

Aula 5

Professores:

Daniel R. Figueiredo
Rosa M. M. Leão

Roteamento: Algoritmos do tipo distance-vector

Conteúdo:

- Princípios, notação e equação de Bellman-Ford
- Descrição do algoritmo distance-vector
- Comportamento do algoritmo na ocorrência de falhas ou atualização nos custos dos enlaces
- Comparação entre os algoritmos do tipo link-state e distance-vector

Príncipios do algoritmo distance-vector

Príncipios do algoritmo distance-vector

→ Iterativo

Príncipios do algoritmo distance-vector

→ Iterativo

- É executado até que os nós não troquem mais informações

Príncipios do algoritmo distance-vector

→ Iterativo

- É executado até que os nós não troquem mais informações
- Termina sozinho sem que seja necessário um “*sinal*” de parada

Príncipios do algoritmo distance-vector

→ Iterativo

- É executado até que os nós não troquem mais informações
- Termina sozinho sem que seja necessário um “*sinal*” de parada

→ Assíncrono

Príncipios do algoritmo distance-vector

→ Iterativo

- É executado até que os nós não troquem mais informações
- Termina sozinho sem que seja necessário um “*sinal*” de parada

→ Assíncrono

- Nós não precisam interagir ao mesmo tempo

Príncipios do algoritmo distance-vector

→ Iterativo

- É executado até que os nós não troquem mais informações
- Termina sozinho sem que seja necessário um “*sinal*” de parada

→ Assíncrono

- Nós não precisam interagir ao mesmo tempo

→ Distribuído

Príncipios do algoritmo distance-vector

→ Iterativo

- É executado até que os nós não troquem mais informações
- Termina sozinho sem que seja necessário um “*sinal*” de parada

→ Assíncrono

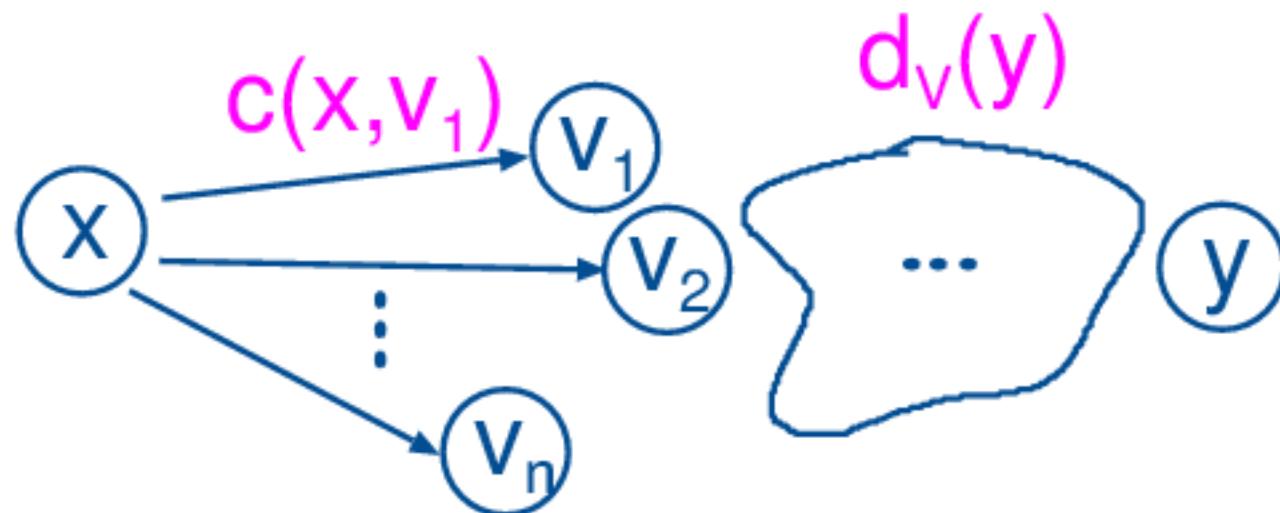
- Nós não precisam interagir ao mesmo tempo

