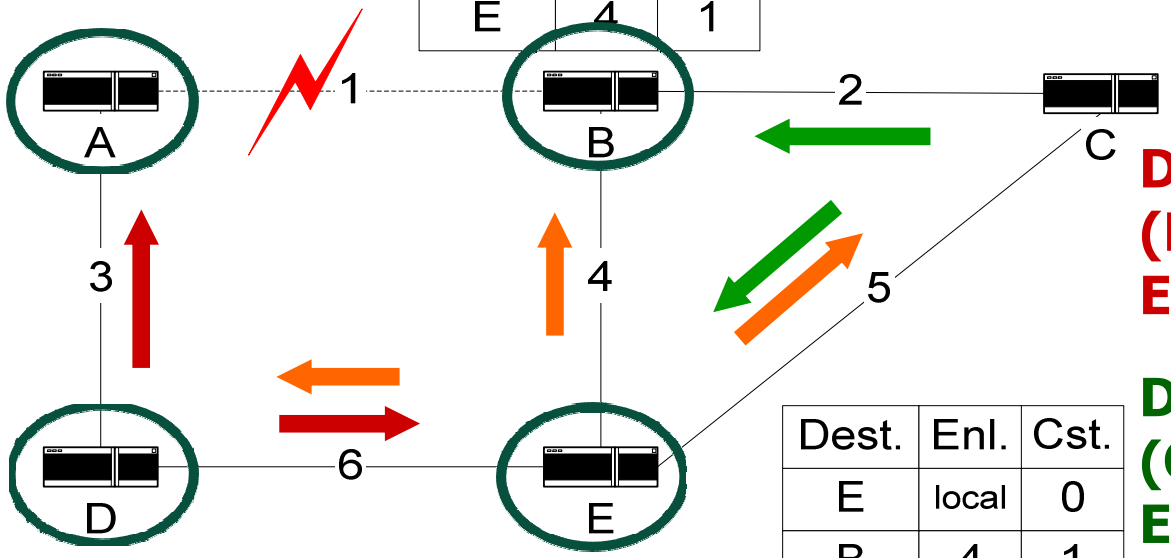
# Falha de um Enlace

**A, B, D** e **E** atualizam suas tabelas

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | inf. |
| D | 3 | 1 |
| C | 3 | 3 |
| E | 3 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | inf. |
| D | 4 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | inf. |
| E | 5 | 1 |
| D | 5 | 2 |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 6 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 6 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

**De D: → (D=0, A=1, B=inf, E=1, C=2)**

**De C: → (C=0, B=1, A=inf, E=1, D=2)**

**De E: → (E=0, B=1, A=inf, D=1, C=1)**

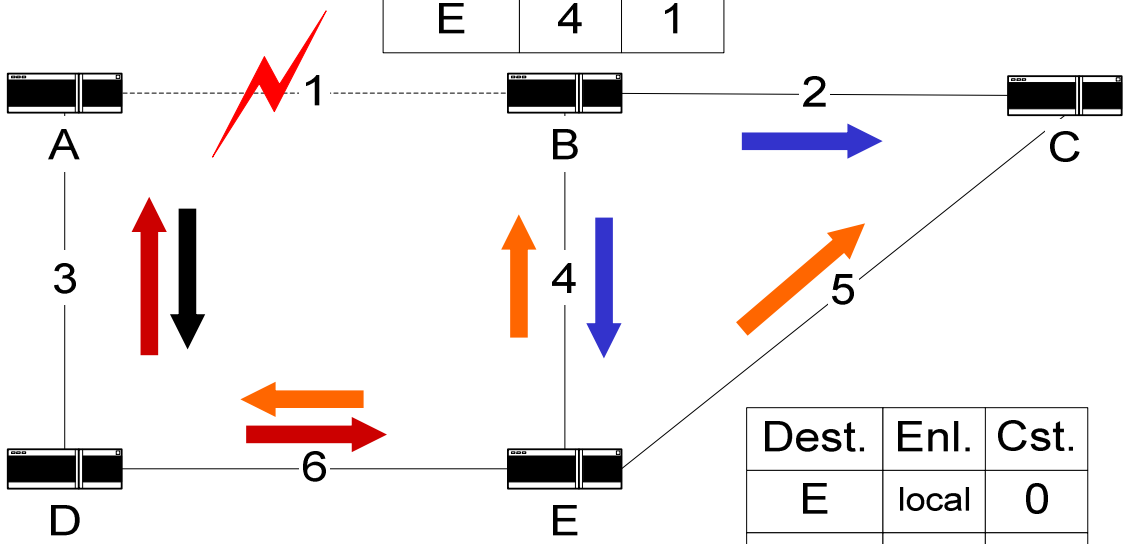
# Falha de um Enlace

**A, B, D** e **E** enviam seus vetores de distância

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 1 | inf. |
| D | 3 | 1 |
| C | 3 | 3 |
| E | 3 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 1 | inf. |
| D | 4 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 2 | inf. |
| E | 5 | 1 |
| D | 5 | 2 |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 6 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 6 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

**De A: →**
**(A=0, B=inf, D=1, C=3, E=2)**

**De B: →**
**(B=0, A=inf, D=2, C=1, E=1)**

**De D: →**
**(D=0, A=1, B=2, E=1, C=2)**

**De E: →**
**(E=0, B=1, A=2, D=1, C=1)**

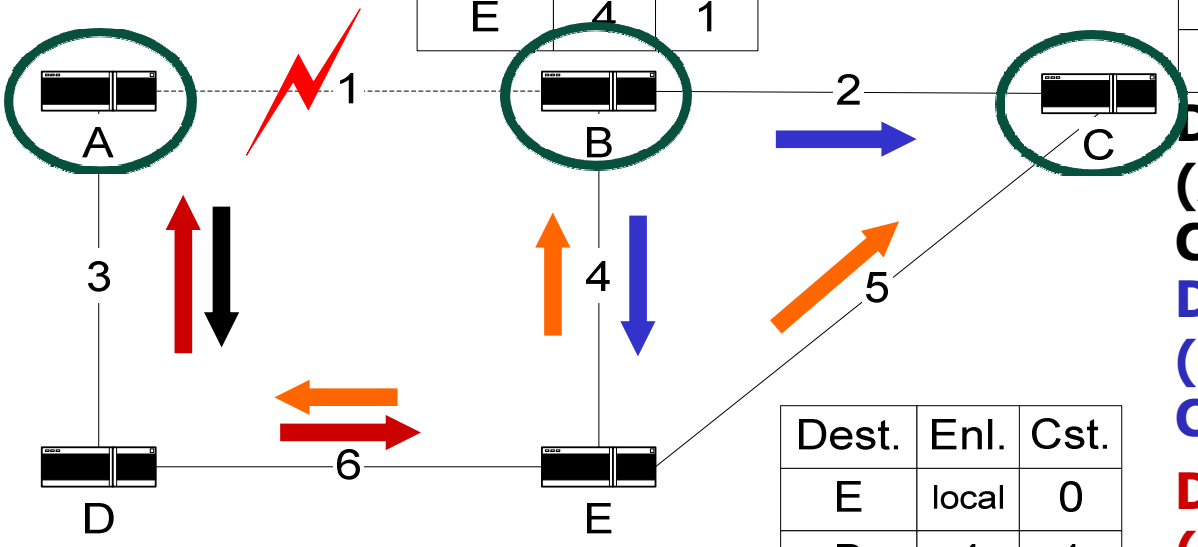
# Falha de um Enlace

**A, B** e **C** atualizam suas tabelas

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 3 | 3 |
| D | 3 | 1 |
| C | 3 | 3 |
| E | 3 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 4 | 3 |
| D | 4 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 5 | 3 |
| E | 5 | 1 |
| D | 5 | 2 |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 6 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 6 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

**De A:** ➡
**(A=0, B=inf, D=1, C=3, E=2)**

**De B:** ➡
**(B=0, A=inf, D=2, C=1, E=1)**

**De D:** ➡
**(D=0, A=1, B=2, E=1, C=2)**

**De E:** ➡
**(E=0, B=1, A=2, D=1, C=1)**

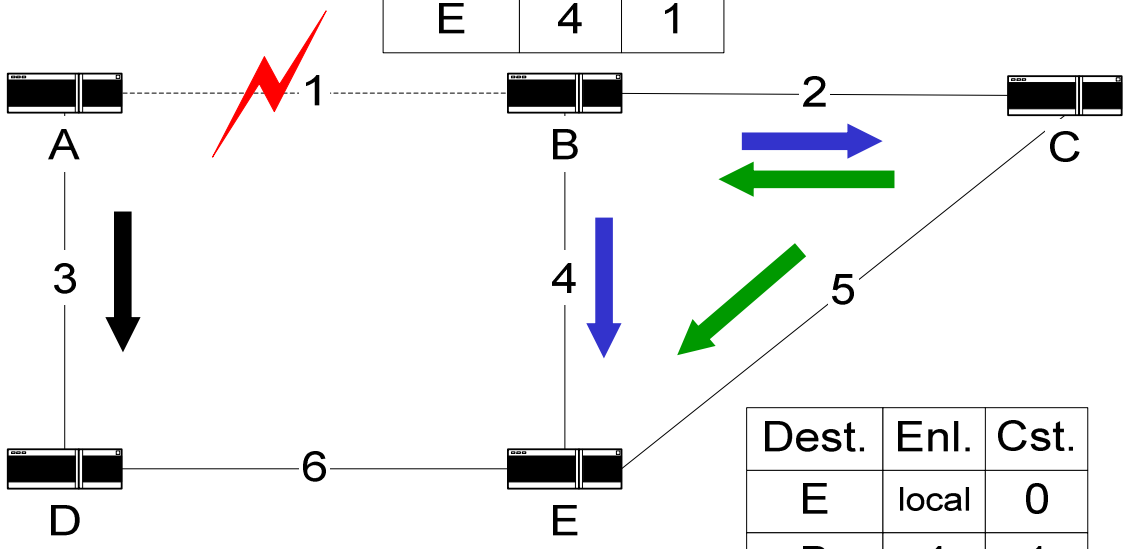
# Falha de um Enlace

**A, B** e **C** enviam seus vetores de distância

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 3 | 3 |
| D | 3 | 1 |
| C | 3 | 3 |
| E | 3 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 4 | 3 |
| D | 4 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 5 | 3 |
| E | 5 | 1 |
| D | 5 | 2 |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 6 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

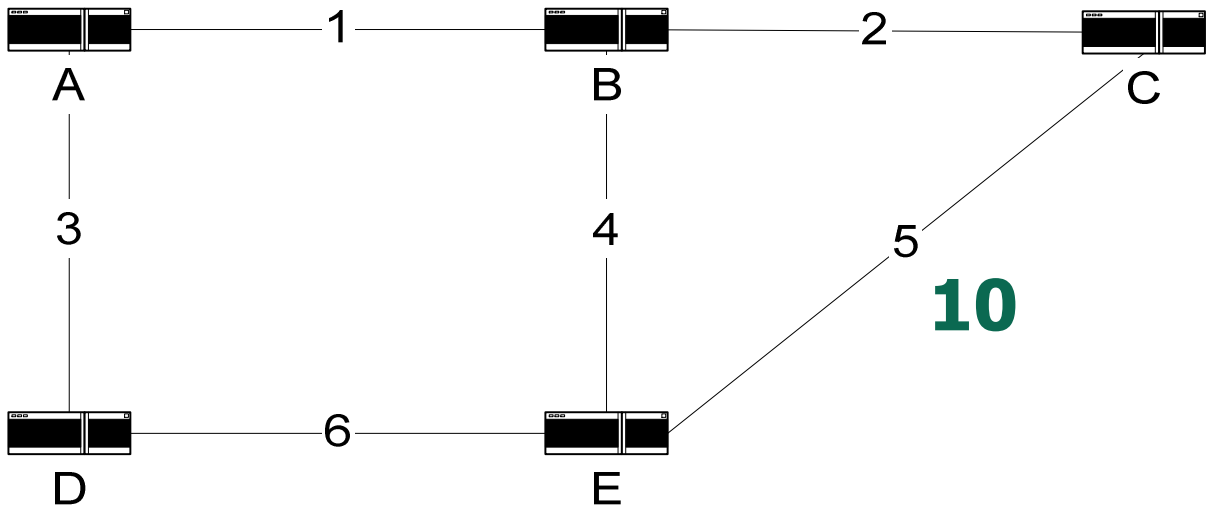
| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 6 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

No entanto, estas mensagens não modificam mais as tabelas. O algoritmo **convergiu!**

