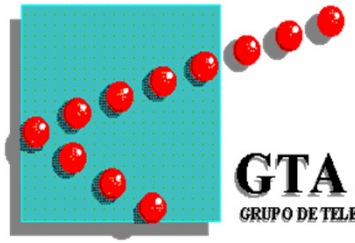


Roteamento em Redes de Computadores
CPE 825

Parte II
Roteamento Unicast na Internet
Vetores de Distância

Luís Henrique M. K. Costa

luish@gta.ufrj.br

Universidade Federal do Rio de Janeiro -PEE/COPPE
P.O. Box 68504 - CEP 21945-970 - Rio de Janeiro - RJ
Brasil - http://www.gta.ufrj.br

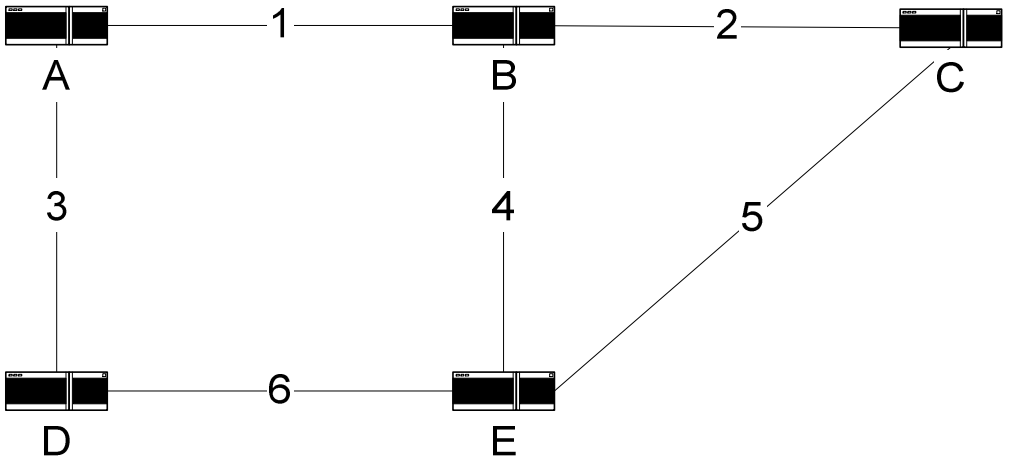
Algoritmos de Roteamento

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

Topologia

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

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



Suponha que todos os enlaces possuem custo igual a 1.

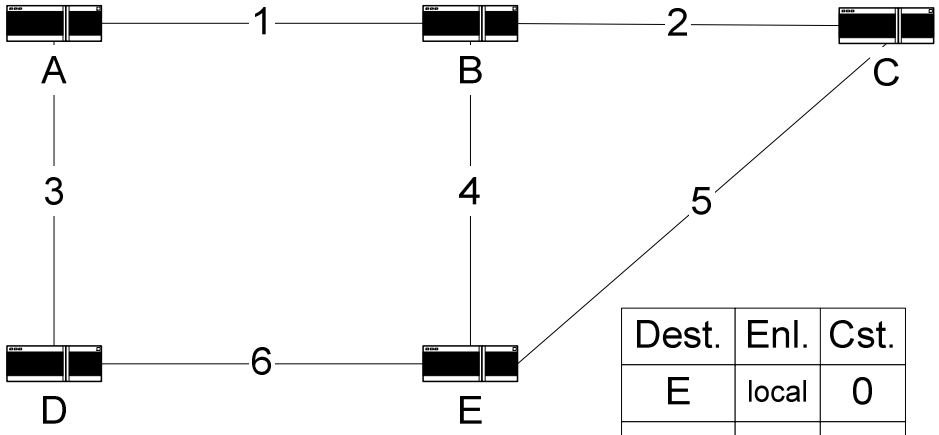
Exemplo – DV

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.

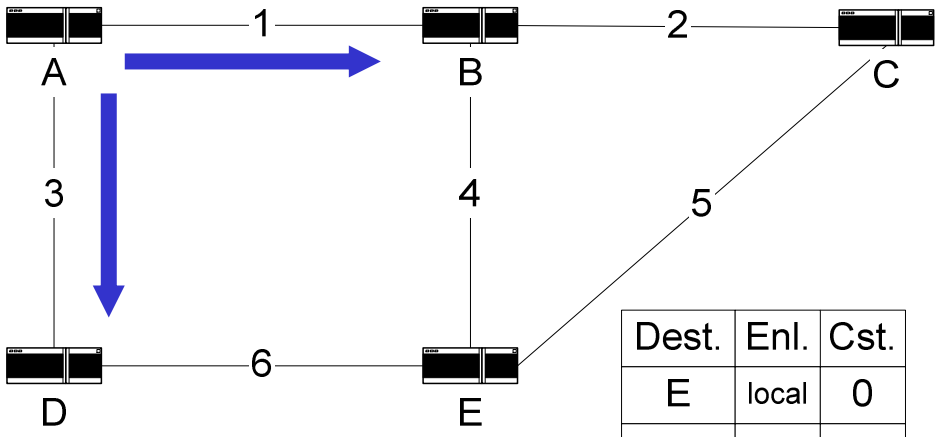
Exemplo – DV

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)

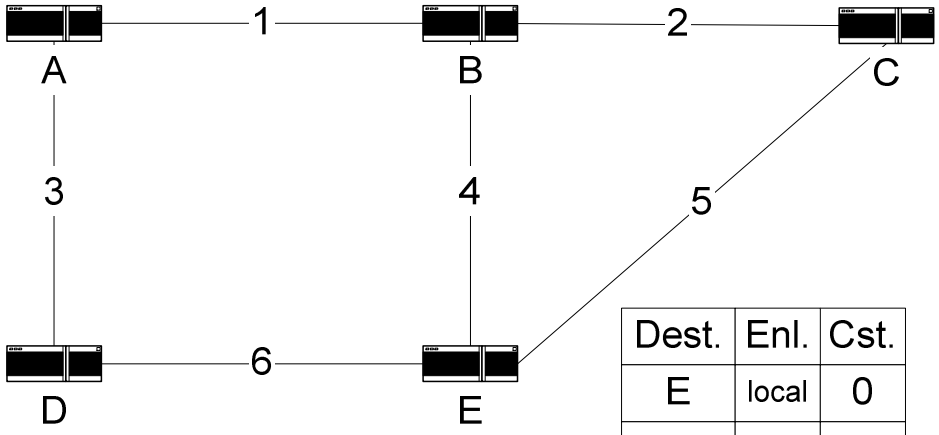
Exemplo – DV

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**.

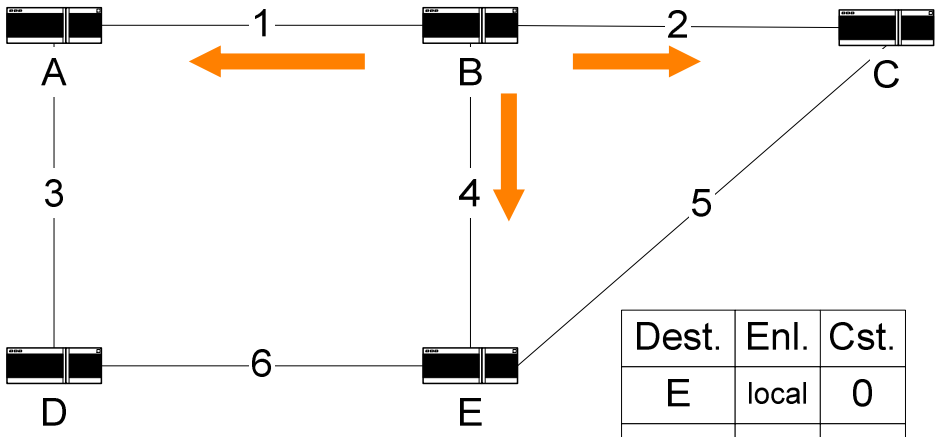
Exemplo – DV

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



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

Dest.	Enl.	Cst.
E	local	0

B envia primeiro.

(B=0, A = 1)

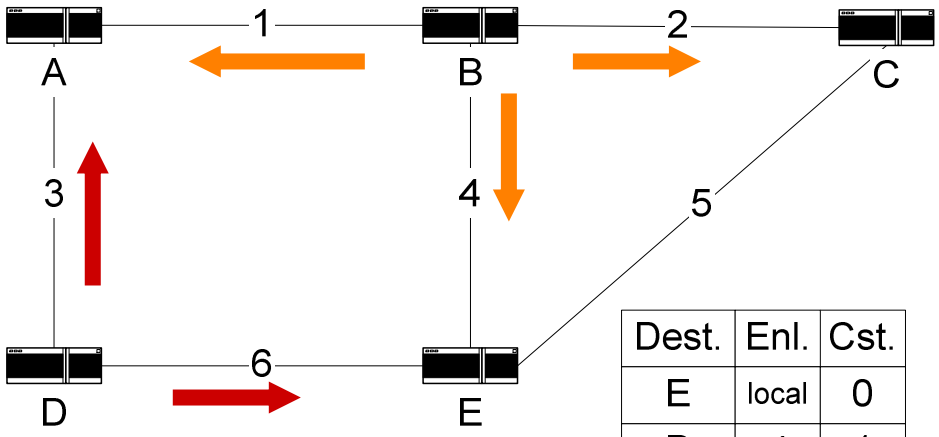
Exemplo – DV

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

(B=0, A = 1)

D envia seu vetor.

(D=0, A = 1)

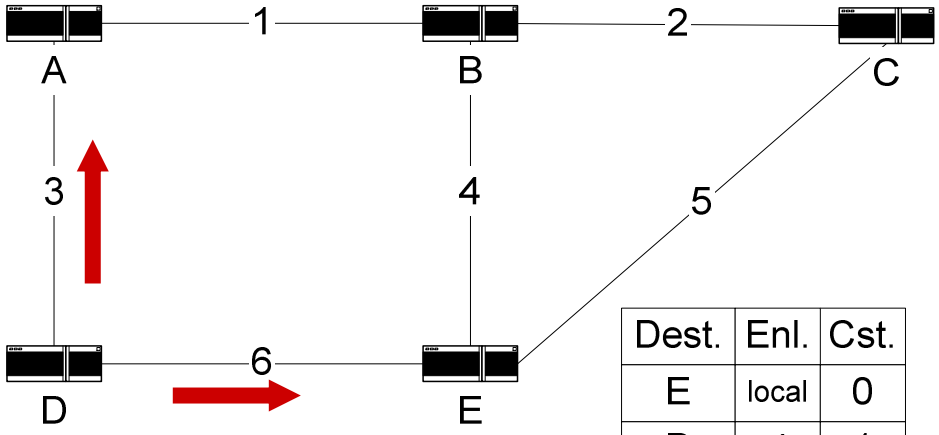
Exemplo – DV

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**.