→ Distribuído

- Cada nó se comunica somente com o vizinho

Equação de Bellman-Ford

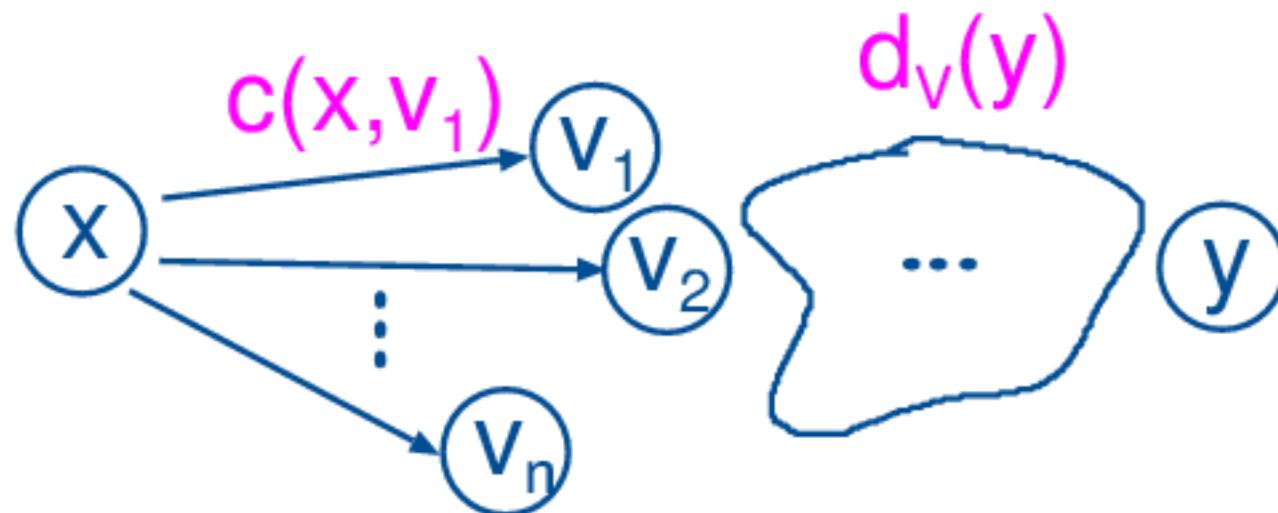
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$