# *Bouncing Effect*

- Custo dos enlaces = 1
- Exceto enlace **5**, custo = **10**

- Enlace **2** falha em um dado momento



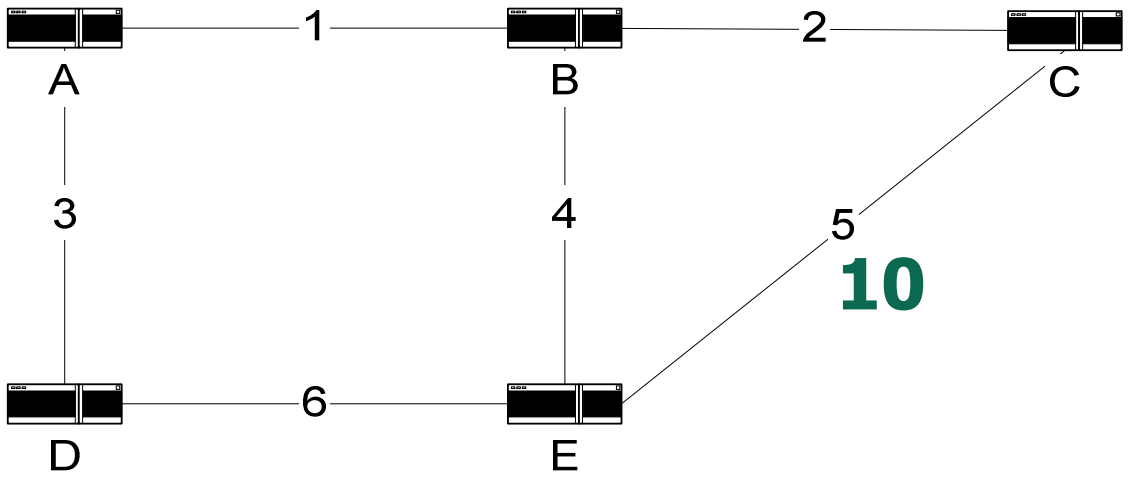
# Bouncing Effect

Rotas para **C** após convergência (enlace **5** possui custo 10)

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 2 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |




| Dest. | Enl. | Cst. |
|---|---|---|
| C | 3 | 3 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 4 | 2 |

# Bouncing Effect

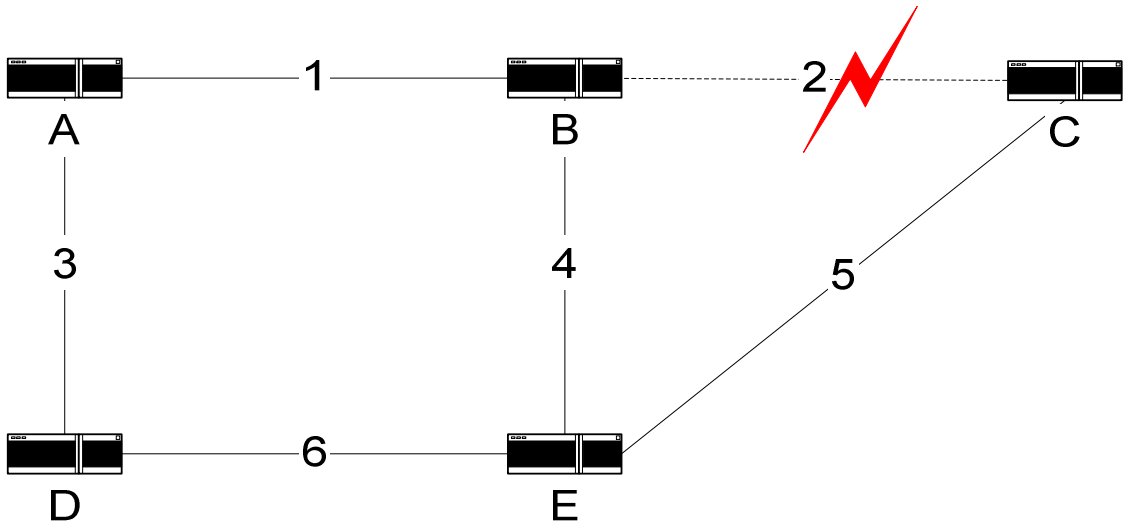
Instituto de computação

Enlace **2** falha

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 2 | inf. |

**B** atualiza imediatamente sua tabela




| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 3 | 3 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 4 | 2 |

# Bouncing Effect

Instituto de computação

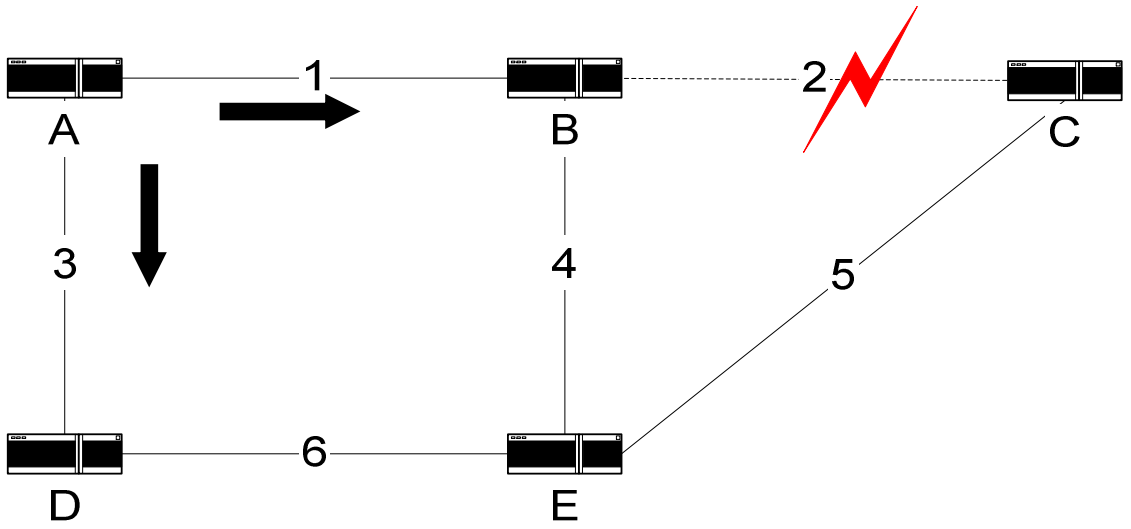
Suponha que **A** envia um vetor de distância

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 3 |

Para **B**, 3 é menor que inf., **B** atualiza sua tabela

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |




**De A: (C=2)**

Para **D**, não há mudanças

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 3 | 3 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 4 | 2 |

# Bouncing Effect

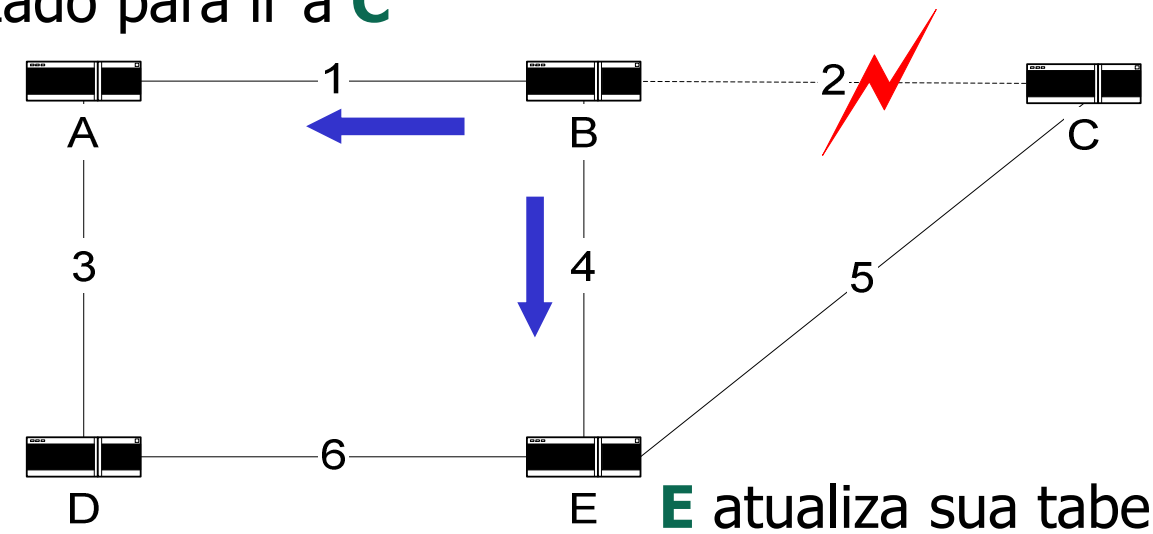
| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 3 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 4 |

Agora, **B** envia um vetor de distância

**A** atualiza sua tabela, pois o DV chegou pelo enlace utilizado para ir a **C**

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |




**De B: (C=3)**

**E** atualiza sua tabela, pois o DV Chegou p/ enlace usado para ir a **C**

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 3 | 3 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 4 | 4 |

# Bouncing Effect

Instituto de computação

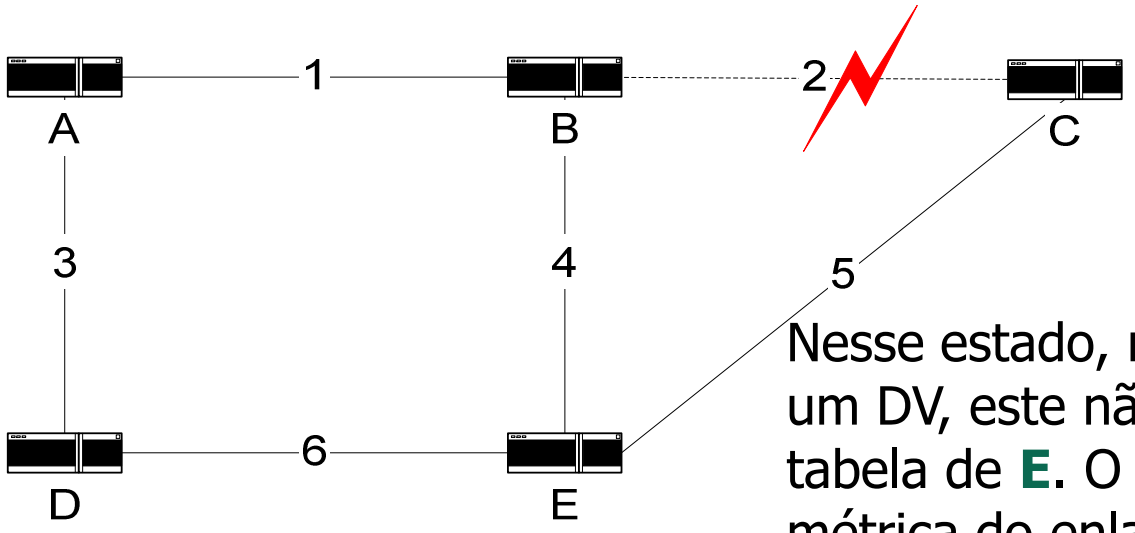
Nesse estado, a rede possui um ***loop*** de roteamento

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 4 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 3 |

**B** aponta para **A**, que aponta para **B**

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |




Nesse estado, mesmo que **C** envie um DV, este não terá efeito na tabela de **E**. O custo anunciado + métrica do enlace **5** (10) é maior que a métrica em **E**.

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 3 | 3 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 4 | 4 |

# Bouncing Effect

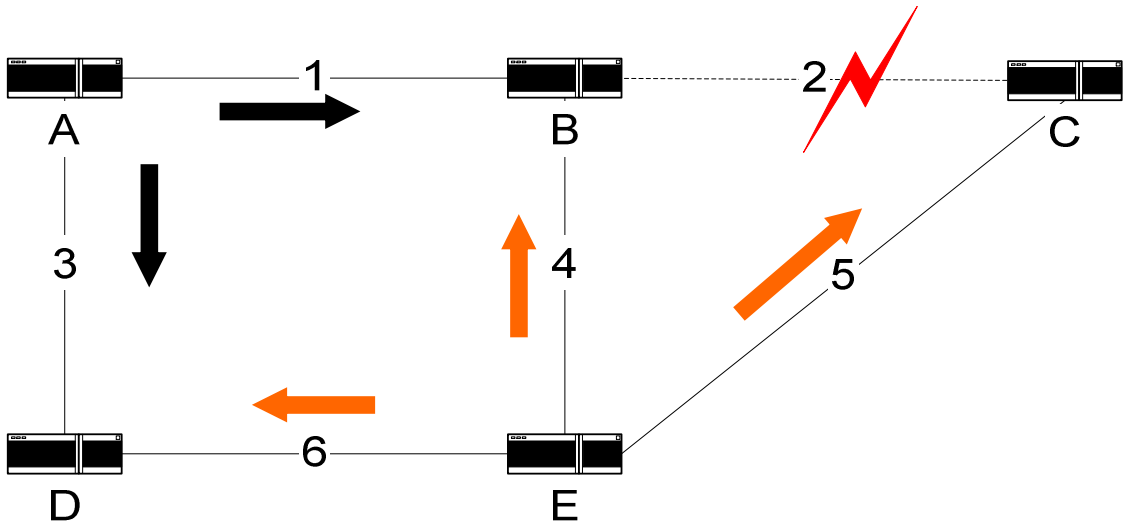
**A** e **E** enviam vetores de distância

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 4 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 5 |