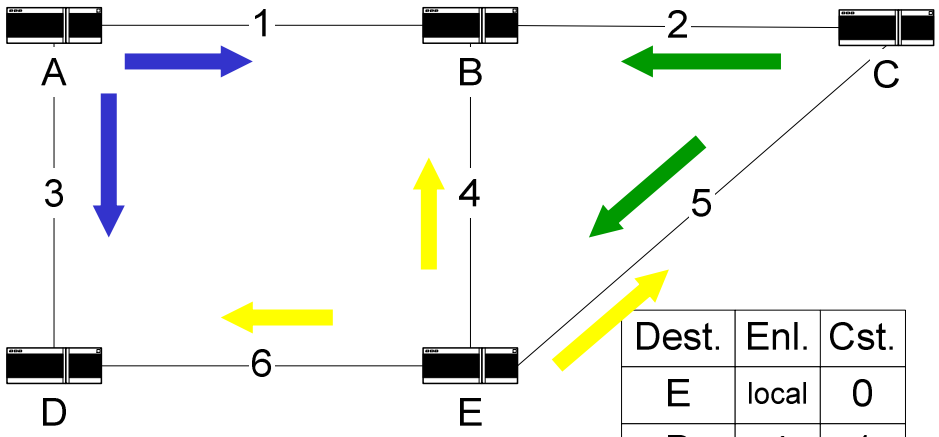
Exemplo – DV

A, C e **E** preparam vetores de distância 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)

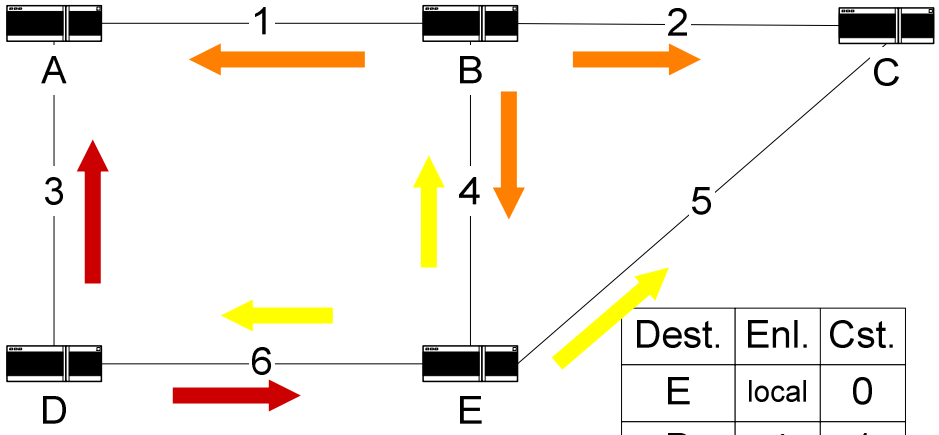
Exemplo – DV

B, D e **E** atualizam suas tabelas e enviam vetores de distância.

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)

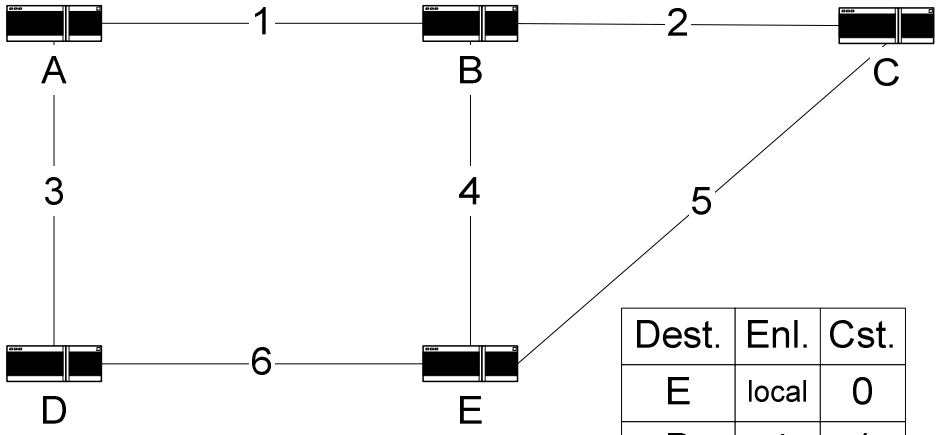
Exemplo – DV

A, C e **D** atualizam suas tabelas e enviam vetores de distância.

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.

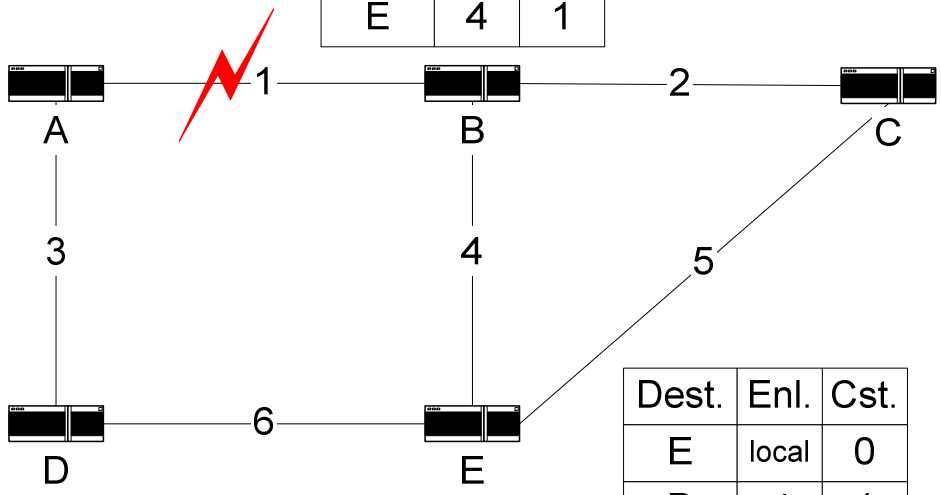
Falha de um Enlace

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

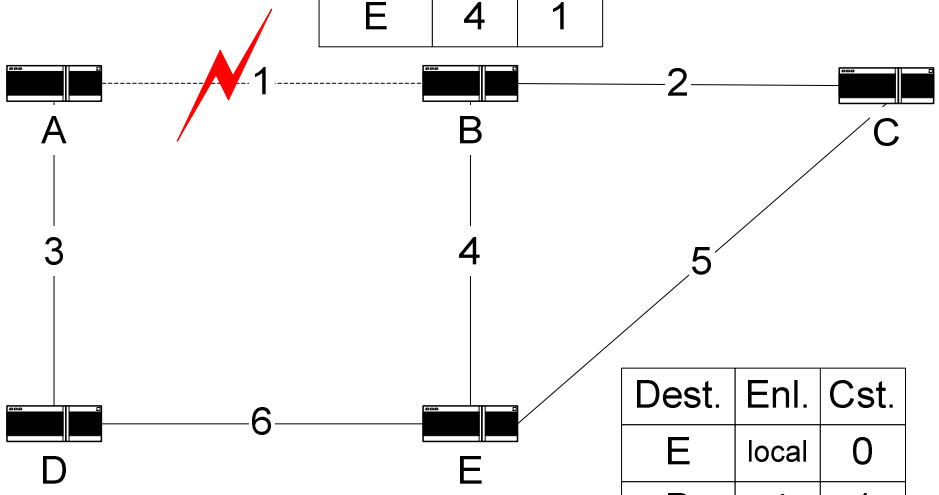
Falha de um Enlace

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.

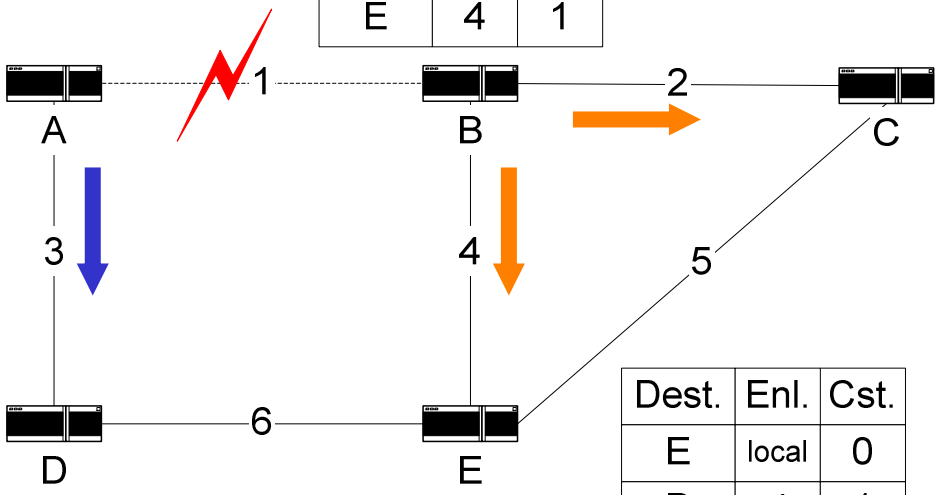
Falha de um Enlace

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)

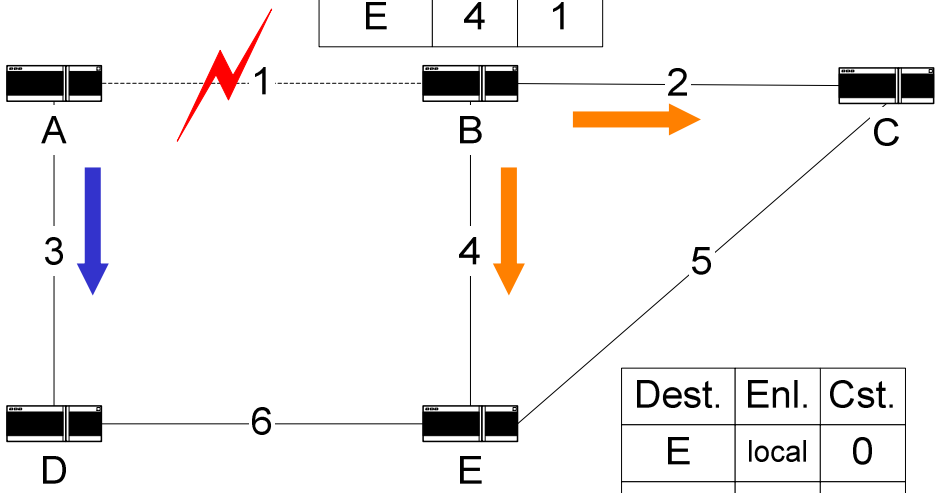
Falha de um Enlace

D atualiza sua tabela. Apenas a rota para **B** é modificada (utiliza 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)**

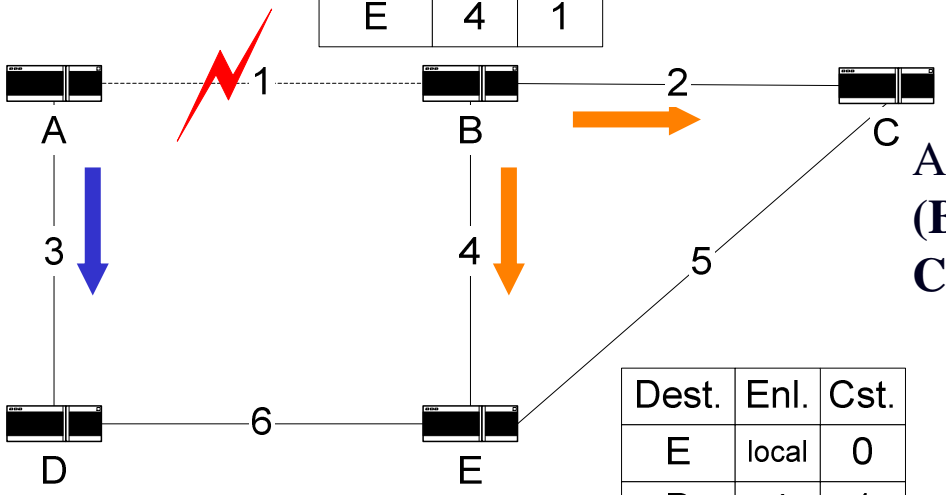
Falha de um Enlace

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)