Equação de Bellman-Ford

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

custo mínimo do nó x até o nó y

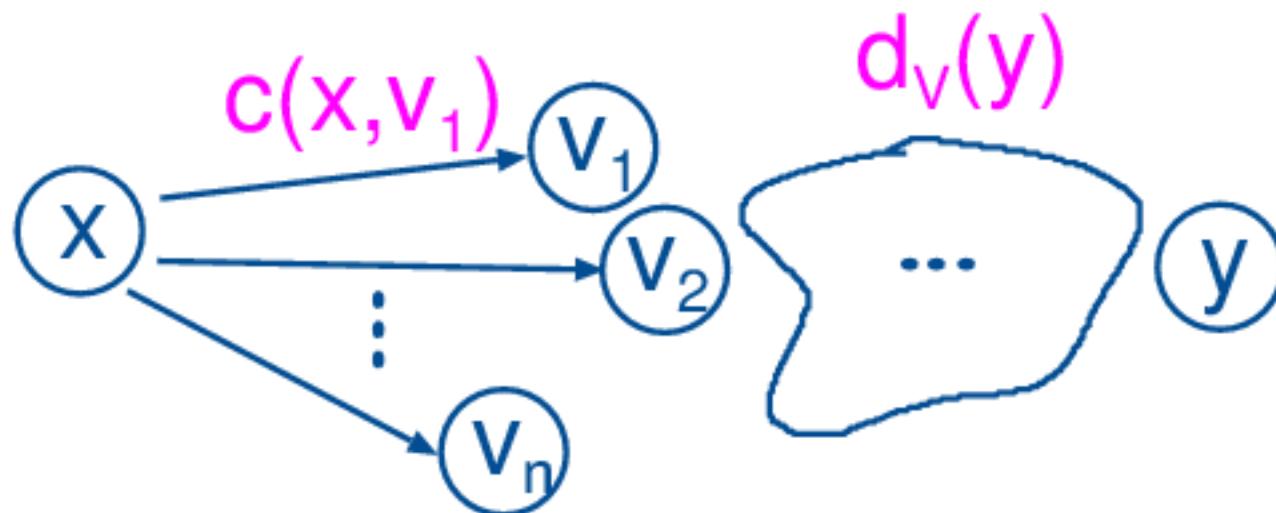


Equação de Bellman-Ford

mínimo custo entre todos os vizinhos v do nó x

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

custo mínimo do nó x até o nó y

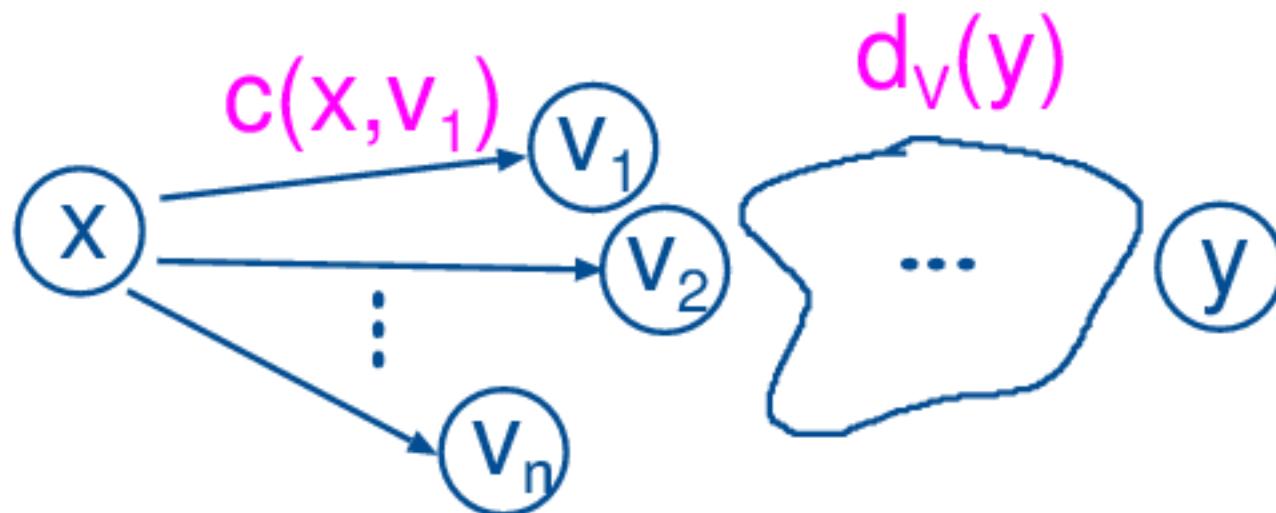


Equação de Bellman-Ford