**B** e **D** atualizam suas tabelas

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |




**De A: (C=4)**

**De E: (C=4)**

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 3 | 5 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 4 | 4 |

# Bouncing Effect

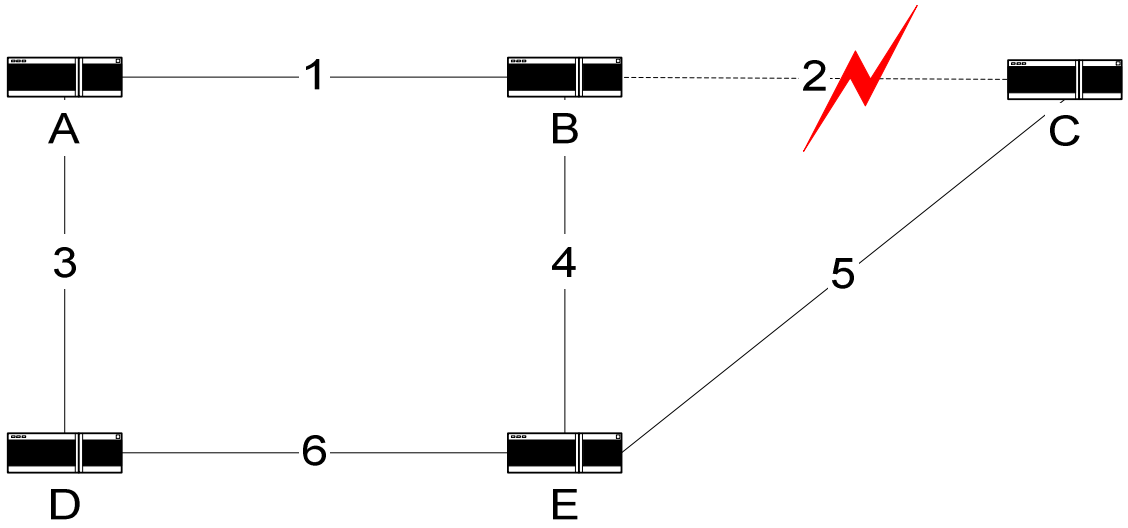
**B** e **D** enviam vetores de distância

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 6 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 5 |

**A** e **E** atualizam suas tabelas

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |




| Dest. | Enl. | Cst. |
|---|---|---|
| C | 3 | 5 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 4 | 6 |

# Bouncing Effect

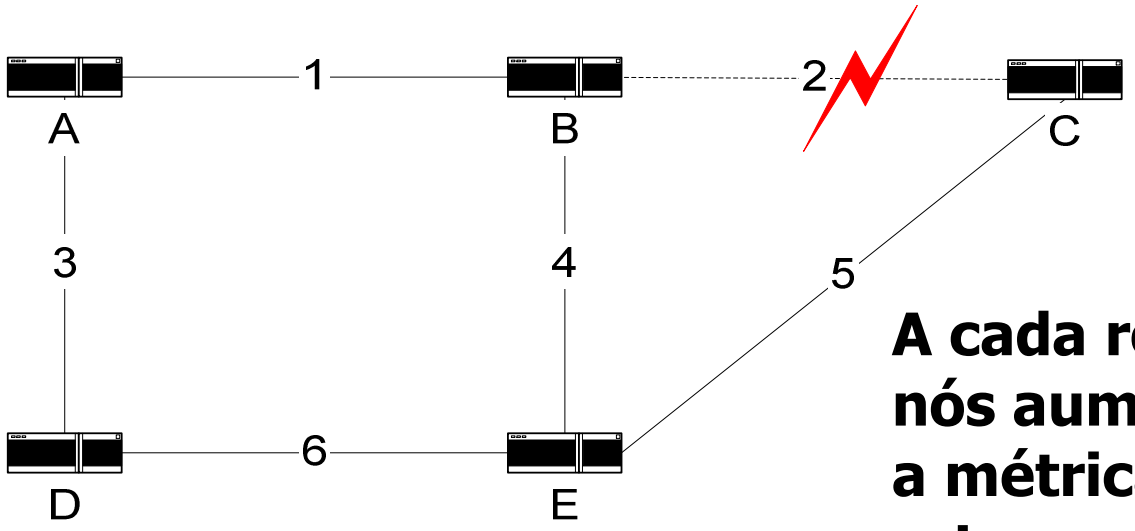
**A** e **E** enviam vetores de distância

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 6 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 7 |

**B** e **D** atualizam suas tabelas.

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |




**A cada rodada, os nós aumentam de 2 a métrica de sua rota para C**

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 3 | 7 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 4 | 6 |

# Bouncing Effect

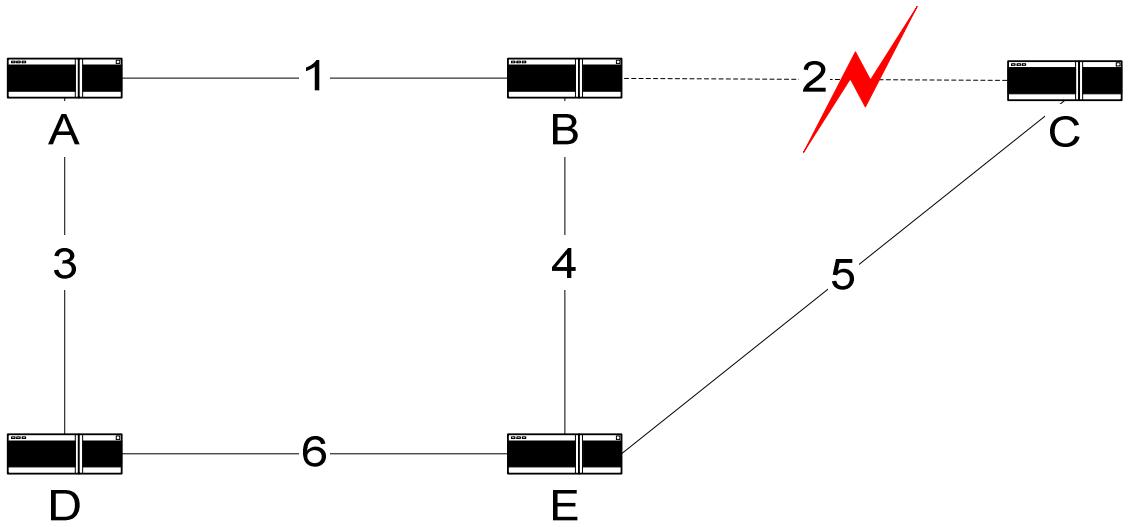
Instituto de computação

Após mais uma rodada...

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 9 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 8 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |




| Dest. | Enl. | Cst. |
|---|---|---|
| C | 3 | 9 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 4 | 8 |

# Bouncing Effect

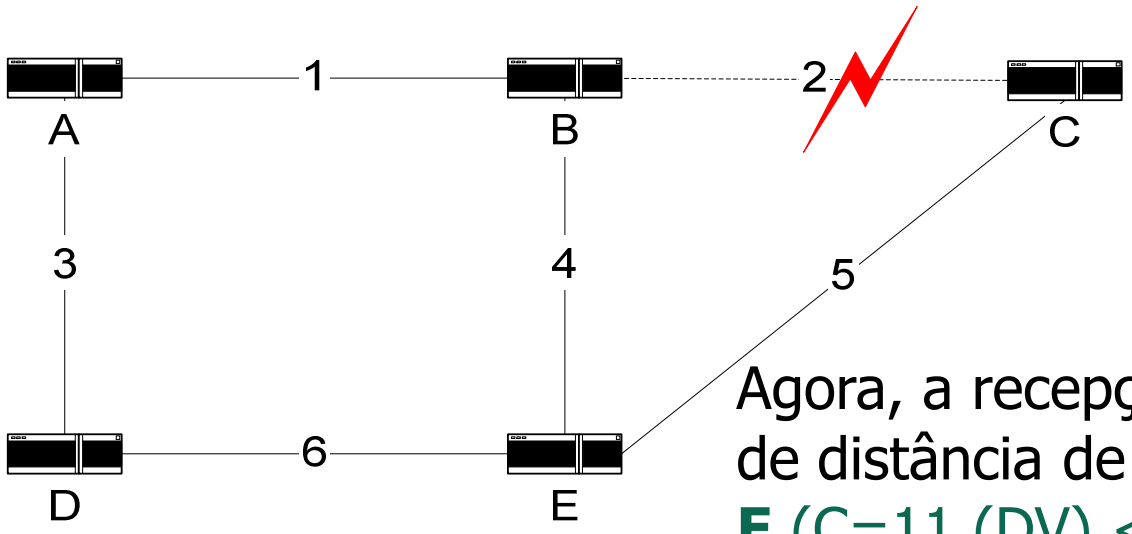
Instituto de computação

Após mais duas rodadas...

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 12 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 11 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |




Agora, a recepção de um vetor de distância de **C** tem efeito em **E** (C=11 (DV) < 12 (tabela))

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 3 | 11 |

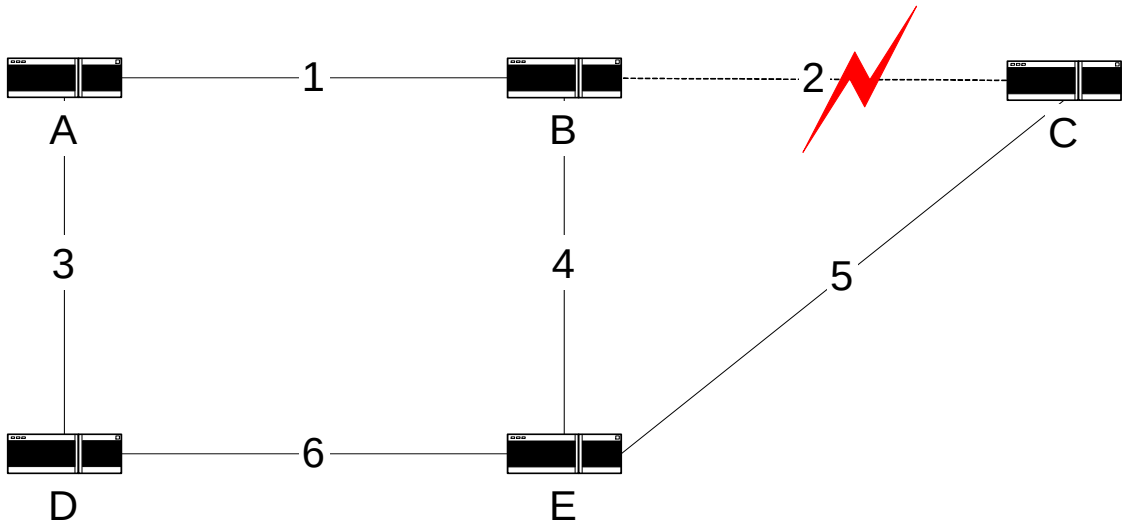
| Dest. | Enl. | Cst. |
|---|---|---|
| C | 4 | 12 |

# Bouncing Effect

**E** atualiza sua tabela

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 11 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 12 |




| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 3 | 11 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 5 | 10 |

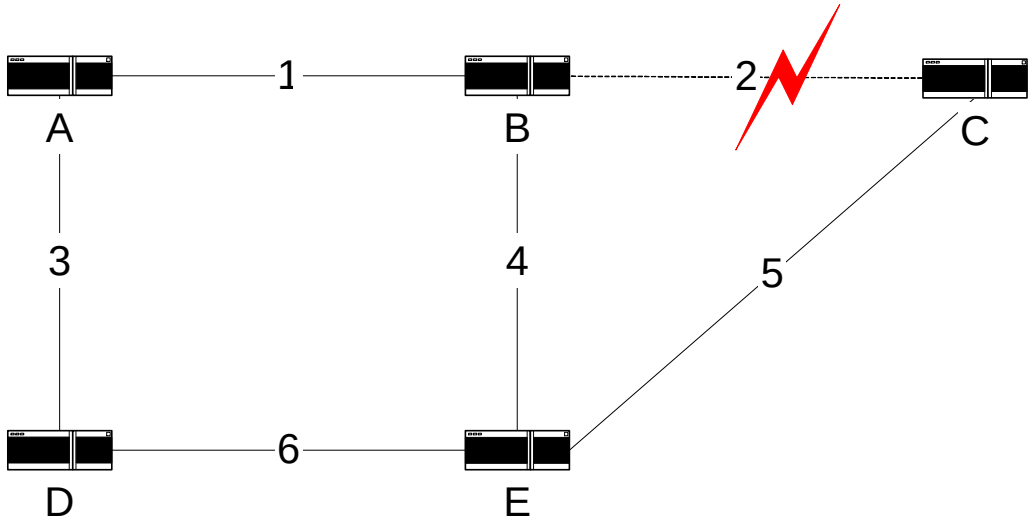
# Bouncing Effect

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 4 | 11 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 1 | 12 |

Após mais alguns passos,
o algoritmo converge

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |




| Dest. | Enl. | Cst. |
|---|---|---|
| C | 6 | 11 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | 5 | 10 |