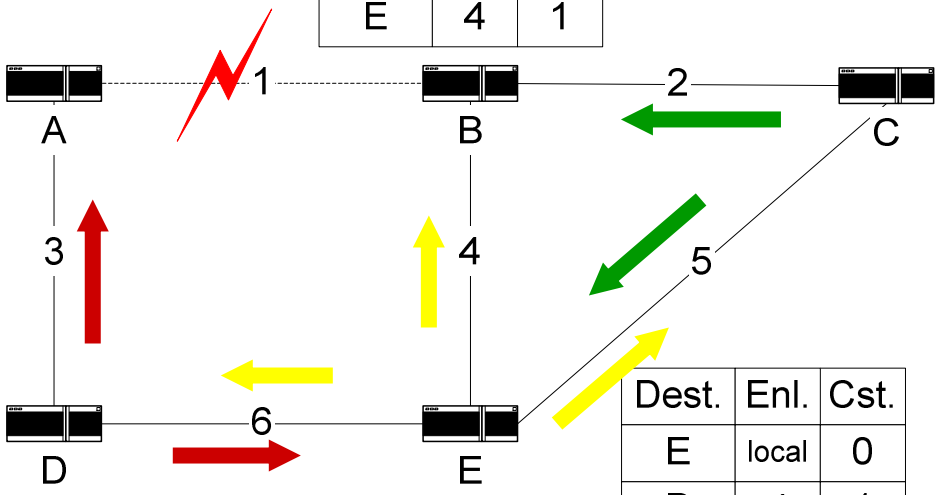
Falha de um Enlace

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)**

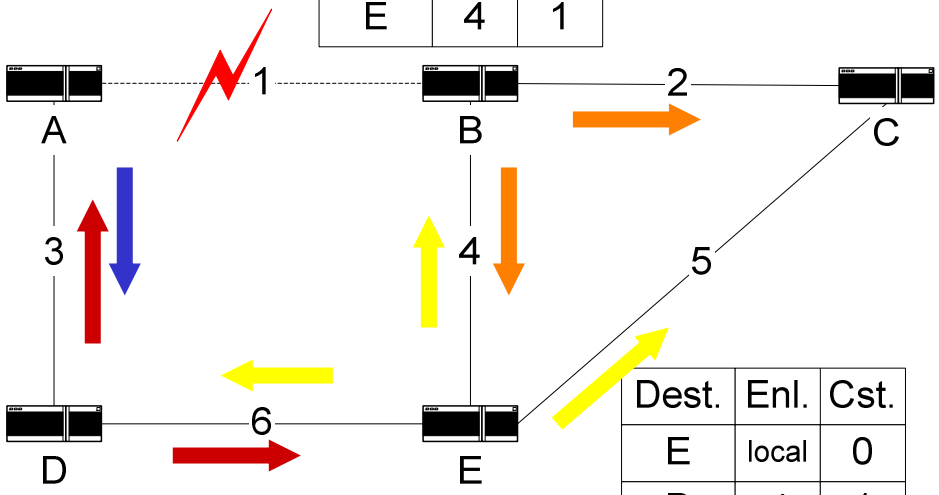
Falha de um Enlace

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

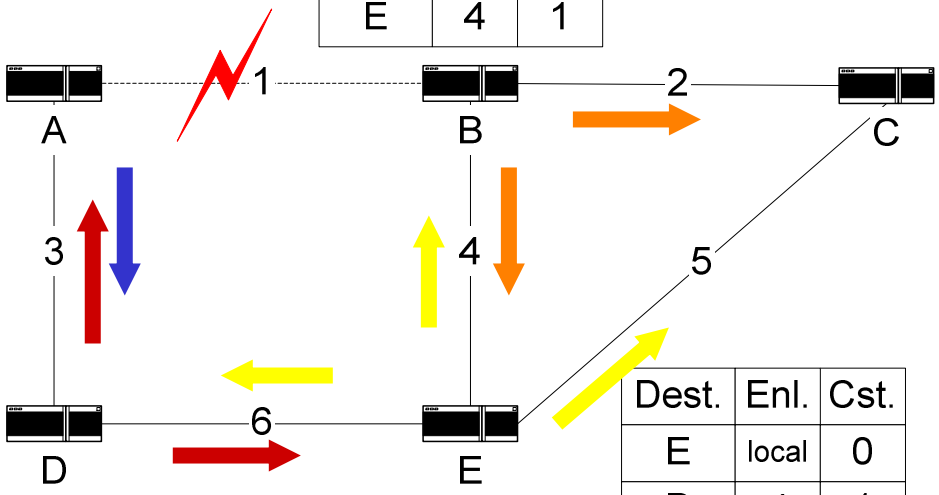
Falha de um Enlace

A, B e **C** atualizam suas tabelas, 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, não há mais mofificações nas tabelas, o algoritmo convergiu.

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

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

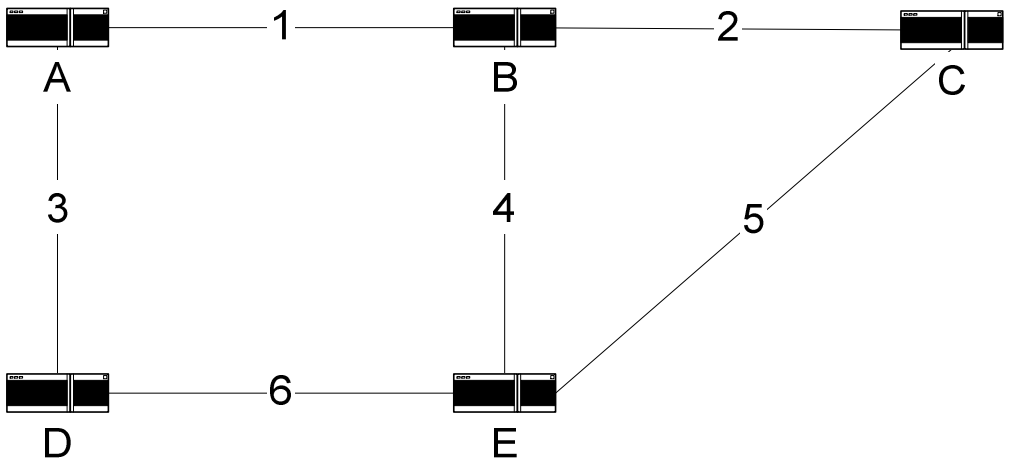
De D: (D=0, A=1, B=2, E=1, C=2)

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

Bouncing Effect

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

- Enlace **2** falha...



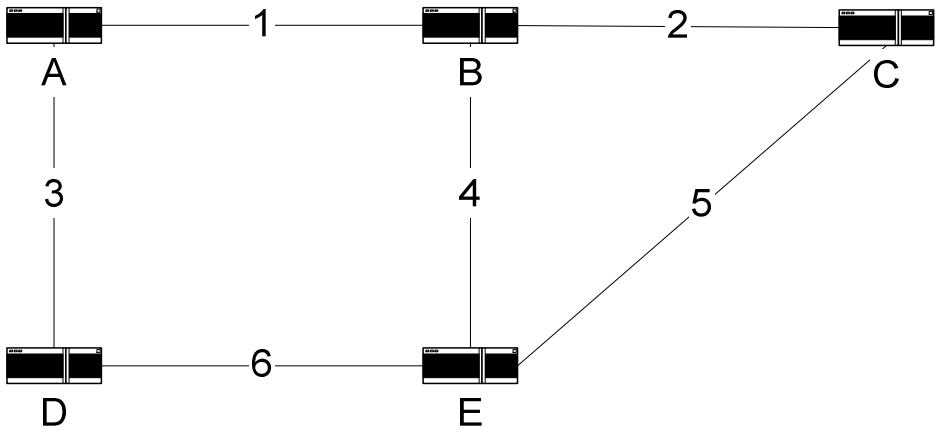
Bouncing Effect

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

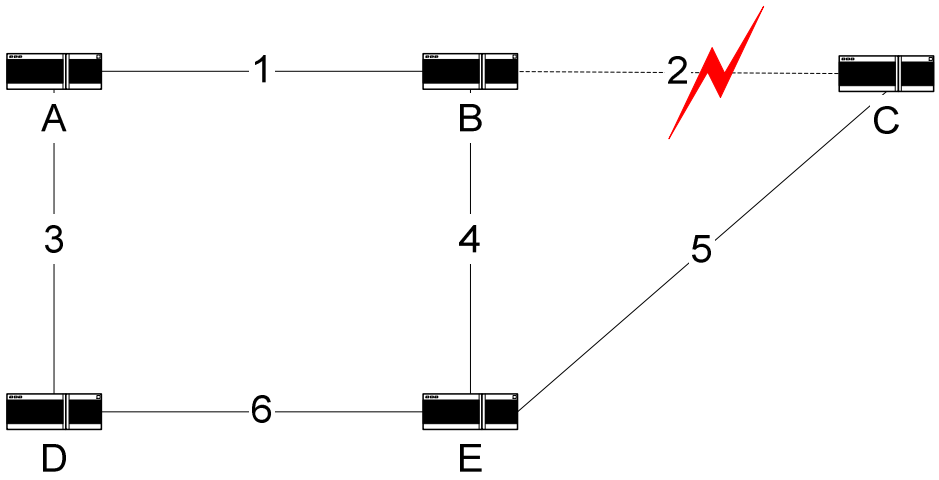
Bouncing Effect

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

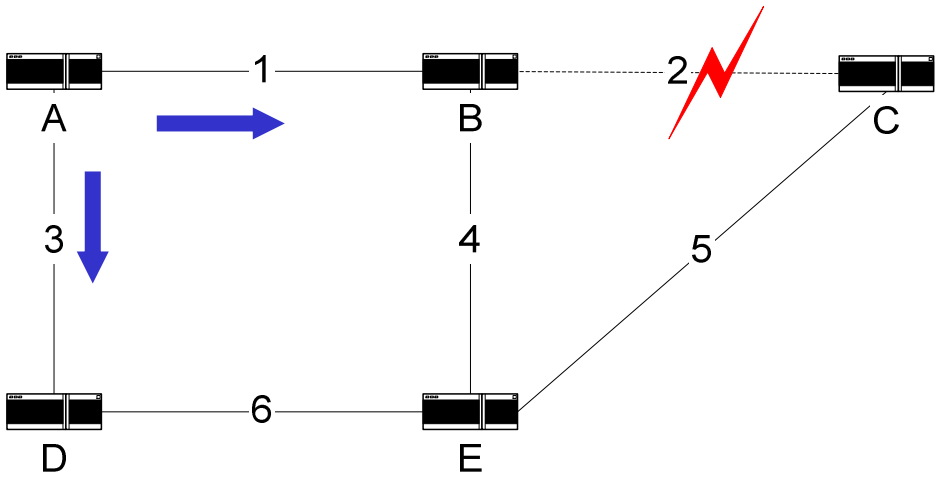
Suponha que **A** envia um vetor de distância.

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

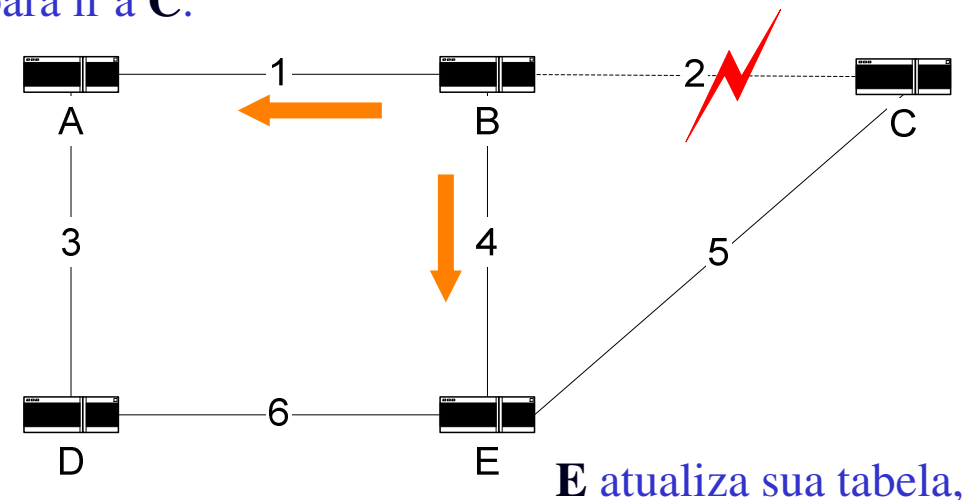
Dest.	Enl.	Cst.
C	1	3

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 pelo enlace utilizado para ir a **C**.