mínimo custo entre todos os vizinhos v do nó x

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

custo mínimo do nó x até o nó y custo do enlace entre o nó x e o nó v



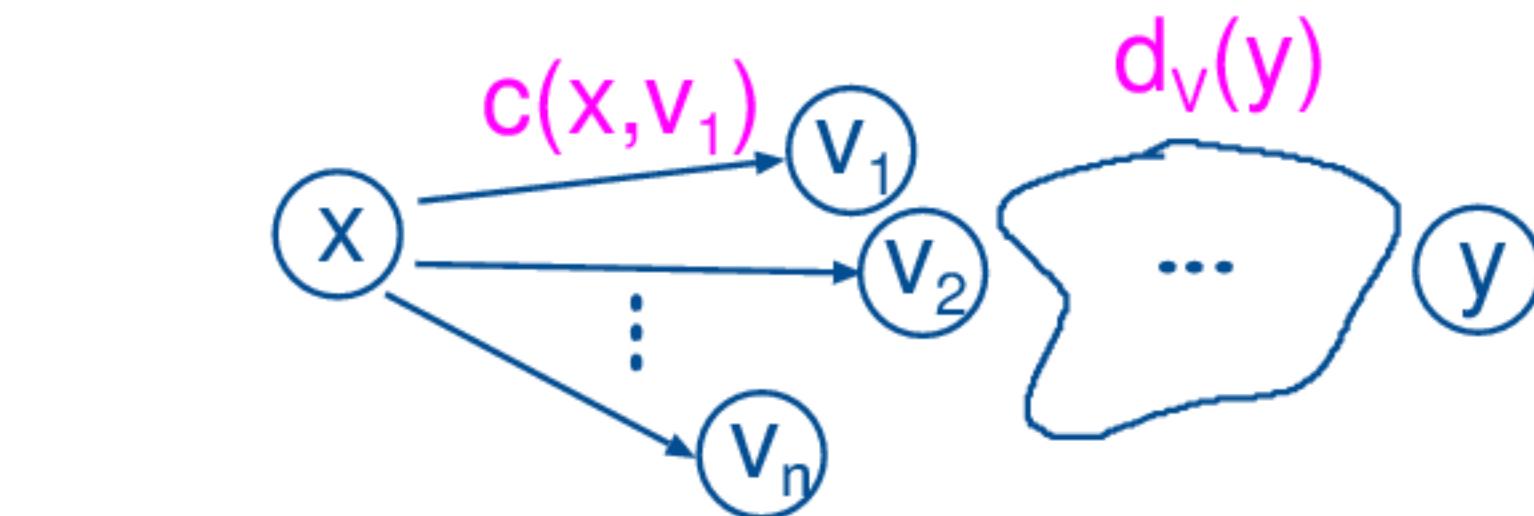
Equação de Bellman-Ford

mínimo custo entre todos os vizinhos v do nó x

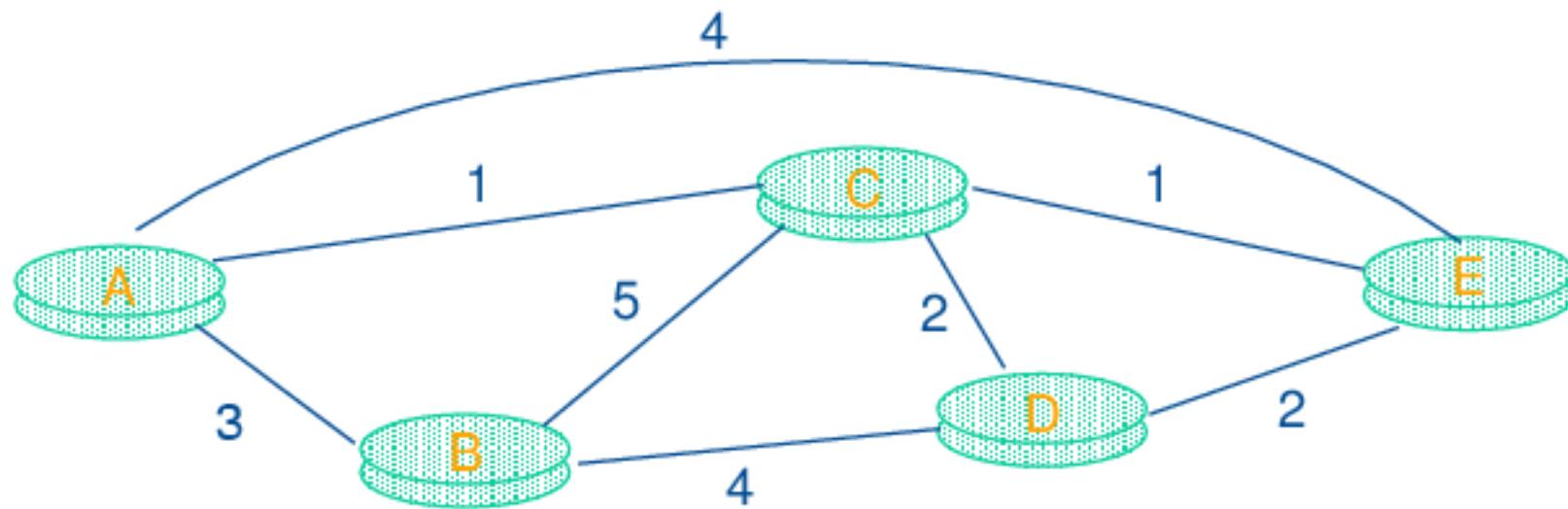
$$d_x(y) = \min_v \{ c(x, v) + d_v(y) \}$$