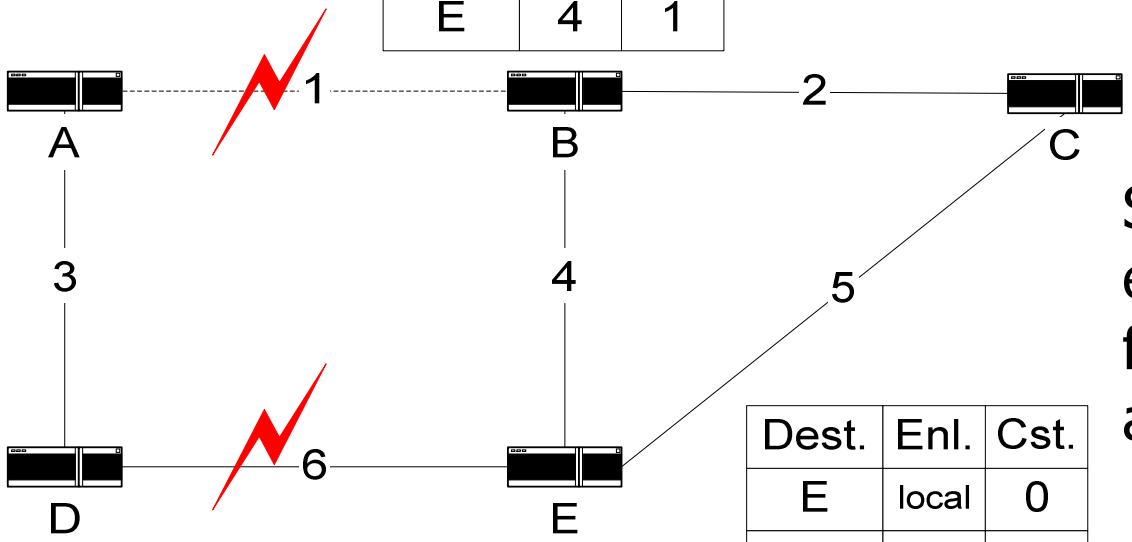
# Contagem até o Infinito

Suponha que o enlace **1** falhou e o algoritmo convergiu

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 3 | 3 |
| D | 3 | 1 |
| C | 3 | 3 |
| E | 3 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 4 | 3 |
| D | 4 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 5 | 3 |
| E | 5 | 1 |
| D | 5 | 2 |




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 6 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

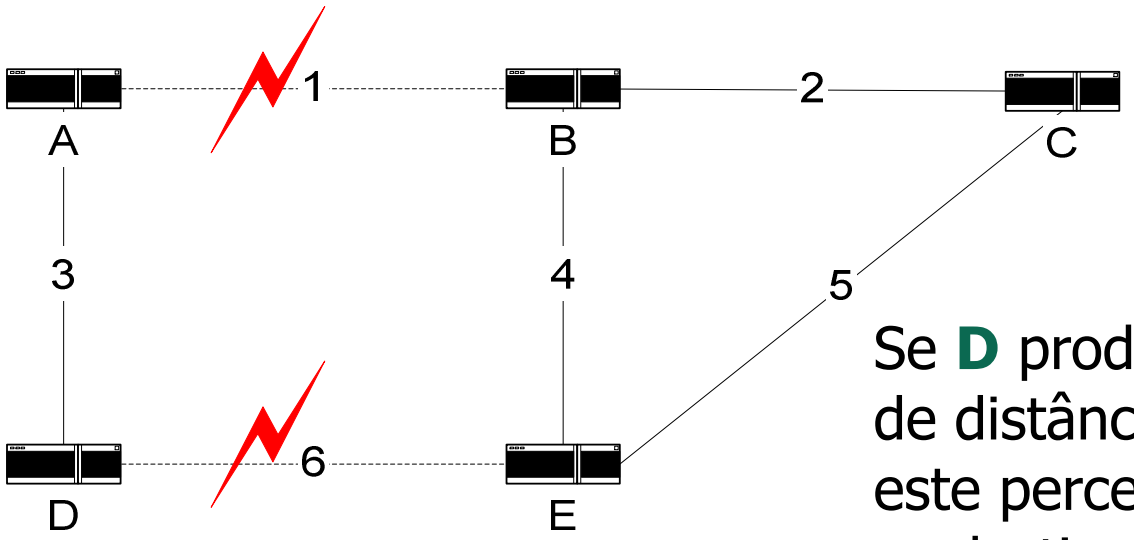
| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 6 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

Suponha que o enlace **6** também falha, separando a rede em duas

# Contagem até o Infinito

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 3 | 3 |
| D | 3 | 1 |
| C | 3 | 3 |
| E | 3 | 2 |

**D** percebe a queda do enlace e atualiza sua tabela de acordo.



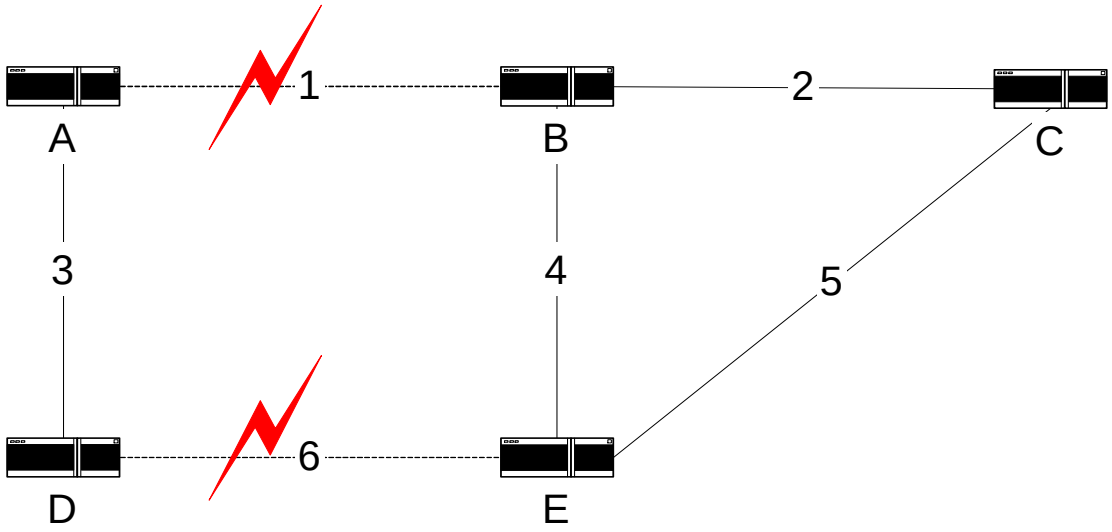

| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 6 | inf. |
| E | 6 | inf. |
| C | 6 | inf. |

Se **D** produzir um vetor de distância antes de **A**, este percebe que todos os destinos exceto **D** estão inalcançáveis.
O algoritmo convergiu.

# Contagem até o Infinito

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 3 | 3 |
| D | 3 | 1 |
| C | 3 | 3 |
| E | 3 | 2 |

No entanto, se **A** enviar seu vetor de distância primeiro, **D** atualizará sua tabela.



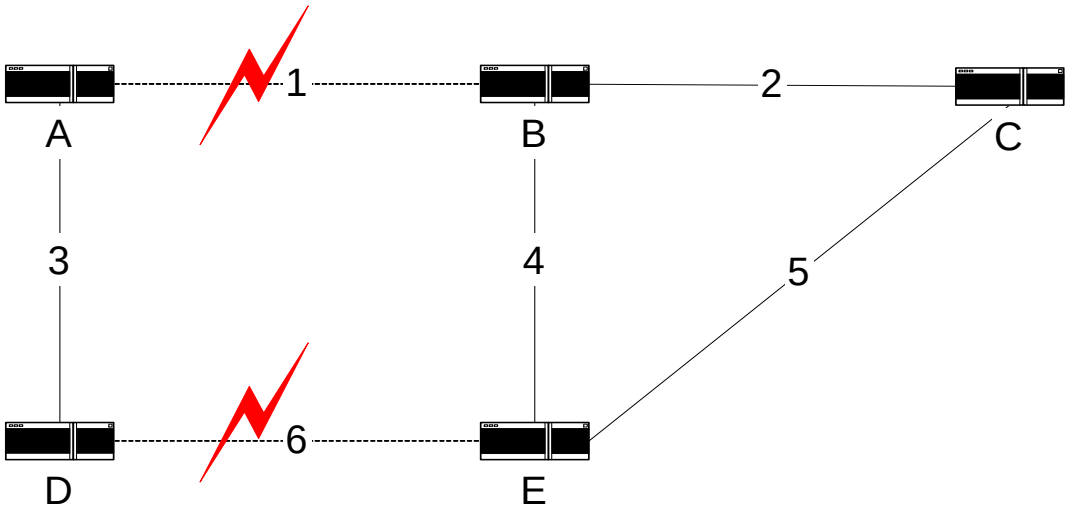

| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | 4 |
| E | 3 | 3 |
| C | 3 | 4 |

# Contagem até o Infinito

**D** enviará um vetor de distância.
**A** atualizará sua tabela.

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 3 | 5 |
| D | 3 | 1 |
| C | 3 | 5 |
| E | 3 | 4 |

Formou-se um *loop* de roteamento entre **A** e **D**.

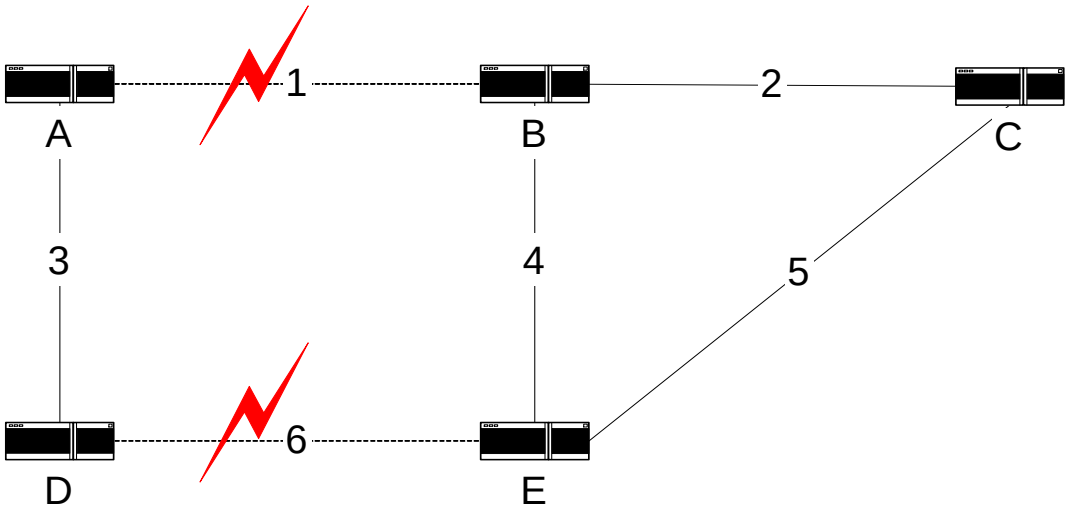



| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | 4 |
| E | 3 | 3 |
| C | 3 | 4 |

# Contagem até o Infinito

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 3 | 7 |
| D | 3 | 1 |
| C | 3 | 7 |
| E | 3 | 6 |

O processo se repete, como no *bouncing effect*. No entanto, a contagem continua até o infinito, uma vez que **B**, **C** e **E** estão isolados de **A** e **D**.