Dest.	Enl.	Cst.
C	3	3

Dest.	Enl.	Cst.
C	4	4

Bouncing Effect

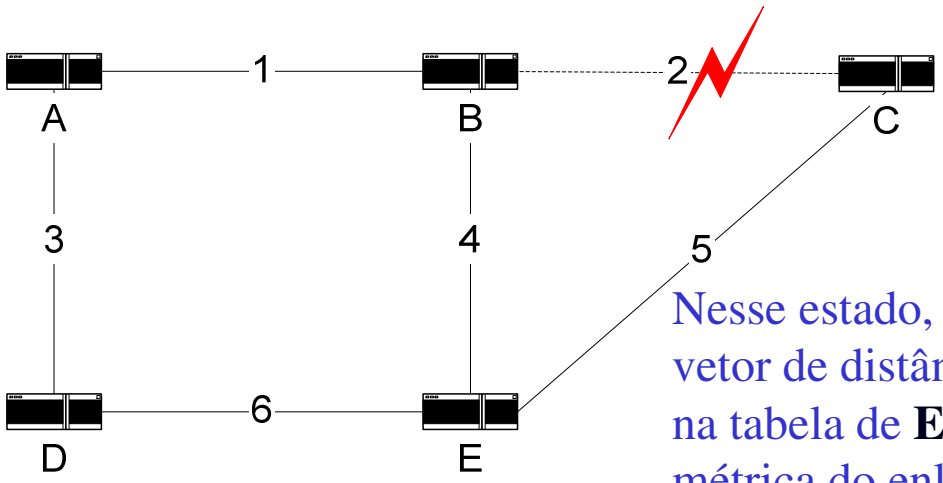
Nesse estado, a rede possui um loop de roteamento.

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 vetor de distância, 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

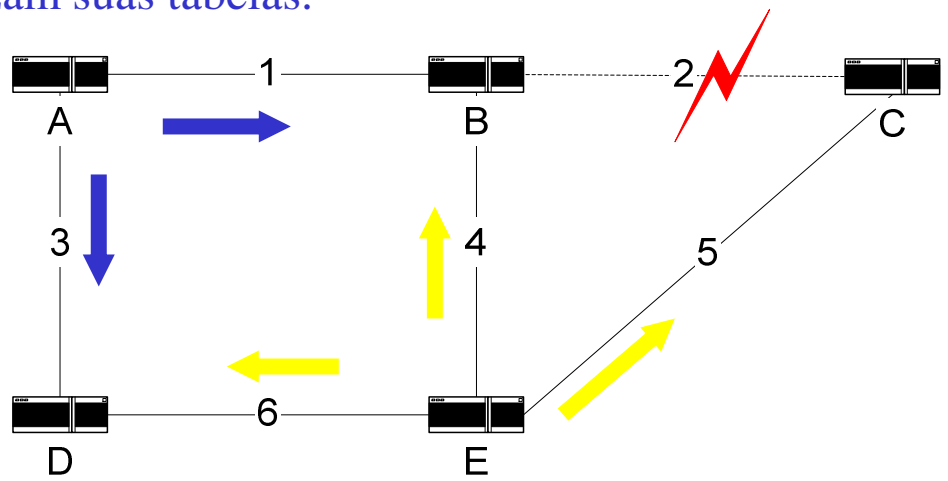
A e **E** enviam vetores de distância.

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

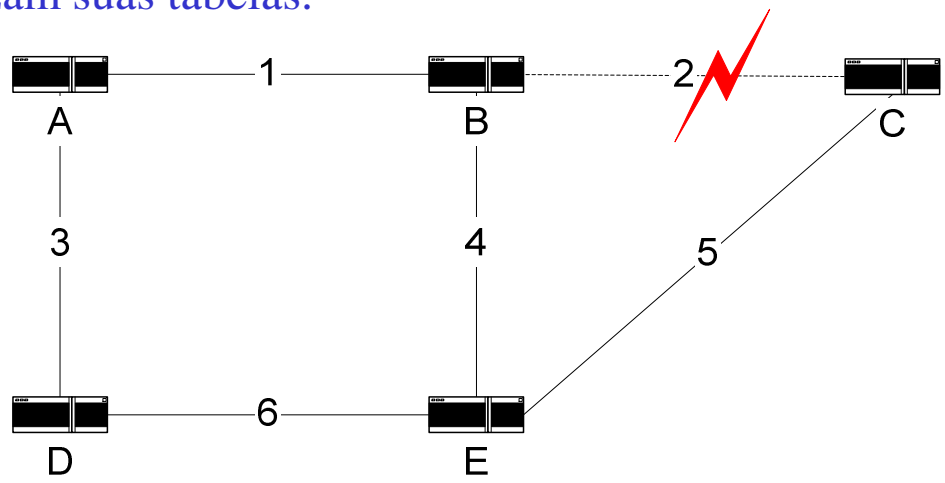
B e **D** enviam vetores de distância.

Dest.	Enl.	Cst.
C	1	5

Dest.	Enl.	Cst.
C	1	6

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

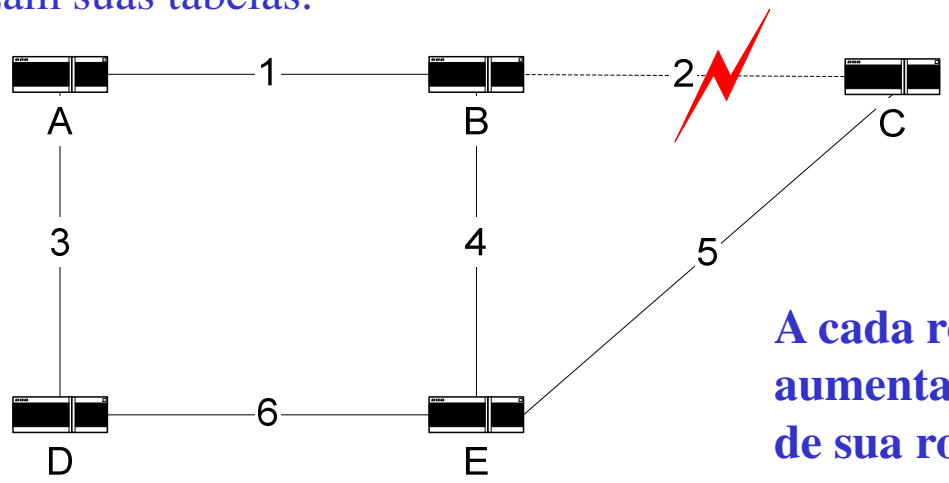
A e **E** enviam vetores de distância.

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

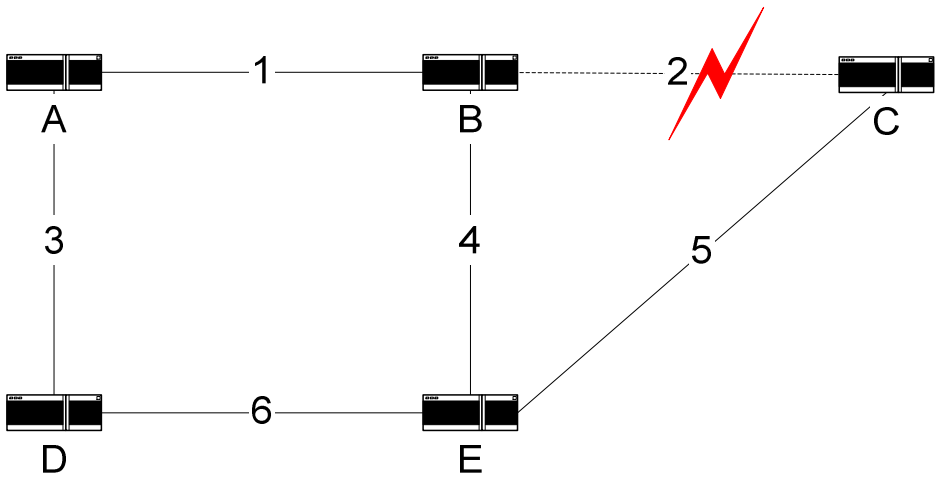
Bouncing Effect

Após mais uma rodada...

Dest.	Enl.	Cst.
C	1	8

Dest.	Enl.	Cst.
C	1	9

Dest.	Enl.	Cst.
C	local	0



Dest.	Enl.	Cst.
C	3	9

Dest.	Enl.	Cst.
C	4	8

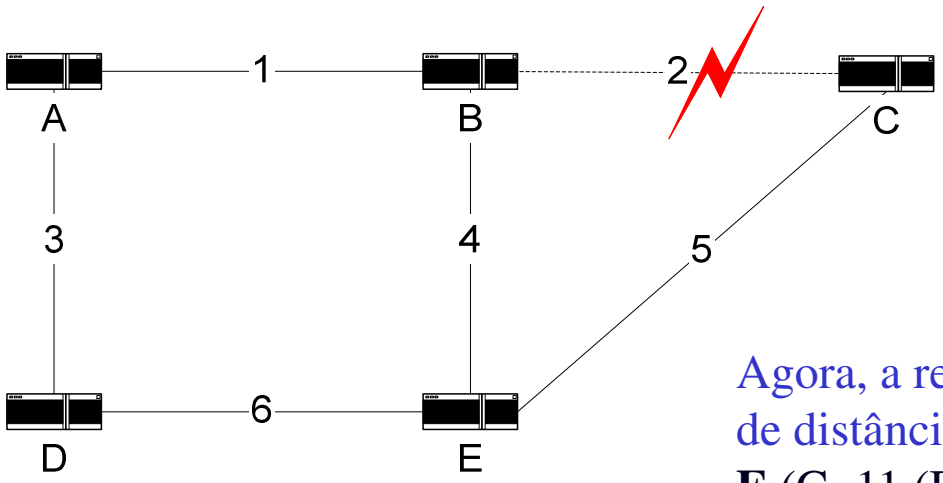
Bouncing Effect

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

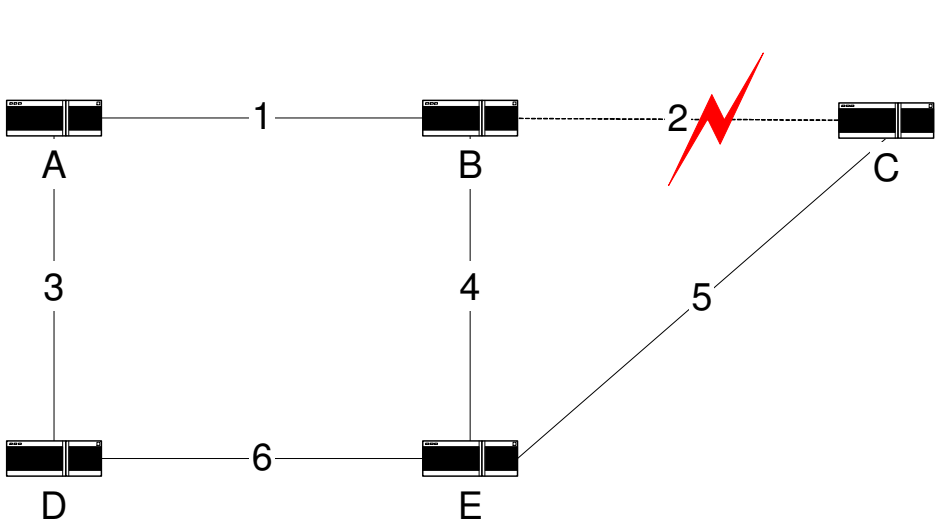
Dest.	Enl.	Cst.
C	4	12

Bouncing Effect

E atualiza sua tabela.