custo mínimo do nó v até o nó y

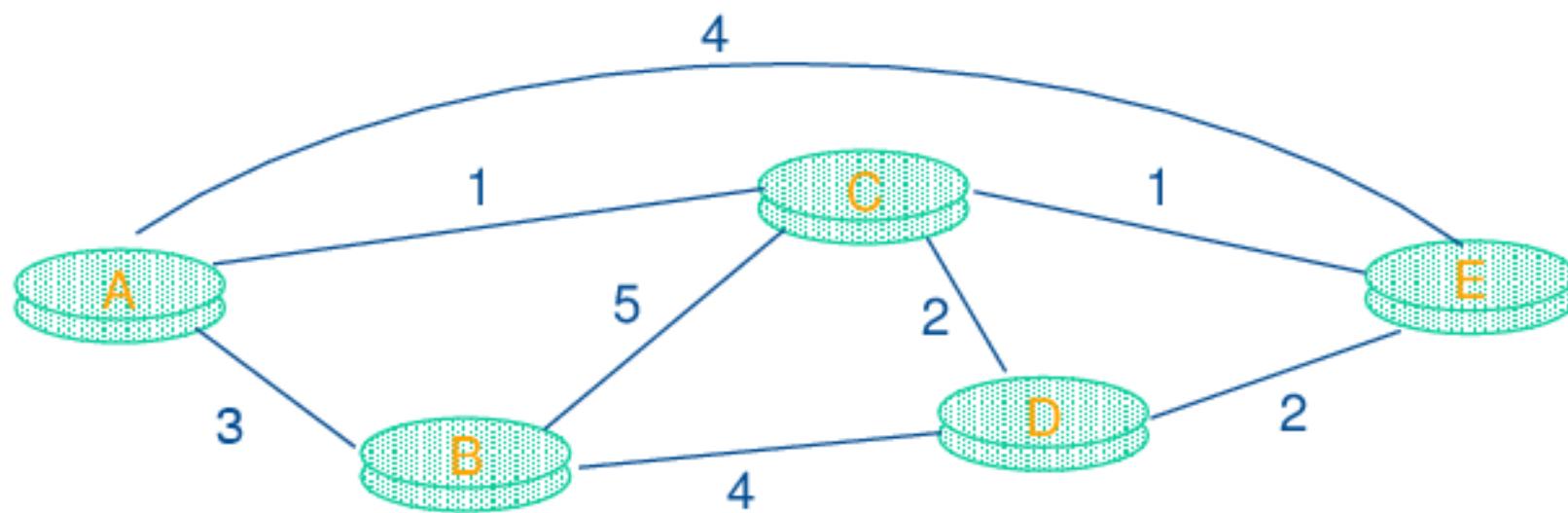
custo mínimo do nó x até o nó y custo do enlace entre o nó x e o nó v