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | 6 |
| E | 3 | 5 |
| C | 3 | 6 |

# Melhorias no Algoritmo BF

Instituto de computação

- *Bouncing effect* e contagem até o infinito
  - Aumento do tempo de convergência

- Melhorias no algoritmo
  - *Split horizon*
  - *Triggered updates*

# *Split Horizon*

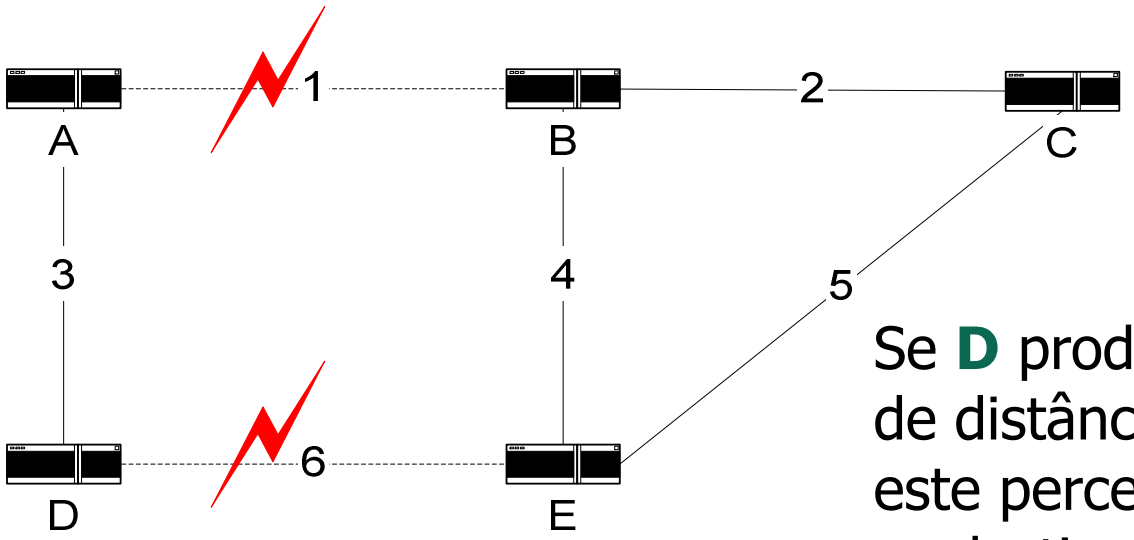
Instituto de computação

- Se A utiliza o nó B para chegar a X, não faz sentido B utilizar A para chegar a X
- Para evitá-lo, A não deve anunciar a B uma rota para X
- Cada nó deve enviar vetores distância diferentes, de acordo com o enlace de saída
  - Rotas que utilizam o enlace E como saída não são anunciadas no vetor distância enviado sobre E

# Contagem até o Infinito

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 3 | 3 |
| D | 3 | 1 |
| C | 3 | 3 |
| E | 3 | 2 |

**D** percebe a queda do enlace e atualiza sua tabela de acordo.



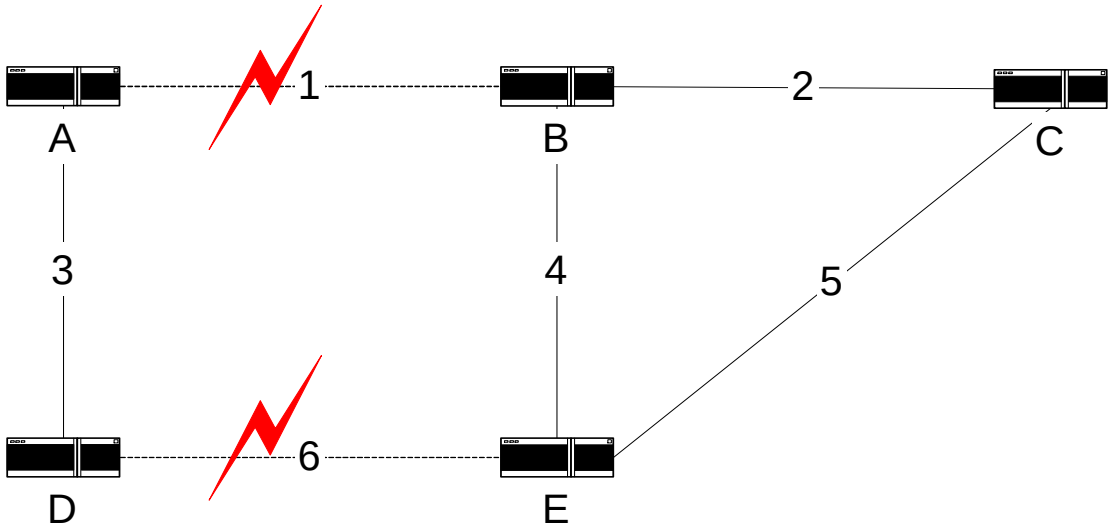

| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 6 | inf. |
| E | 6 | inf. |
| C | 6 | inf. |

Se **D** produzir um vetor de distância antes de **A**, este percebe que todos os destinos exceto **D** estão inalcançáveis.
O algoritmo convergiu.

# Contagem até o Infinito

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 3 | 3 |
| D | 3 | 1 |
| C | 3 | 3 |
| E | 3 | 2 |

No entanto, se **A** enviar seu vetor de distância primeiro, **D** atualizará sua tabela.




| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 3 | 4 |
| E | 3 | 3 |
| C | 3 | 4 |

**Com o *split horizon* isso não acontece!**

A envia DV para D somente com A=0

# *Split Horizon*

Instituto de computação

- Versão simples
  - Nós omitem do vetor de distância *destinos* alcançados através do enlace no qual o vetor é enviado
- *Split horizon with poisonous reverse*
  - Nós incluem no vetor de distância destinos alcançados através do enlace no qual o vetor é enviado, mas com distância ***infinita***
  - O mecanismo evita *loops* com dois saltos
  - Mas não evita *loops* em certos cenários

# Contagem até o Infinito (2)

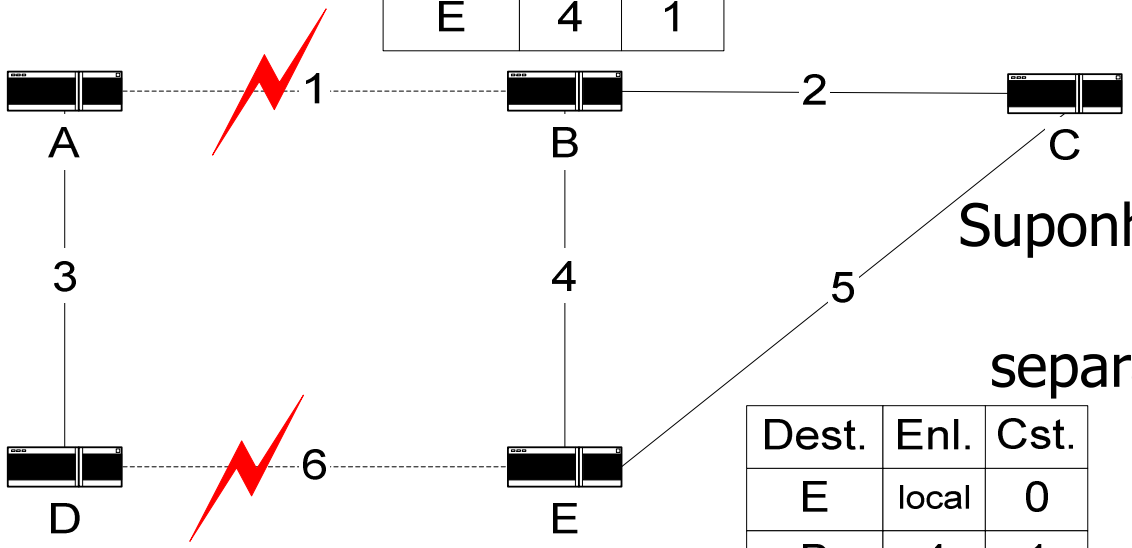
Instituto de computação

Suponha que o enlace **1** falhou e o algoritmo convergiu

| Dest. | Enl. | Cst. |
|---|---|---|
| A | local | 0 |
| B | 3 | 3 |
| D | 3 | 1 |
| C | 3 | 3 |
| E | 3 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| B | local | 0 |
| A | 4 | 3 |
| D | 4 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

| Dest. | Enl. | Cst. |
|---|---|---|
| C | local | 0 |
| B | 2 | 1 |
| A | 5 | 3 |
| E | 5 | 1 |
| D | 5 | 2 |



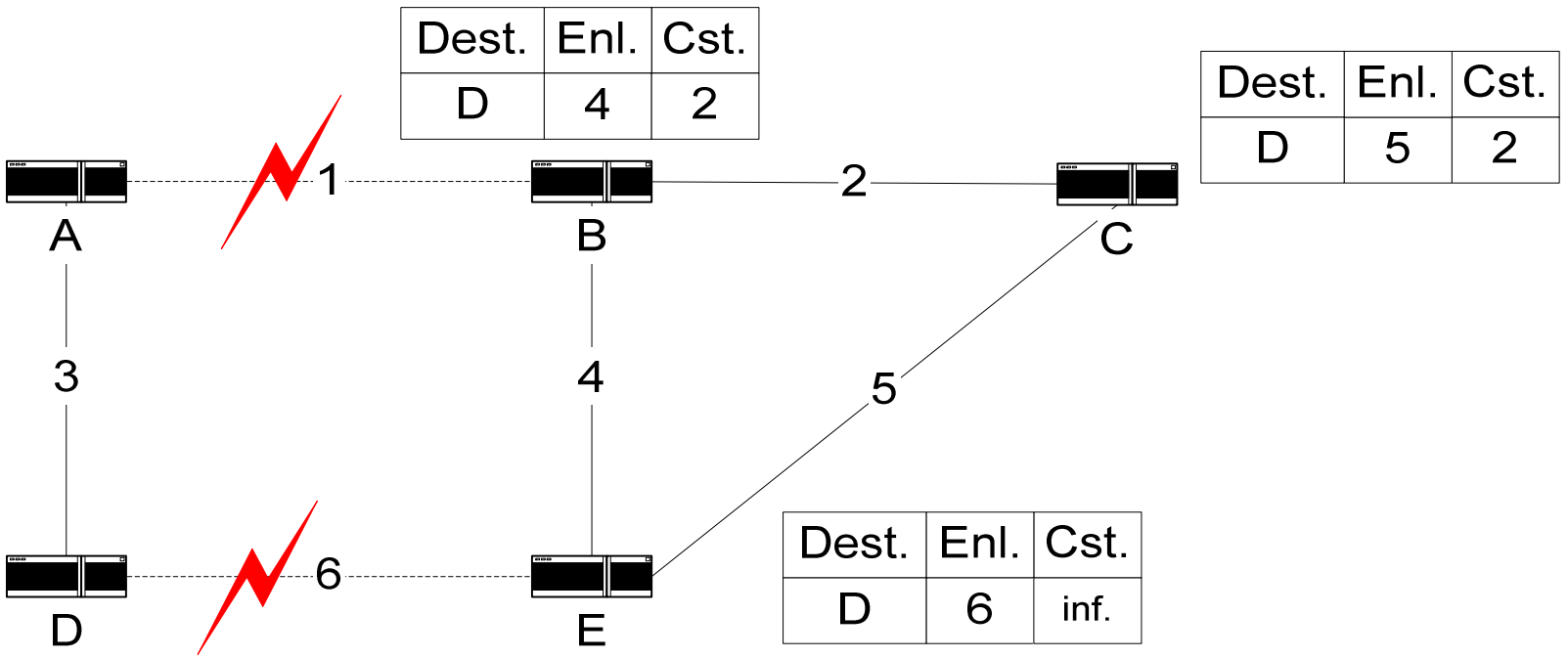

Suponha que o enlace **6** também falha, separando a rede em 2.

| Dest. | Enl. | Cst. |
|---|---|---|
| D | local | 0 |
| A | 3 | 1 |
| B | 6 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

| Dest. | Enl. | Cst. |
|---|---|---|
| E | local | 0 |
| B | 4 | 1 |
| A | 6 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

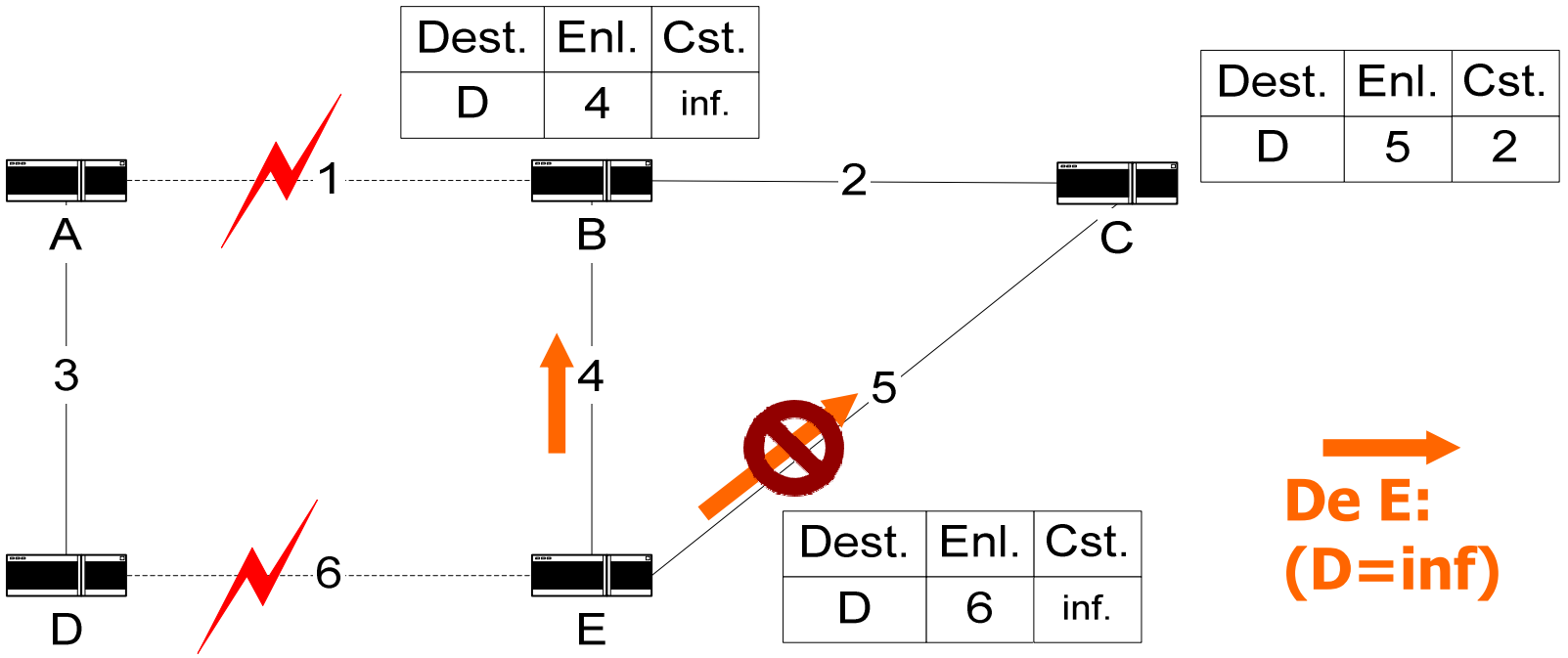
# Contagem até o Infinito (2)

Logo após a falha do enlace **6**, **E** atualiza sua tabela.