Dest.	Enl.	Cst.
C	1	12

Dest.	Enl.	Cst.
C	1	11



Dest.	Enl.	Cst.
C	local	0

Dest.	Enl.	Cst.
C	3	11

Dest.	Enl.	Cst.
C	5	10

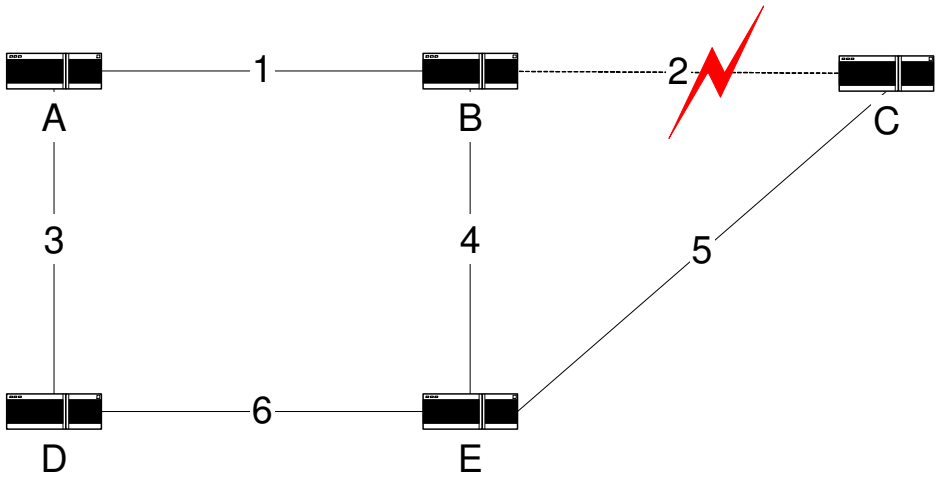
Bouncing Effect

Após mais alguns passos,
o algoritmo converge.

Dest.	Enl.	Cst.
C	1	12

Dest.	Enl.	Cst.
C	4	11

Dest.	Enl.	Cst.
C	local	0



Dest.	Enl.	Cst.
C	6	11

Dest.	Enl.	Cst.
C	5	10

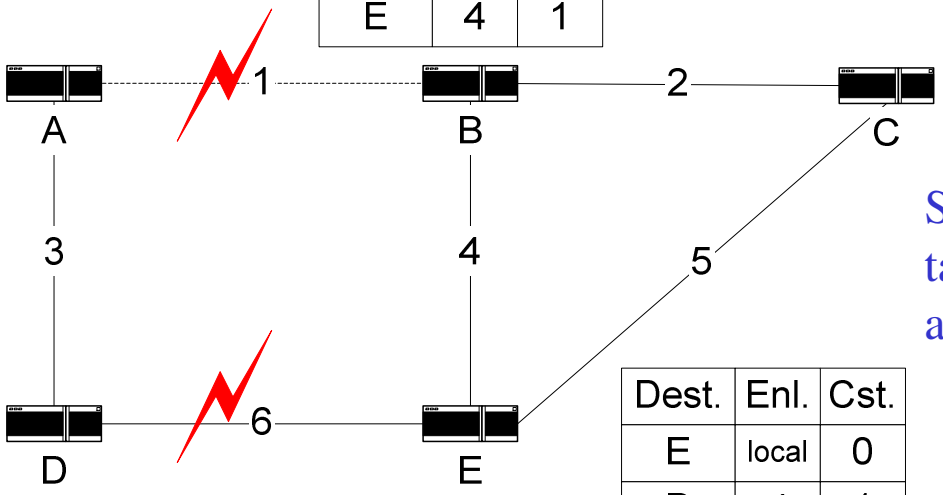
Contagem até o Infinito

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

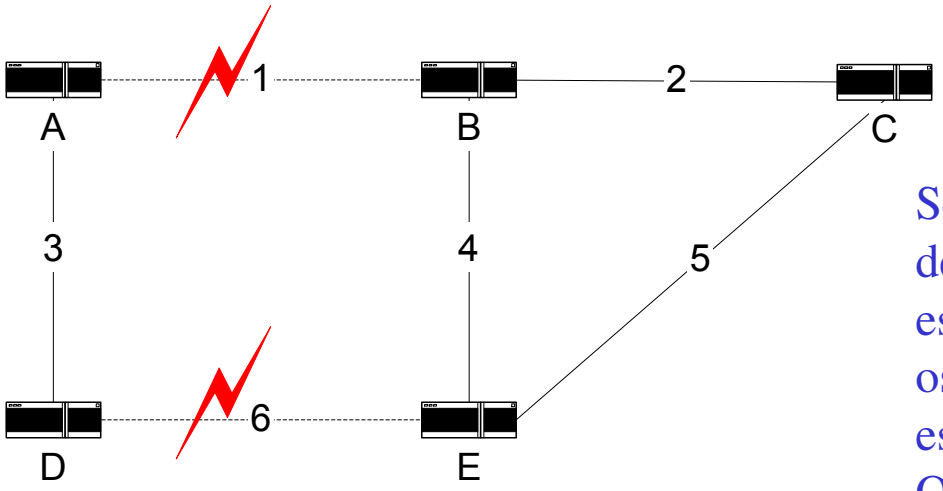
Dest.	Enl.	Cst.
E	local	0
B	4	1
A	6	2
D	6	1
C	5	1

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.



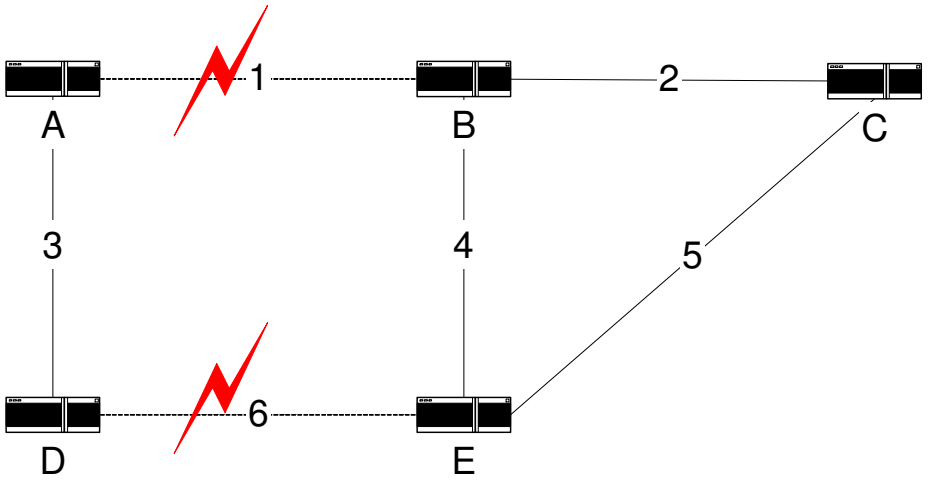
Dest.	Enl.	Cst.
D	local	0
A	3	1
B	6	inf.
E	6	inf.
C	6	inf.

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.



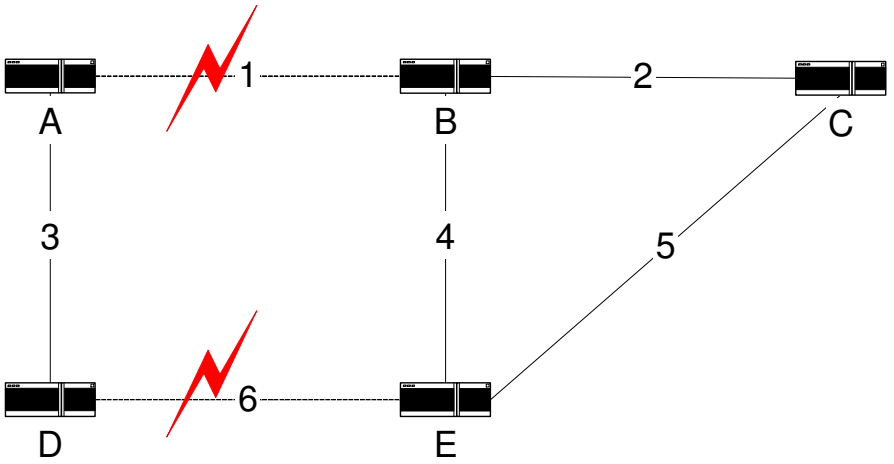
Dest.	Enl.	Cst.
D	local	0
A	3	1
B	3	4
E	3	3
C	3	4

Contagem até o Infinito

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

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

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

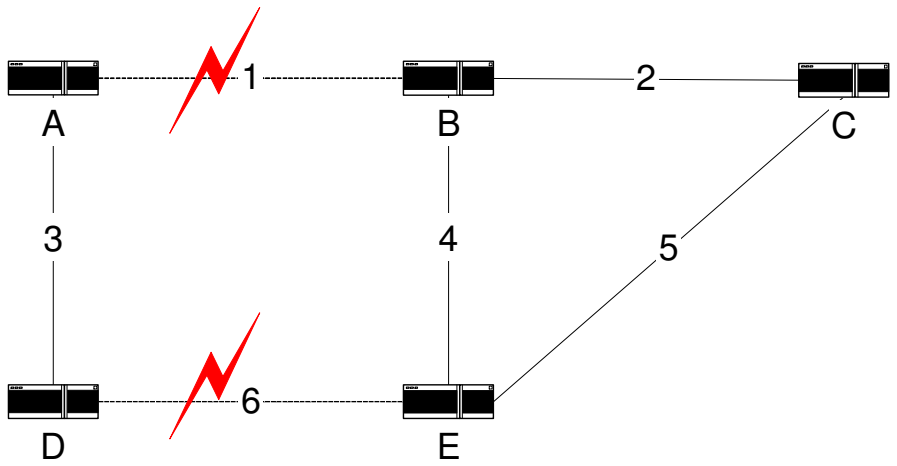


Dest.	Enl.	Cst.
D	local	0
A	3	1
B	3	4
E	3	3
C	3	4

Contagem até o Infinito

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.
A	local	0
B	3	7
D	3	1
C	3	7
E	3	6



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

Melhorias no Algoritmo BF

- Bouncing effect e contagem até o infinito
  - Aumento do tempo de convergência
- Melhorias no algoritmo
  - *Split horizon*
  - *Triggered updates*
- *Split horizon*
  - 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**

Split horizon

- 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...