Exemplo do algoritmo Bellman-Ford

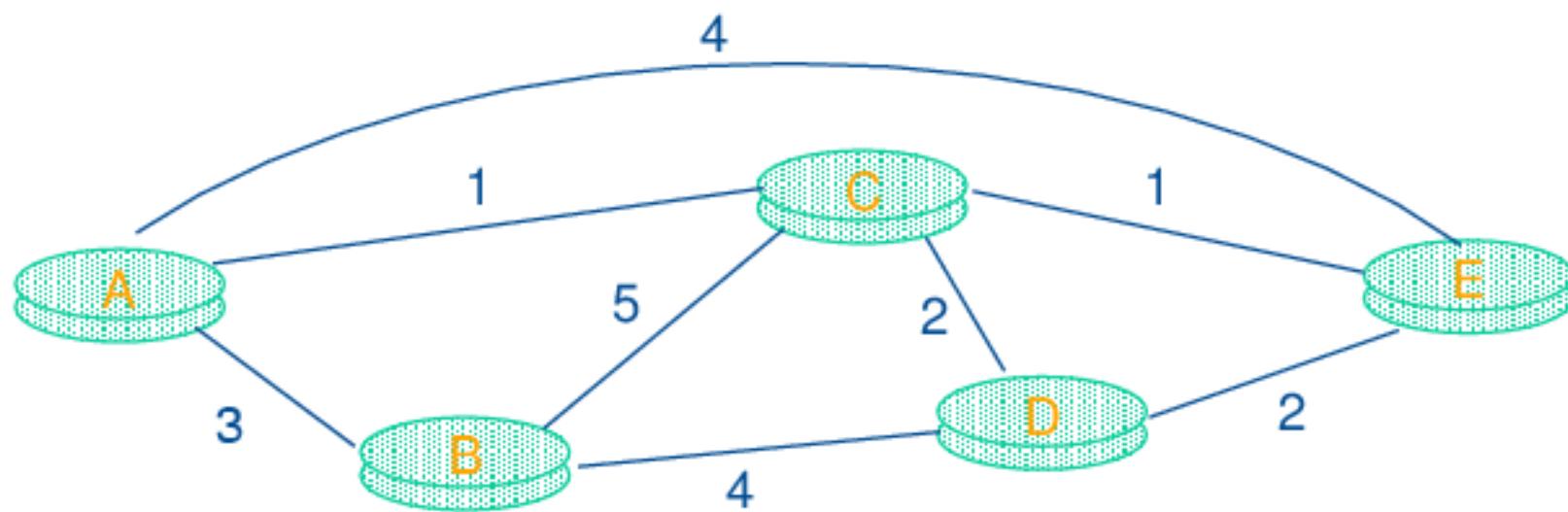


Exemplo do algoritmo Bellman-Ford



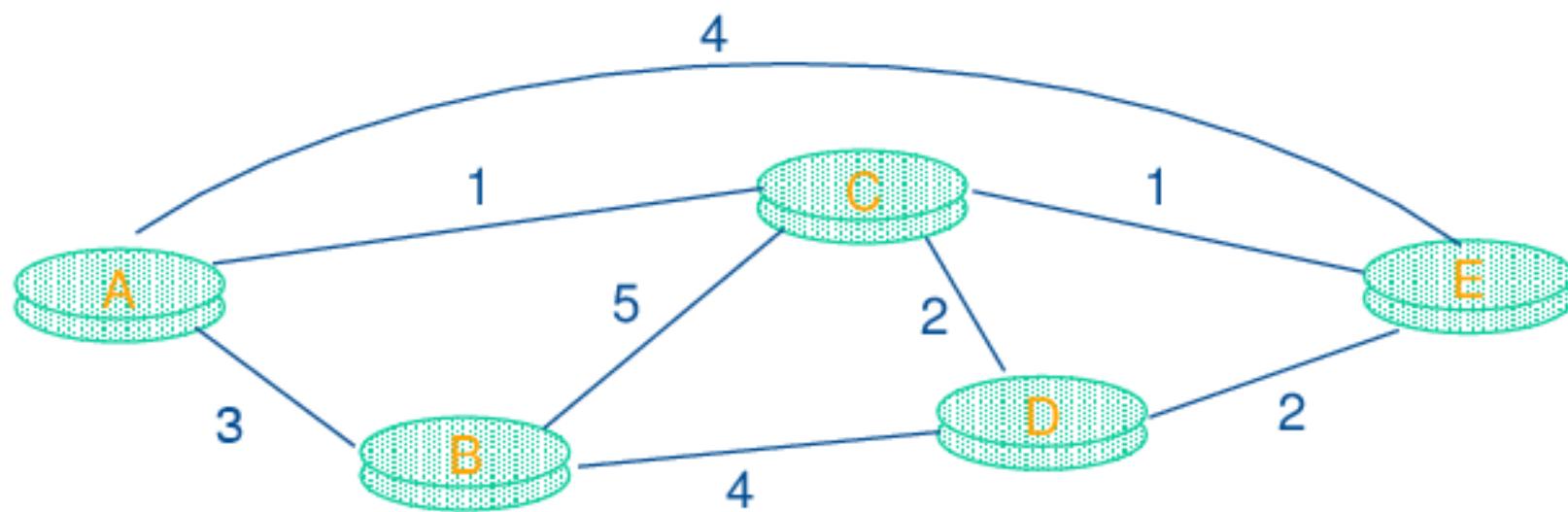
$$d_A(E) = \min_{V=\{B,C,E\}} \{c(A,v) + d_V(E)\}$$