# Contagem até o Infinito (2)

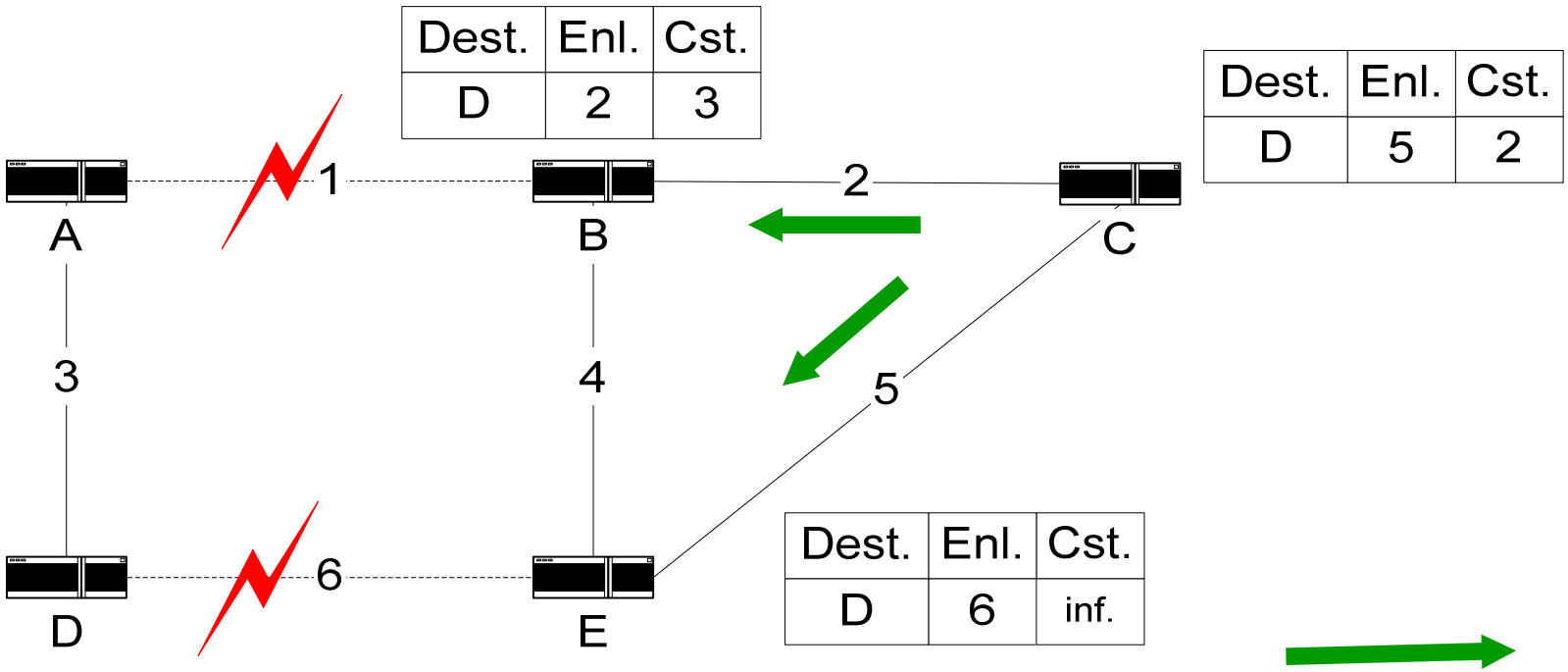
Suponha que **E** envia um vetor de distância, que chega a **B** mas não é recebido por **C** devido a um erro de transmissão.




Apenas **B** atualiza sua tabela.

# Contagem até o Infinito (2)

Agora, **C** envia seus vetores de distância, utilizando *poisonous reverse*.



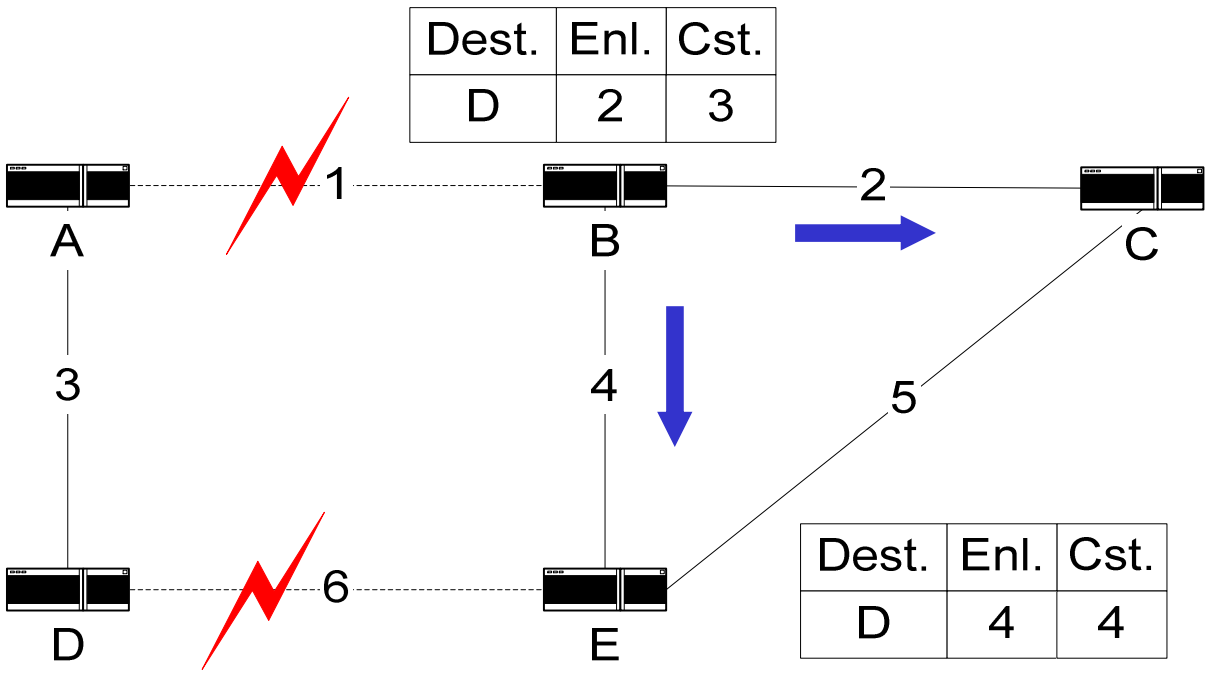

**B** atualiza sua tabela.

**De C:**
**(D=2) no enlace 2**
**(D=inf) no enlace 5**

# Contagem até o Infinito (2)

Instituto de computação

Agora, **B** envia seus vetores de distância. O destino **D** é anunciado no enlace **2** usando o *split horizon with poisonous reverse*.




**De B:** →
**(D=3) no enlace 4**
**(D=inf) no enlace 2**

**E** atualiza sua tabela.
Um *loop* de três saltos se formou (**B → C → E → B**).
A contagem para infinito ocorre entre os três nós.

# *Triggered Updates*

Instituto de computação

- Problema
  - Mudança na rede ocorre **logo depois** da emissão de um DV
  - Roteador deve **esperar** o momento de envio do próximo DV para informar a mudança da rede aos seus vizinhos

# *Triggered Updates*

Instituto de computação

- Envio do vetor de distância logo após a detecção de uma mudança na rede
- Acelera a convergência da rede
  - Alguns dos problemas de convergência são causados por roteadores que re-enviam seu estado logo antes da mudança da rede ser comunicada
- No entanto, problemas ainda podem ocorrer
  - Vetores de distância podem ser perdidos
  - A convergência passa pela contagem até o infinito

# Temporização das Rotas

Instituto de computação

- Entradas nas tabelas de roteamento são voláteis
  - Entradas são associadas a temporizadores
  - Mensagens confirmando a rota reiniciam os temporizadores
  - Se a entrada não é atualizada
    - Conclui-se que um roteador vizinho falhou

# Temporização das Rotas

Instituto de computação

- O tempo de estouro do temporizador deve ser maior que o período de envio das mensagens
  - Ou a perda de um único pacote levaria a marcar um roteador como "morto" desnecessariamente
- O período de envio não deve ser curto demais...
  - Excesso de tráfego de controle
- Nem muito longo
  - Resposta lenta às mudanças da rede

# RIP (*Route Information Protocol*)

Instituto de computação

- Apareceu como componente do UNIX BSD
  - Implementado dentro do **`routed`** (*route management daemon*)
- RIP versão 1, RFC 1058 (1988)
  - Sugere *split horizon* e *triggered updates*, ausente do programa original
- O RIP é um IGP (*Internal Gateway Protocol*)
  - Projetado para troca de informação dentro de um sistema autônomo (AS – *Autonomous System*), ou para redes de **tamanho limitado**

# Endereços no RIPv1

Instituto de computação

- Tabelas RIP
  - Endereços IP de 32 bits
  - Podem representar uma estação, rede, ou sub-rede
    - Porém não há indicação de tipo de endereço nas mensagens
      - Não envia a máscara
- Classificação do endereço
  - Separação rede + sub-rede/estação a partir da classe (A, B ou C)
    - Se sub-rede/estação = 0, endereço de rede
    - Senão, sub-rede ou estação
      - Discrimina-se entre os dois usando a máscara de sub-rede

# Endereços e Rotas no RIPv1

Instituto de computação

- Assume que as máscaras não estejam disponíveis fora da **rede**
  - Portanto, as entradas de **sub-rede** não devem ser propagadas para fora da **rede** à qual elas pertencem
  - As entradas de sub-rede devem ser resumidas em uma entrada de rede correspondente
- O suporte a rotas para estações é opcional
  - Diminuição das tabelas

- O endereço 0.0.0.0 representa uma rota *default*
  - Rota para redes fora deste sistema autônomo (AS)

# Características Básicas do RIPv1

Instituto de computação

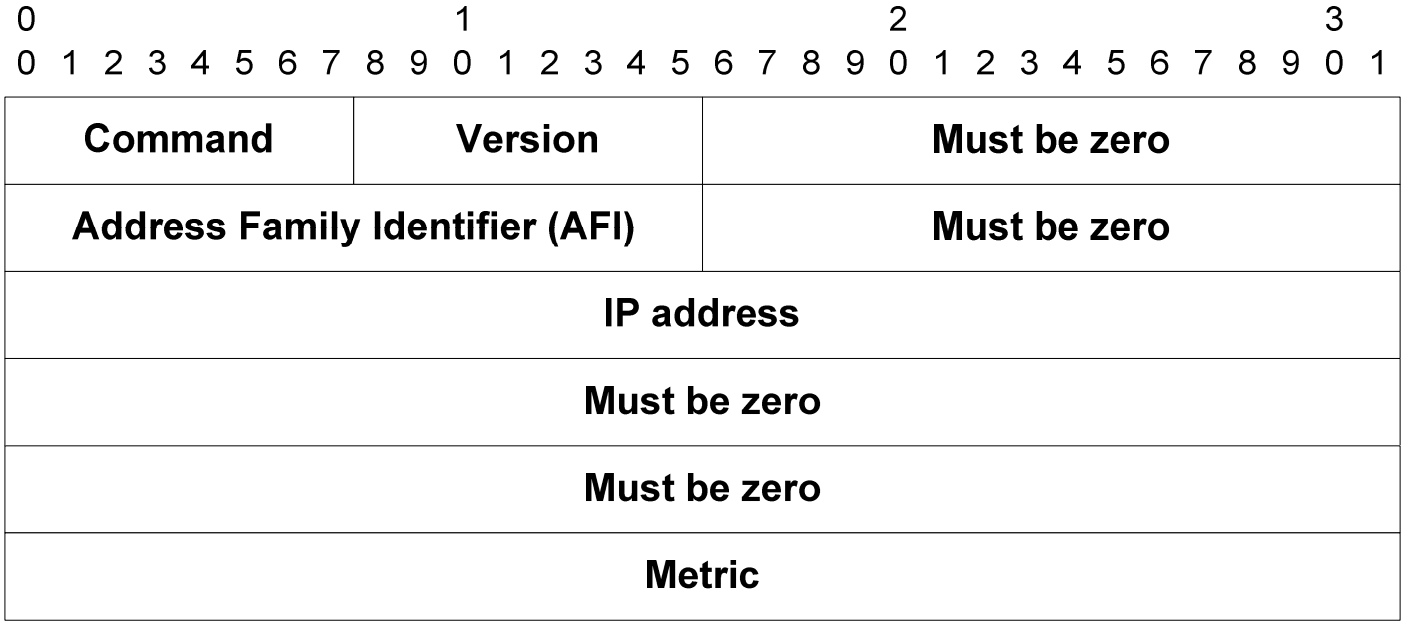
- Métrica por padrão
  - Distância em número de enlaces, ou saltos, para o destino (*hop count*)
  - Inteiro variando entre 1 e 15
  - 16 = "infinito"
    - O baixo valor dificulta a implementação de métricas mais complexas
- Suporta enlaces ponto-a-ponto e de difusão

# Características Básicas do RIPv1

Instituto de computação

- Mensagens RIP
  - UDP Porta 520, para emissão e recepção
    - Porta abaixo de 1024 – processos privilegiados apenas (BSD)
  - **Enviadas em *broadcast***
    - Ex. todos os roteadores em um segmento Ethernet as recebem
  - A cada 30 s (+ rand(1 to 5s))
    - Em 180 s a entrada torna-se inválida (métrica = inf.)