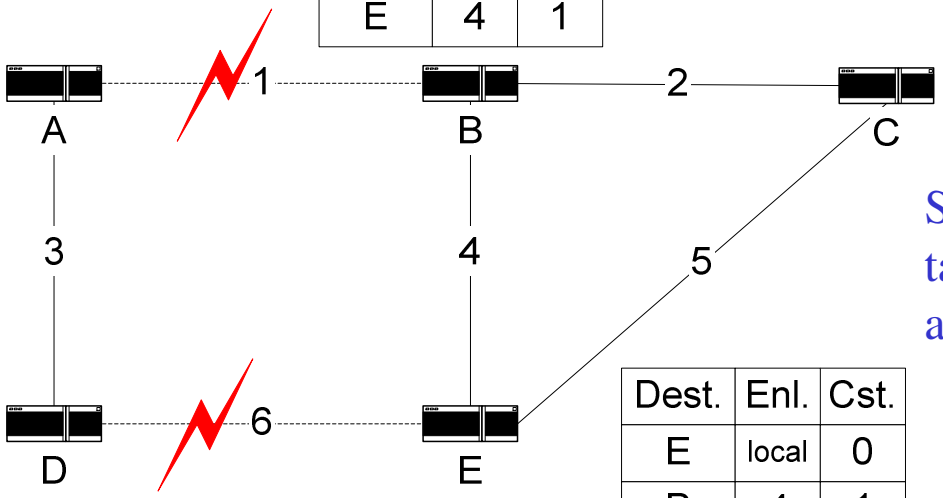
Contagem até o Infinito (2)

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



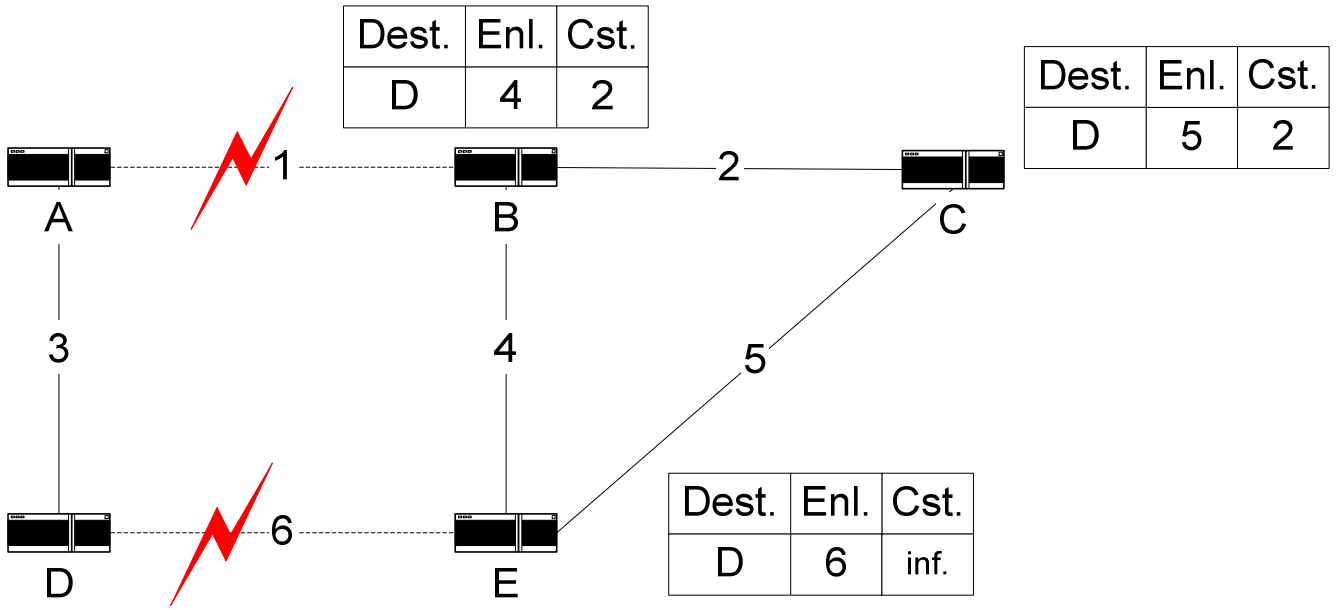
Dest.	Enl.	Cst.
D	local	0
A	3	1
B	6	2
E	6	1
C	6	2

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.

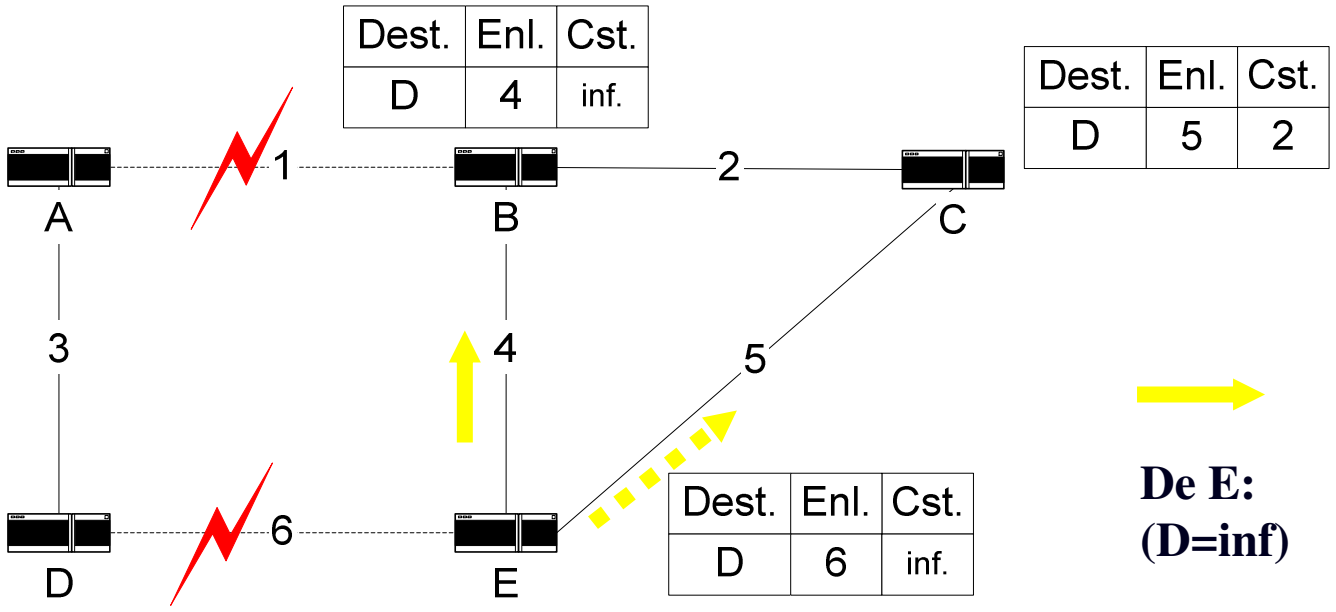
Contagem até o Infinito (2)

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



Contagem até o Infinito (2)

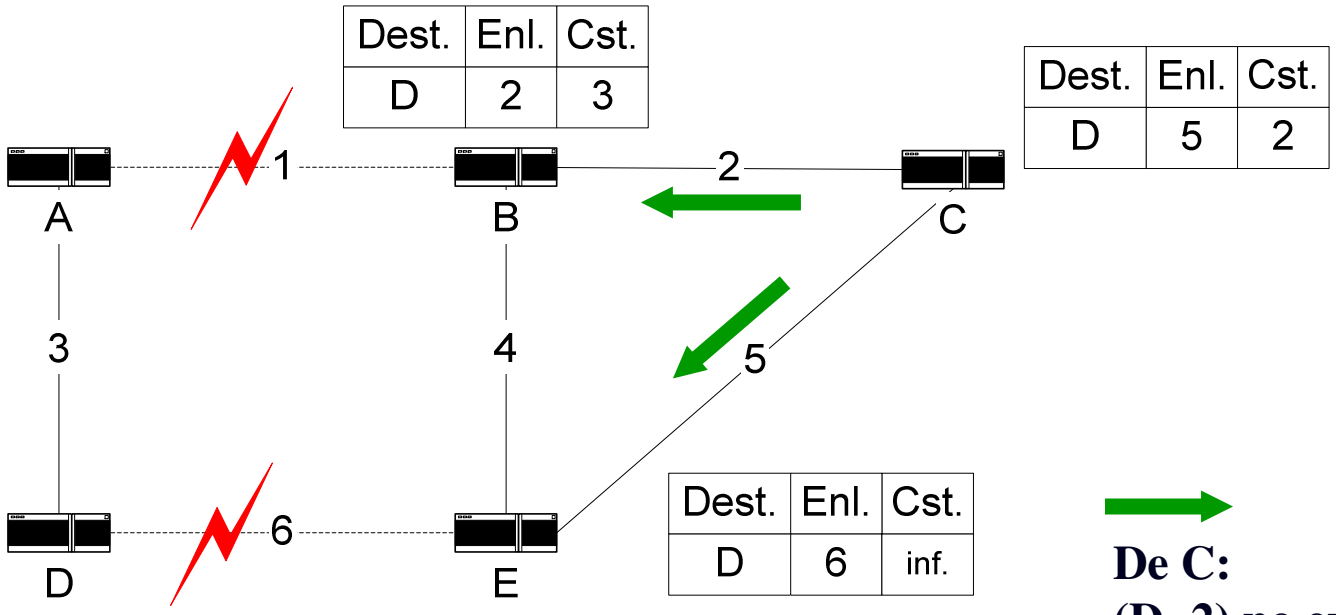
Suponha que **E** envia um vetor de distância, que chega a **B** mas não é recebido por **C** devido a um erro de transmissão.



Apenas **B** atualiza sua tabela.

Contagem até o Infinito (2)

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

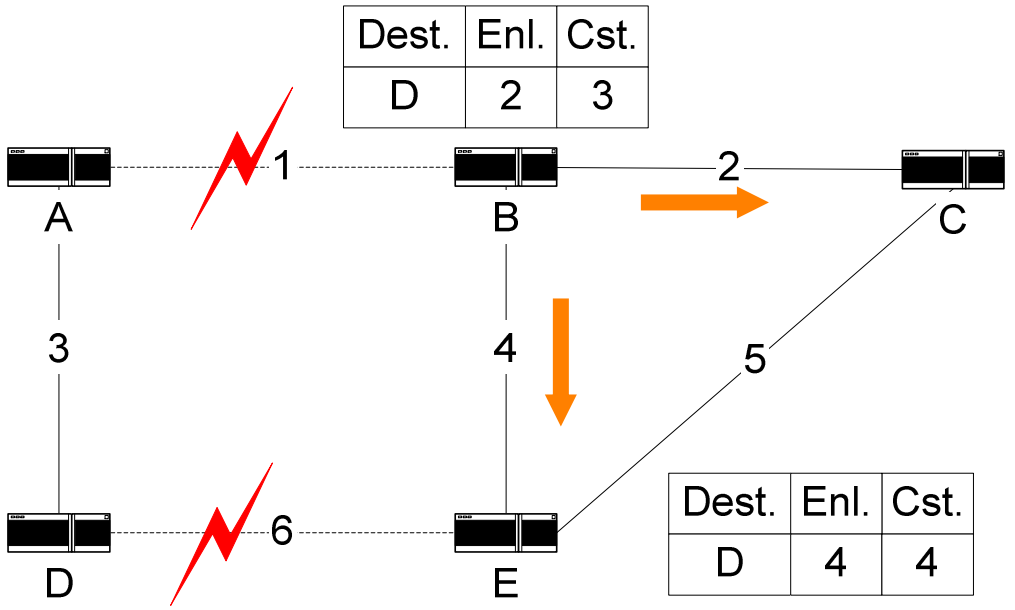


De C:
(D=2) no enlace 2
(D=inf) no enlace 5

B atualiza sua tabela.

Contagem até o Infinito (2)

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.

Temporização das Rotas

- 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
- 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

Triggered Updates

- 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*
  - 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

Algoritmo de Bellman-Ford

- Variáveis

Seja **N** o número de nós, **M** o número de enlaces

Seja **L** um vetor de enlaces de tamanho **M**, onde

L[l].m é a métrica do enlace **l**,

L[l].s o nó fonte e **L[l].d** o nó destino do enlace **l**.

Seja **D** uma tabela de tamanho **[N,N]**, onde **D[i,j]** é a distância entre os nós **i** e **j**.

Seja **H** uma tabela de tamanho **[N,N]**, onde **H[i,j]** é o enlace sobre o qual **i** roteia pacotes para **j** (**H[i,j]** é o próximo salto de **i** na direção de **j**)

Algoritmo de Bellman-Ford

Passo 1

Iniciar todos os D[i,j] para 0 se i=j, para inf. se i!=j.

Iniciar todos os H[i,j] para -1.

Passo 2

Para todo enlace l, para todo destino k:

```
i = L[l].s, j= L[l].d
d = L[l].m + D[j,k]
Se d < D[i,k]
        D[i,k] = d;
        H[i,k] = l;
```

Passo 3

Se pelo menos um D[i,k] foi modificado, repita o Passo 2. Senão, o algoritmo terminou.

Algoritmo de Bellman-Ford

- Complexidade
  - $O(M*N^2)$
- Versão distribuída
  - Cada nó calcula uma parte das tabelas de distâncias e de rotas
  - Cada nó, i, se encarrega dos
    - enlaces que partem do nó i
    - da coluna D[i,*] da tabela de distâncias
    - da coluna H[i,*] da tabela de rotas
  - A coluna D[i,*] corresponde ao vetor de distância...

Endereços no RIPv1

- Tabelas RIP
  - Endereços Internet de 32 bits
  - Podem representar uma estação, rede, ou sub-rede
    - Porém não há indicação de tipo de endereço nas mensagens
- 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

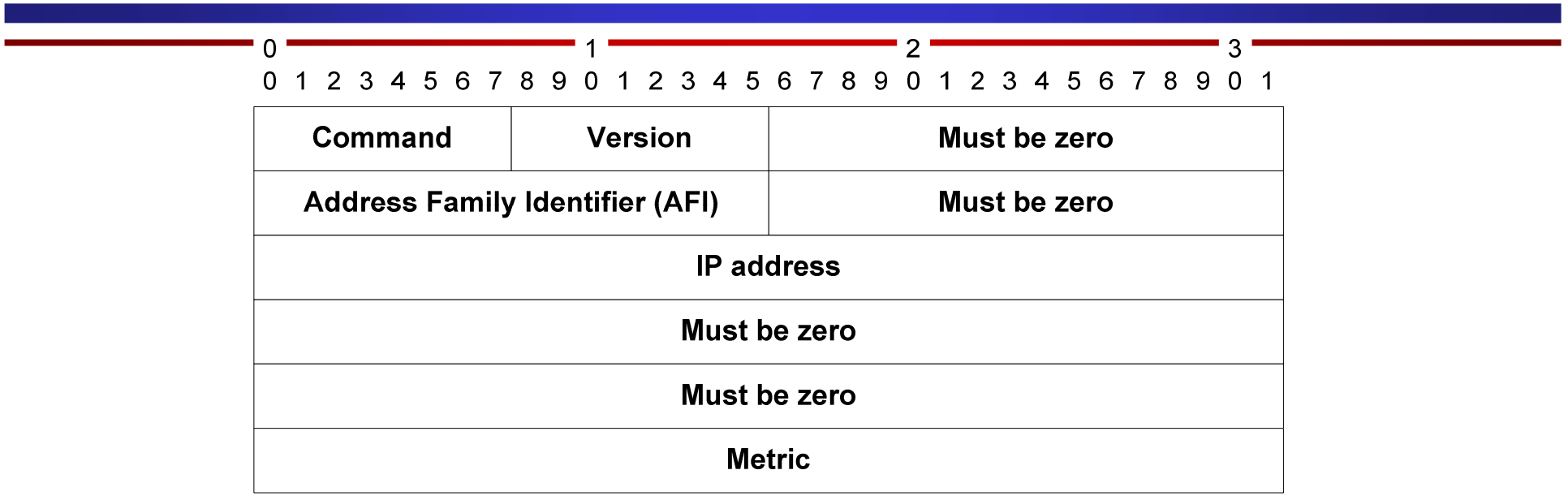
Endereços e Rotas no RIPv1

- RFC1058
  - 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

- Métrica por default
  - 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
- 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)

Ineficiência da Codificação

- Intenção inicial era suportar outros protocolos de rede
  - Mas na prática, AFI = 2 (IP)
- Métrica
  - Só varia entre 0 e 16, mas codificada em 32 bits
    - Alinhamento em palavras de 32 bits...

Processamento das Mensagens RIP

- *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

- 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

- 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*
- 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

- 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

- 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 5s 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

- 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
    - No BSD, a interface de entrada não é passada na API UDP

Geração das Respostas

- 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
- 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

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

Nós Silenciosos (RFC-1058)

- Hoje em dia, a distinção entre roteador e estação é clara, mas não era o caso quando o RIP foi projetado...
  - Anúncios passam na rede local a cada 30 s, por que não ouví-los e construir sua própria tabela?
  - Estações *multihomed*
    - poderiam escolher a melhor interface
  - Vários roteadores na rede local
    - Escolha da melhor saída
- Nós silenciosos
  - Nunca enviam respostas, mas podem enviar pedidos
  - Funcionou apenas enquanto todos os roteadores eram RIPv1
    - Hoje em dia, RIPv2, OSPF, IGRP...
    - O comportamento recomendado é utilizar um roteador *default* e o ICMP redirect

Configuração do RIP

- Configuração básica
  - Lista de interfaces, endereços e máscaras associados
  - Métrica 1 por *default* para todas as interfaces
  - 1 entrada na tabela para cada uma das sub-redes
    - com distância 1
  - Mensagem de Pedido aos vizinhos para preencher a tabela
  - Mensagens de Resposta enviadas em *broadcast*

Mas...

- Em alguns casos o DV não é difundido em ***todas*** as interfaces
  - Quando há apenas ***este*** roteador na sub-rede
    - Evita o desperdício de recursos
  - Algumas interfaces podem operar
    - com rotas fixas,
    - ou com outro protocolo,
    - e o administrador pode validar/invalidar interfaces.
- Em interfaces sem capacidade de difusão (*non-broadcast*)
  - Mensagens enviadas em ponto-a-ponto
  - Endereço dos vizinhos deve ser conhecido (configurado)

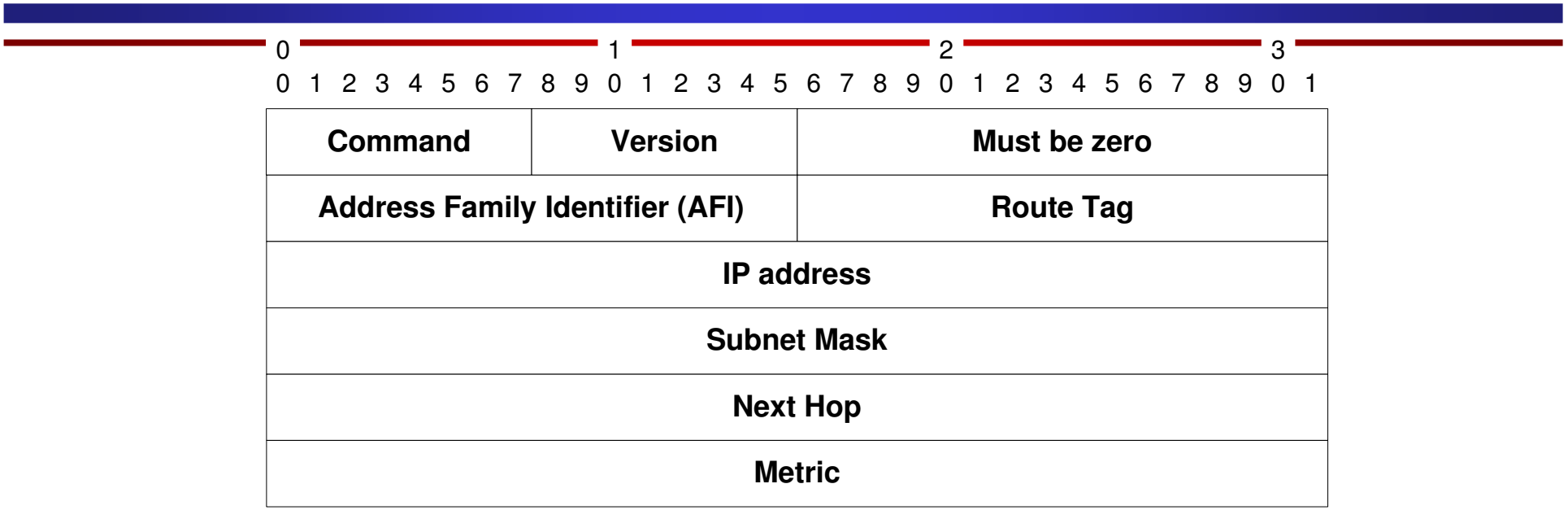
Configuração do RIP

- Configuração de métricas
  - Alterar o valor de métricas associadas a interfaces pode privilegiar o uso de uma ou outra rota
- Rotas fixas ou estáticas
  - Inseridas permanentemente na tabela (por configuração)
- Destinos incomunicáveis (máquinas a evitar)
  - São filtrados das mensagens de resposta (DV) recebidas

RIP Versão 2

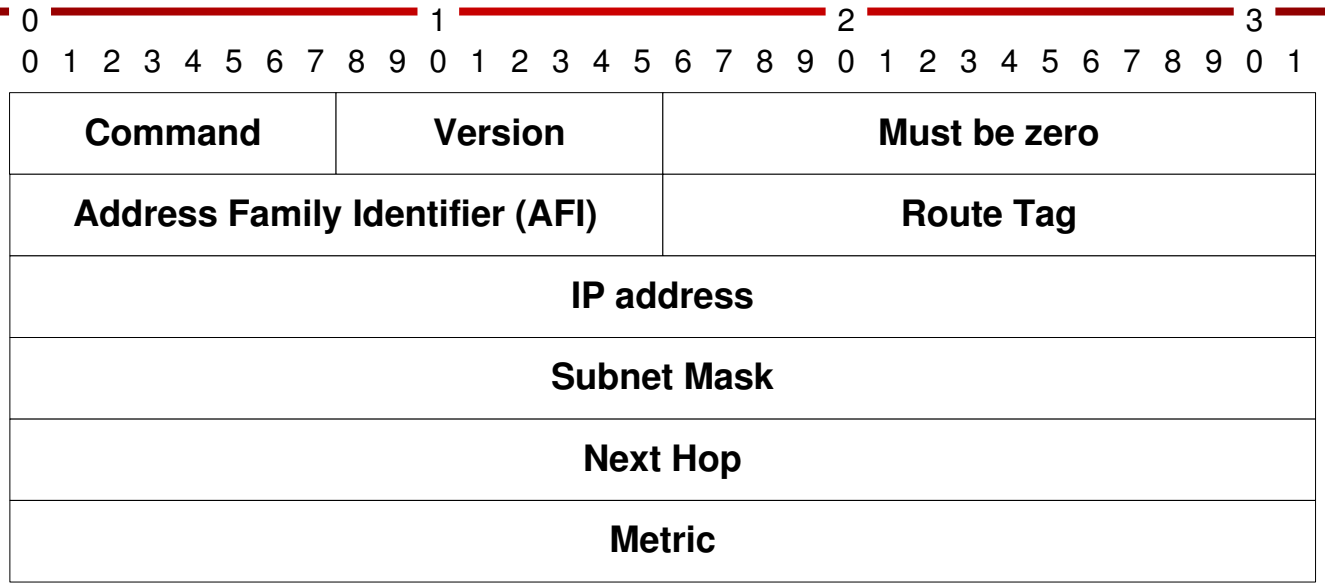
- RFC-1388 - RIP Version 2 Carrying Additional Information
  - *Updates* RFC-1058
  - *Obsoleted by* RFC-1723 (*Obsoleted by* RFC-2453)
- RFC-1389 - RIP Version 2 MIB Extension
  - Estruturas de dados para gerenciamento
- RFC-1387 – RIP Version 2 Protocol Analysis
  - *Obsoleted by* RFC-1721
  - Informational