# Formato das Mensagens




- Command
  - Pedido (request code = 1)
  - Resposta (response code = 2)
- Version
  - Igual a 1

- Entradas de rotas (20 bytes cada)
  - Address Family Identifier (AFI)
  - Endereço IP
  - Métrica (32 bits), 0→15 (!?)

# Processamento das Mensagens RIP

Instituto de computação

- *Broadcast* de respostas
  - A cada **30 s** ou disparadas por atualizações
  - Respostas atualizam entradas na tabela

- Entradas na tabela
  - Endereço do destino
  - Métrica
  - Endereço do próximo roteador (próximo salto)
  - *Flag*: "atualizada recentemente"
  - Temporizadores

# Processamento das Mensagens RIP

Instituto de computação

- Ao receber a resposta, entradas de rota  analisadas uma a uma
  - Endereço válido? (classe A, B ou C)
  - Número de rede diferente de 127 e zero (exceto 0.0.0.0)?
  - Parte estação do endereço diferente de 255 (broadcast)?
  - Métrica menor ou igual a infinito (16)?

- Se sim a todas
  - Procura-se a entrada na tabela de roteamento e processa-se o vetor de distância

# Processamento do DV

Instituto de computação

- Se a entrada não está na tabela e a métrica não é infinito
  - *Criar* a entrada, com a *métrica* recebida, *próx. salto* o roteador que enviou o DV, iniciar *temporizador* pra essa entrada
- Se a entrada já existe com métrica maior que o DV
  - Atualizar a *métrica* e o *próx. salto* e reiniciar o *temporizador*

# Processamento do DV

Instituto de computação

- Se a entrada já existe e o próx. salto é o roteador que enviou o DV
  - Atualizar a *métrica* se esta mudou, reiniciar o *temporizador*
- Senão, esta entrada de rota do DV é ignorada

# Processamento das Mensagens RIP

Instituto de computação

- Se após o processamento do DV, a *métrica* ou o *próximo salto* mudaram
  - Entrada é marcada como "atualizada recentemente" (flag)

- Métricas iguais
  - RFC-1058: heurística
    - Se a métrica recebida é igual com próximo salto diferente, mas a entrada está próxima do estouro do temporizador, atualizar a entrada aceitando o novo próximo salto

# Geração das Respostas

Instituto de computação

- A cada 30s, ou ***disparada***
  - Rajada de respostas disparadas
    - Aumento excessivo da carga da rede
  - Para evitá-la, resposta não é disparada imediatamente mas entre 1 e 5 s após a atualização da tabela
  - Além disso, *updates* recebidos de outros vizinhos neste intervalo podem ser incluídos no DV
    - Diminuição adicional da carga da rede
- Uma resposta é gerada por interface
  - *Split horizon*
  - Resumo de sub-redes

# Geração das Respostas

Instituto de computação

- A resposta normalmente inclui *todas* as entradas da tabela de roteamento
  - Exceção: respostas disparadas incluem apenas as entradas modificadas
    - Uso do flag "atualizada recentemente"
- Tamanho máximo: 512 bytes
  - Equivale a 25 entradas por mensagem
  - Mais de 25 entradas: várias mensagens de resposta
- Endereço origem
  - ***Deve*** ser o da interface

# Geração das Respostas

Instituto de computação

- Entradas de sub-redes
  - O RIPv1 supõe que as máscaras de sub-redes não são conhecidas fora desta rede
  - Só são anunciadas se a interface pertence à mesma ***rede*** que a sub-rede
  - Em outras interfaces
    - Todas as entradas de sub-rede devem ser resumidas em uma rota de rede

# Geração das Respostas

Instituto de computação

- Entradas com métrica infinito
  - Só devem ser anunciadas se modificadas recentemente
    - Não há problema em deixá-las "morrer"
    - Diminuição da carga da rede
  - O mesmo se aplica a entradas anunciadas com infinito devido ao *split horizon*
    - Só precisam ser anunciadas se o próx. salto mudou recentemente

# Mensagens de Pedido no RIP

Instituto de computação

- Pedidos RIP (*requests*)
  - Normalmente utilizados quando um roteador é ligado
  - Obtém-se um valor inicial para a tabela de roteamento

# Mensagens de Pedido no RIP

Instituto de computação

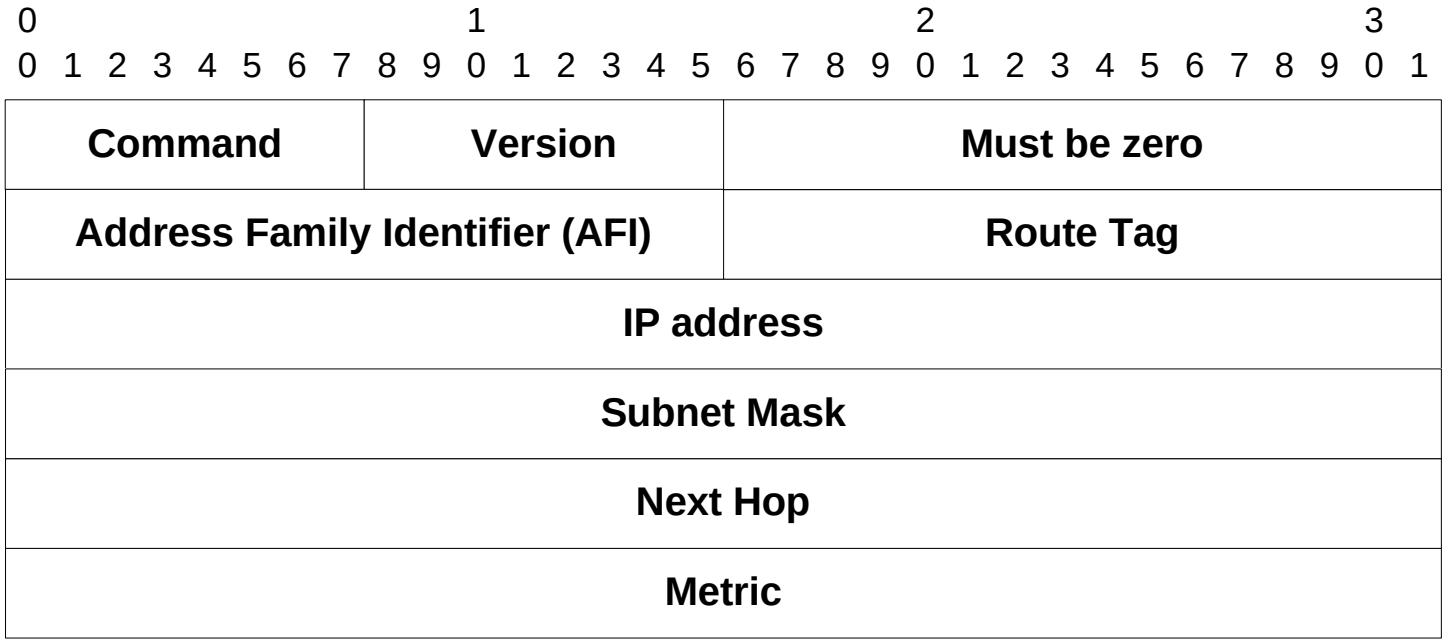
- Tipos de pedidos
  - Pedido de toda a tabela
  - Pedido de rotas específicas
- Pedido completo
  - Endereço **`0.0.0.0`**, métrica infinito
  - Provoca uma resposta “normal”
- Pedido específico
  - Resposta contém apenas as entradas pedidas
    - Enviada em ponto-a-ponto
    - Mais utilizada para diagnóstico de problemas

# RIP Versão 2

Instituto de computação

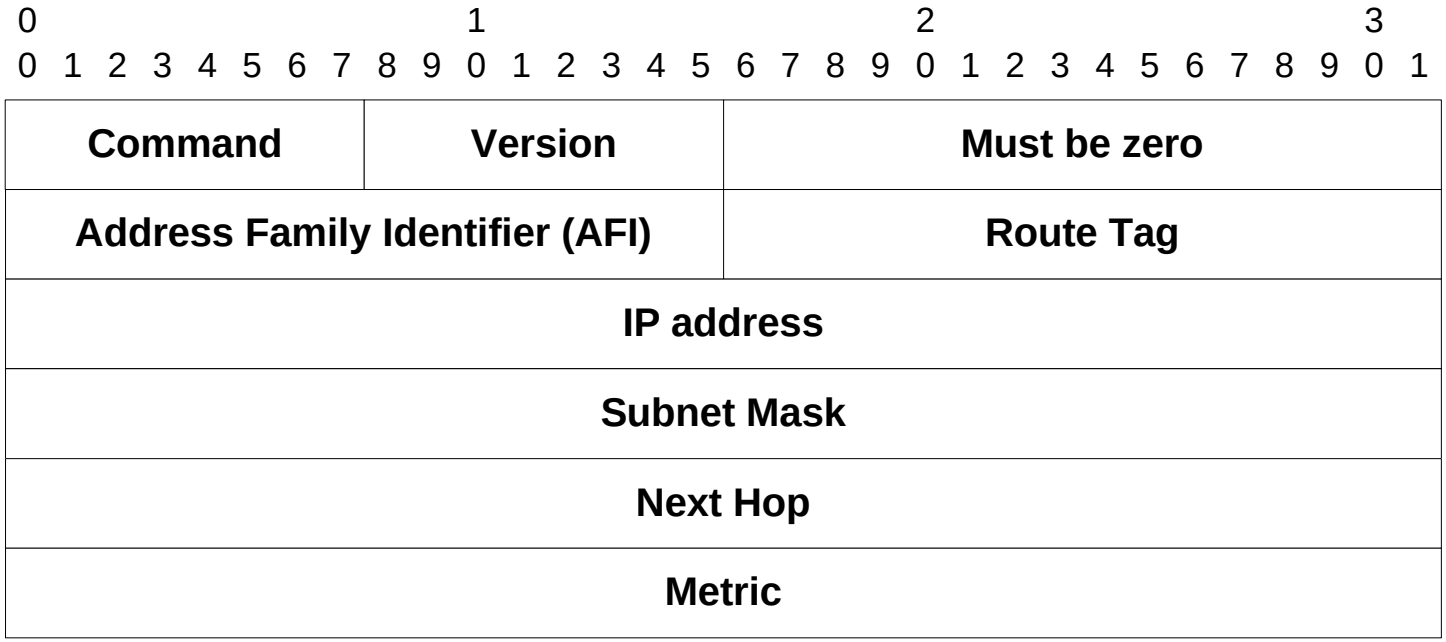
- RFC-1388 - RIP Version 2 Carrying Additional Information
  - *Updates* RFC-1058
  - *Obsoleted by* RFC-1723 (*Obsoleted by* RFC-2453)
- RFC-1389 - RIP Version 2 MIB Extension
  - Estruturas de dados para gerenciamento
- RFC-1387 – RIP Version 2 Protocol Analysis
  - *Obsoleted by* RFC-1721
  - Informational

# Formato das Mensagens




- Campos em comum com o RIPv1
  - AFI (Address Family Identifier)
    - Contém um código para dados de autenticação
  - Endereço IP
  - Métrica

# Formato




- Novos campos
  - Próximo salto (Next Hop): elimina saltos duplos na mesma sub-rede
  - Máscara (Subnet Mask): melhora o roteamento por sub-rede
  - Route Tag: marca rotas externas (utilizado com BGP/EGP)