Formato das Mensagens



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

Formato



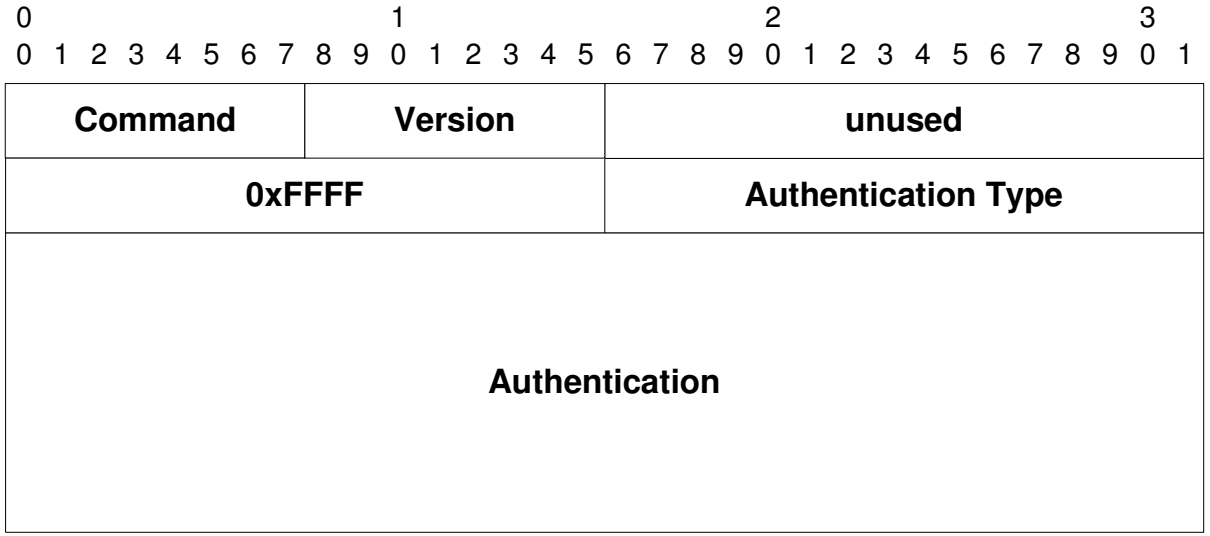
- Novos campos
  - Próximo salto (Next Hop)
    - Elimina saltos duplos na mesma sub-rede
  - Máscara (Subnet Mask)
    - Melhora o roteamento por sub-rede
  - Route Tag
    - Marca rotas externas (utilizado com BGP/EGP)

Autenticação

- 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”

Autenticação

- Definida na RFC-2453



- AFI = 0xFFFF – identifica entrada de autenticação
  - Compatibilidade – RIPv1 ignora esta entrada (AFI!=2)
- Authentication Type
- Authentication (16 bytes de dados de autenticação)

Autenticação

- Ao receber o pacote, o roteador RIPv2
  - Verifica que a primeira entrada é de autenticação e se esta comprova a “origem” do pacote
    - O administrador pode obrigar a verificação de todos os pacotes RIP
- RFC-2453
  - Define apenas o uso simples de uma senha
  - Authentication type = 2
  - Dados transportam a senha
  - Não garante nenhuma segurança...

Autenticação usando MD5

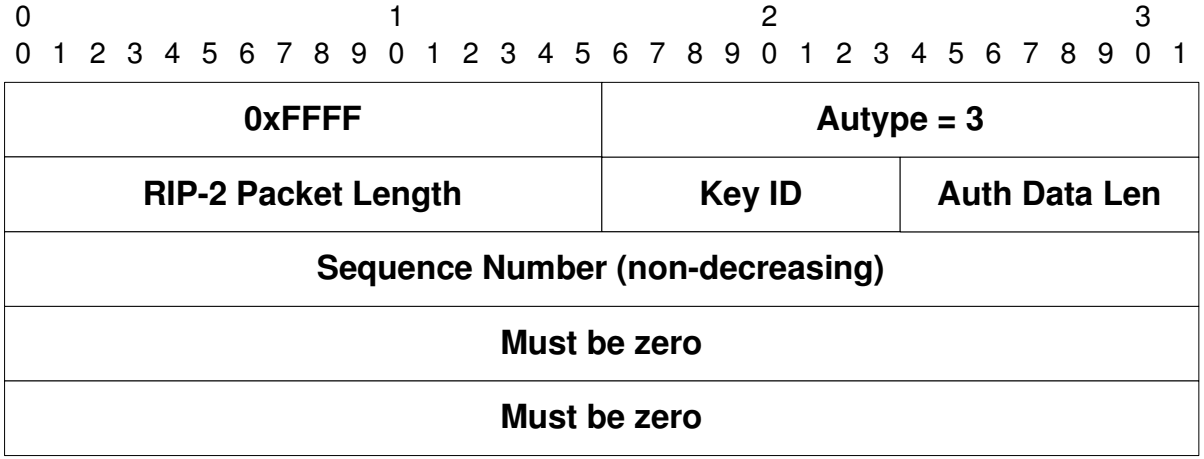
- RFC-2082 - RIP-2 MD5 Authentication
  - Evita passar segredos “em claro” na rede
  - Integridade das mensagens
  - Proteção contra ataques de repetição (*replay attacks*)
  - Distribuição segura de chaves

  - Formato do pacote RIP
    - security header + security trailer

Command (inalterado)
Security header: AFI = 0xFFFF, Autype = 3
Route entries
Security trailer: AFI = 0xFFFF, Autype = 1

Autenticação usando MD5

- *Security Header*

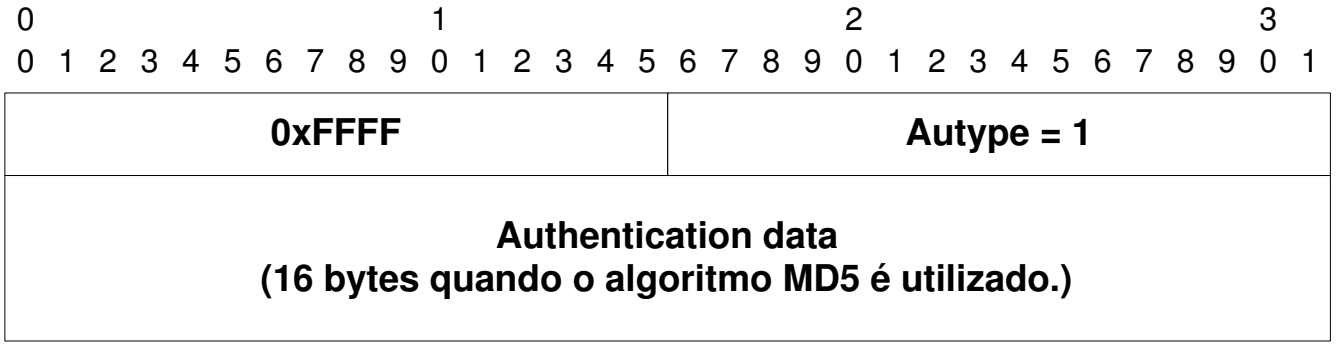


- Authentication type = 3 – "Keyed Message Digest"
- RIP-2 Packet Length
  - Tamanho normal do pacote RIP (**sem contar** o *security trailer*)
- Key-ID - Chave utilizada para proteger a mensagem
- Auth Data Len
  - Tamanho dos dados de autenticação contidos no *security trailer*

Autenticação usando MD5

- Sequence Number – inteiro de 32 bits
  - Proteção contra ataques de repetição
    - Roteador ignora qualquer mensagem cujo número de sequência não é maior que o último recebido para a chave identificada por Key-ID

- *Security trailer*



- Número variável de palavras de 32 bits

Autenticação usando MD5

- Contexto (representado pela Key-ID)
  - Chave secreta + algoritmo de autenticação
    - Configuração manual ou procedimento de troca de chaves
- Envio da mensagem usando o MD5
  - Todos os campos até os primeiros 32 bits do auth trailer são preenchidos
  - Auth header inicializada
    - Key-ID, comprimento dos dados de autenticação e da mensagem

Autenticação usando MD5

- Pseudo-mensagem
  - Auth data = valor da chave (segredo MD5)
  - + bytes de enchimento
  - + 64 bits com o comprimento real da mensagem
- Calcula-se o hash MD5 da pseudo-mensagem
  - Resultado = authentication data

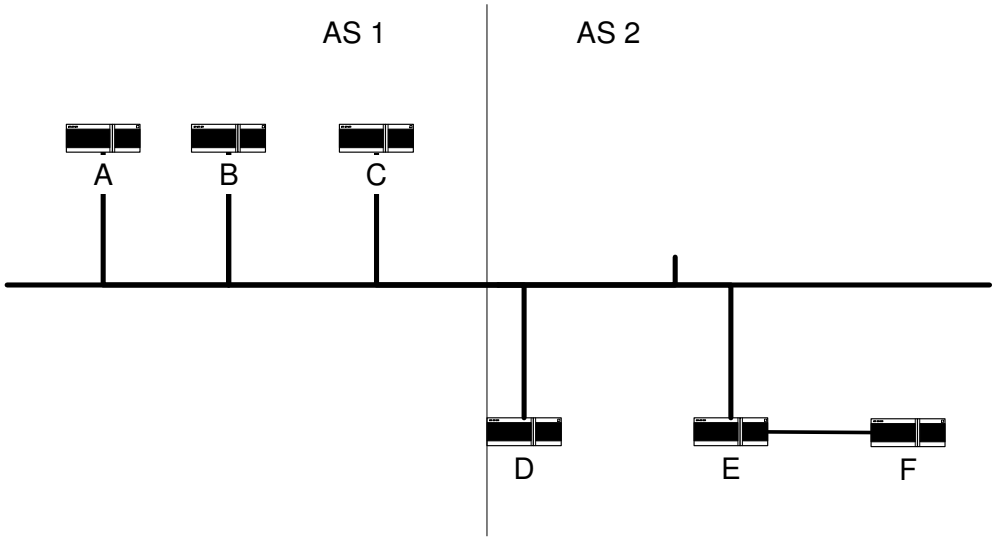
Initial message	Pseudo-message	Transmitted message
Command	Command	Command
Authentication header	Authentication header	Authentication header
Route entries	Route entries	Route entries
First 32 bits of trailer	First 32 bits of trailer	First 32 bits of trailer
	Authentication data: MD5 secret	Authentication data: result of MD5 hash
	Pad bytes (per RFC-1321)	
	32 MSB of length	
	32 LSB of length	

Autenticação usando MD5

- Na recepção, processo inverso
  - Constrói-se uma pseudo mensagem com o segredo correspondente a Key-ID e o *comprimento* da mensagem recebida
  - Compara-se o hash MD5 da pseudo-mensagem com os dados de autenticação recebidos

  - Valores iguais, dados autênticos
  - Mensagem descartada senão...

Próximo Salto



- D é o “roteador de interface” para fora do AS2
- Pacotes enviados por A para F passam por D
  - E pelo segmento Ethernet duas vezes...
- Next Hop
  - A distância para F é x, mas o próximo salto não sou eu (que originei o DV) mas o roteador E (contido no campo next hop)