# Compatibilidade

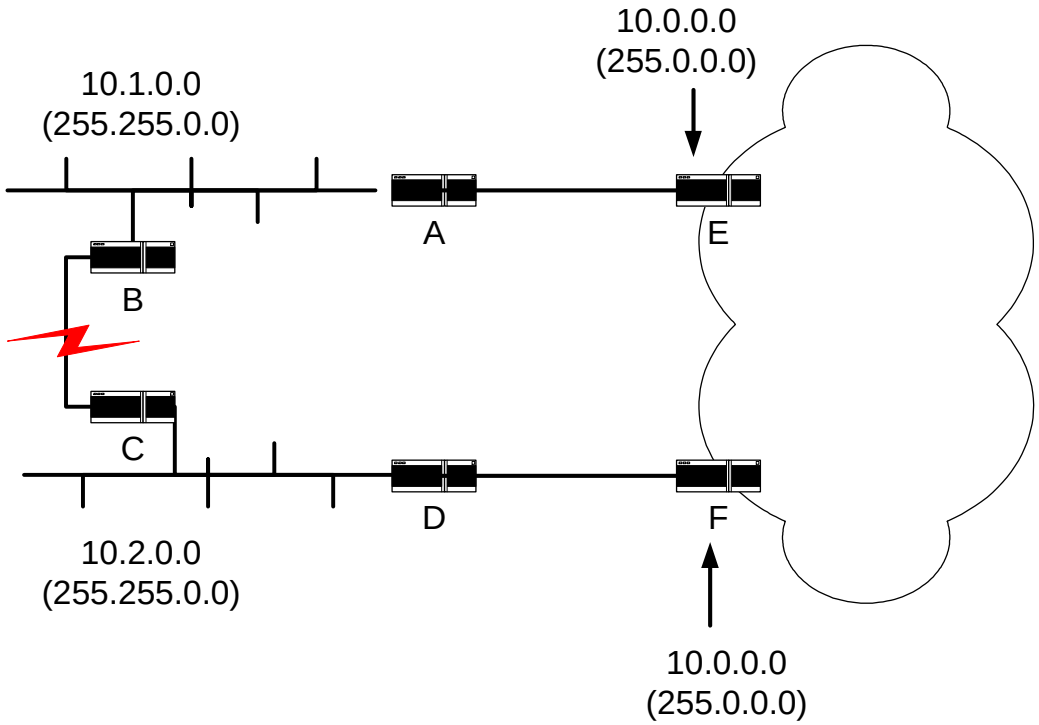
Instituto de computação

- Versão = 2
- Entradas com campos “must be zero” no RIPv1 são ignoradas por roteadores RIPv1
  - Entradas sem “opções” emitidas pelo RIPv2 são entendidas
    - (entradas de rede sem sub-rede)

# Roteamento por Sub-rede

- RIPv1
  - sub-redes não podem ser anunciadas pra fora
  - Roteadores de fora sempre utilizam o roteador mais próximo, independente da sub-rede

- A e D anunciam rota para 10.0.0.0
- Pacote para 10.2.0.1 pode passar por E ou F
- Se o pacote chegar por E, o roteador B enviará um ICMP destination unreachable




- RIPv2
  - Subnet mask permite o roteamento por sub-rede, CIDR
  - Entradas de sub-rede são ignoradas por roteadores RIPv1

# Autenticação

Instituto de Computação

- RIPv1 é inseguro
  - Basta ter acesso a uma máquina em super-usuário
    - Envio na porta UDP 520
  - Exemplo de problema
    - Envio de vetores com distância 0 para todos os destinos

- RIPv2
  - Primeira entrada de rota da mensagem RIP
    - Substituída por um "segmento de autenticação"

# Multicast

Instituto de Computação

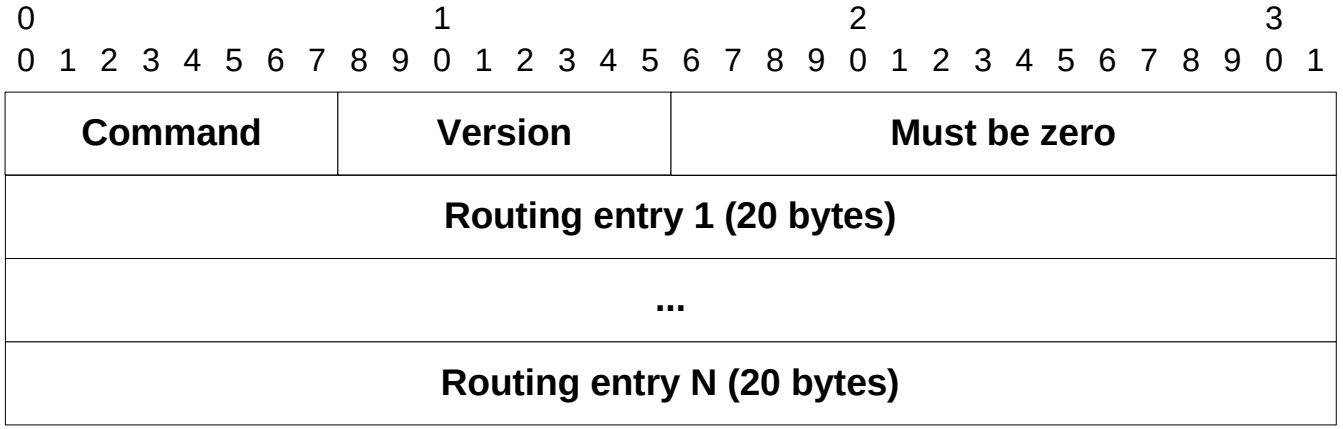
- RIPv2 utiliza o endereço 224.0.0.9 em vez de broadcast
  - Evitar que todas as máquinas num segmento Ethernet recebam os pacotes RIP
- Problema
  - Compatibilidade com RIPv1
- RFC-1388 – Três modos de operação
  - Envio de pacotes **RIPv1 em broadcast**
    - Compatibilidade total
  - Envio de pacotes **RIPv2 em broadcast**
    - Transição entre v1 e v2
    - Roteadores RIPv1 recebem todos os pacotes, mas em alguns casos tratam partes deles apenas
  - Envio de pacotes **RIPv2 em multicast**
    - Todos os roteadores da rede são RIPv2

# RIPng (IPv6)

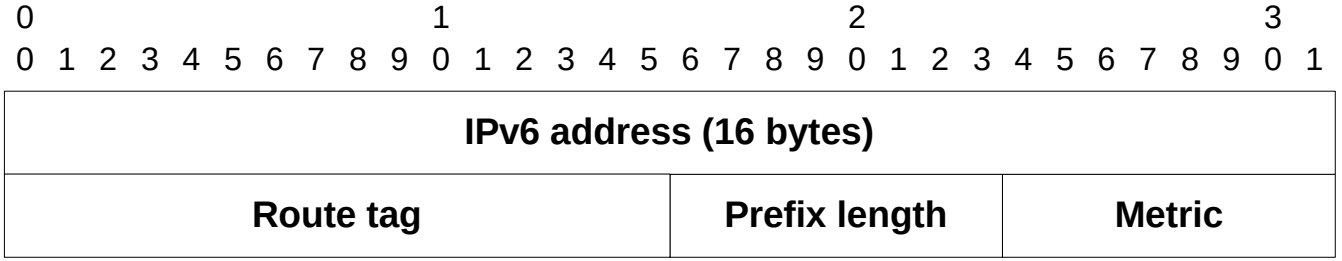
Instituto de computação

- Bastante semelhante ao RIP para IPv4, porém
  - Utiliza mecanismos de segurança do IPv6 em vez de entradas de autenticação
  - Os formatos de pacotes devem ser adaptados
    - Endereços IPv6 possuem 128 bits
- Segurança
  - Cabeçalhos de autenticação protegem todo o pacote IP
  - Também pode ser usado o serviço de criptografia
  - Conseqüências
    - Mecanismo de senha simples descartado
    - Não é necessário diferenciar entradas de rotas de entradas de autenticação
      - Protocolo mais simples, não há necessidade do campo AFI (*Address Family Identifier*)

# Mudanças no Formato



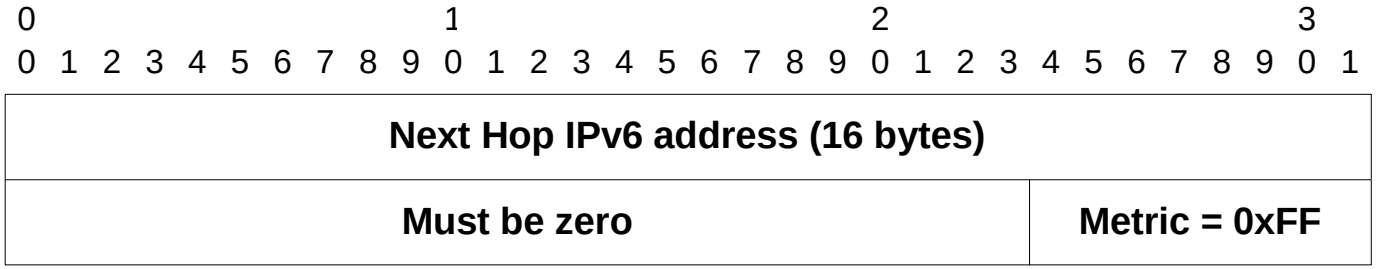

- Command e Version como no RIPv1 e v2
- Não há AFI




  - IPv6 address + prefix length identificam a rota
  - Route tag sinaliza rotas externas, como no RIPv2
  - Métrica (1 byte)
    - 0 a 16 como nos RIPv1 e v2

# Mudanças no Formato

- Next hop
  - Não é um campo como no RIPv2, mas uma entrada de roteamento especial
    - Evita 16 bytes de overhead em toda mensagem RIP




    - Identificada pela métrica 255

  - A informação de próximo salto vem ***antes*** da entrada de roteamento que ela qualifica

# Problemas de Sincronização

Instituto de computação

- Van Jacobson and Sally Floyd, *"Synchronization of Periodic Routing Messages",* SIGCOMM'93

- Problema
  - Medidas com sessões de voz e sondas "ping":
  - A cada 30s, a rede exibia longos atrasos ou altas taxas de perda
    - Como se houvesse congestionamento a cada 30s

- Diagnóstico
  - ***Todos*** os roteadores RIP enviavam seus vetores de distância (*updates*) a cada 30s, ***sincronizados***

# Problemas de Sincronização

Instituto de Computação

- Efeitos
  - Aumento de tráfego de controle
    - Atrasos de encaminhamento maiores
    - Monopolização dos recursos dos roteadores
      - Perda de pacotes com opções especiais
  - Uma das implementações não enviava pacotes enquanto a tabela não estivesse atualizada...
    - Comportamento errado, pois não existe garantia absoluta de que uma rota esteja atualizada, de qualquer forma
- Problema
  - Como evitar a sincronização?

# Problemas de Sincronização

Instituto de computação

- Projeto do RIPv1 (RFC-1058)
  - Roteadores iniciam em momentos aleatórios
  - Envio de updates precisamente a cada 30s
  - Se o roteador não consegue manter a periodicidade exata
    - Adicionar um pequeno intervalo de tempo, aleatório
      - O intervalo em geral era bem pequeno, o objetivo era evitar colisões em redes de difusão

# Problema Observado

Instituto de computação

- Após a expiração do temporizador de 30s
  - Preparo do vetor de distância
  - Se durante esta fase, um update é recebido
    - processado imediatamente
  - Temporizador é re-iniciado após a emissão do DV

- Na prática
  - Período T = 30s + $t_E$ + $t_R$
  - $t_E$ = tempo de emissão do DV
  - $t_R$ = tempo de processamento dos DVs recebidos
  - Quanto mais DVs recebidos, maior é o $t_R$
    - Quanto maior o grupo de roteadores sincronizados, maior é o $t_R$

# Problema Observado

Instituto de computação

- Conseqüência
  - T(grupo de roteadores) > T(roteador isolado)
  - Depois de algumas interações
    - Envio do roteador isolado coincide com o do grupo
    - Entrada em sincronia
  - Quanto maior o grupo, mais facilmente roteadores entram no grupo
  - Todos os roteadores da rede terminam por se sincronizar...

# Evitando a Sincronização

Instituto de computação

- Solução
  - Para o roteador sair do grupo, deve usar um período diferente
  - RFC-1058 recomenda um tempo aleatório, porém muito pequeno
    - Não era suficiente para quebrar a sincronização
- Jacobson e Floyd
  - Período aleatório deve ser maior que o período de processamento de DVs...
    - Dependente do tamanho do grupo de roteadores
    - Do tamanho da rede, em última análise
  - Recomendação:
    - Tempo aleatório entre 15 e 45s
      - Média de 30s mantida

Instituto de computação

Programa de Pós-Graduação
Instituto de Computação
Universidade Federal Fluminense

**Aula 17**

# Camada de Rede

## Roteamento intradomínio: vetor de distância e RIP

Igor Monteiro Moraes

Redes de Computadores