Exemplo do algoritmo Bellman-Ford



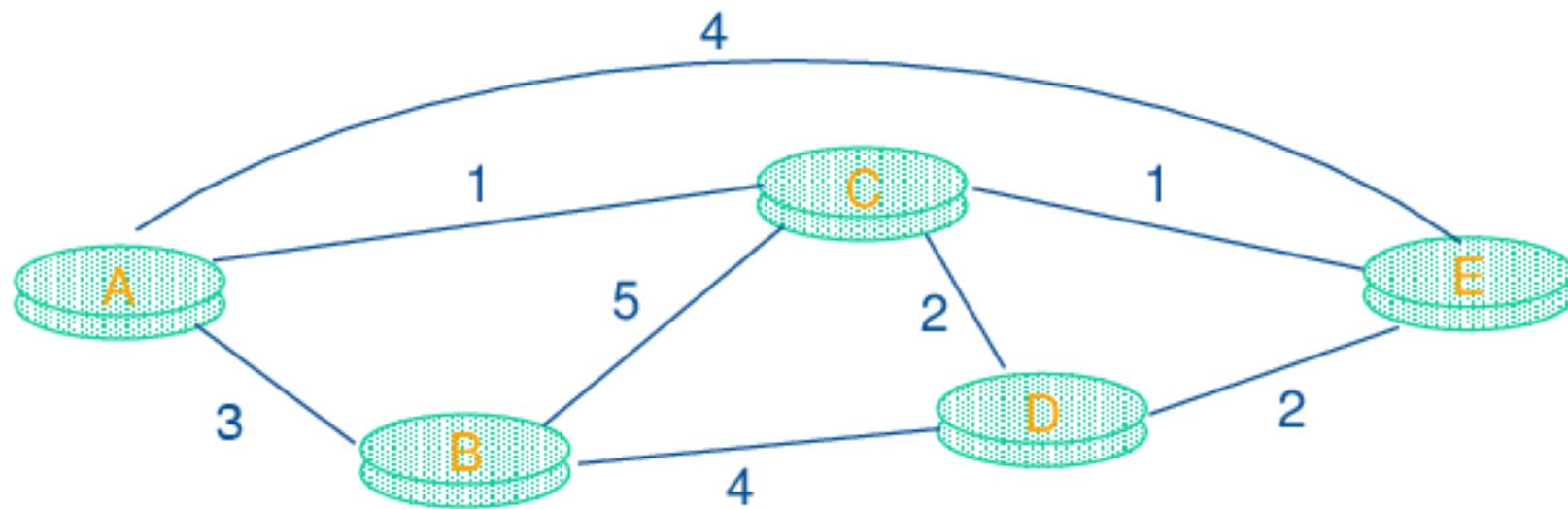
- $d_A(E) = \min_{V=\{B,C,E\}} \{c(A,v) + d_V(E)\}$
- $d_A(E) = \min \{c(A,B)+d_B(E), c(A,C)+d_C(E), c(A,E)\}$

Exemplo do algoritmo Bellman-Ford



- $d_A(E) = \min_{V=\{B,C,E\}} \{c(A,v) + d_v(E)\}$
- $d_A(E) = \min \{c(A,B)+d_B(E), c(A,C)+d_C(E), c(A,E)\}$
- $d_A(E) = \min \{3+6, 1+1, 4\} = \min \{9, 2, 4\} = 2$

Exemplo do algoritmo Bellman-Ford



- $d_A(E) = \min_{V=\{B,C,E\}} \{c(A,v) + d_v(E)\}$
- $d_A(E) = \min \{c(A,B)+d_B(E), c(A,C)+d_C(E), c(A,E)\}$
- $d_A(E) = \min \{3+6, 1+1, 4\} = \min \{9, 2, 4\} = 2$
- nó C é o próximo nó no caminho de A até E

Algoritmo distance-vector: informações armazenadas pelos nós

Algoritmo distance-vector: informações armazenadas pelos nós

→ Considere um nó x

Algoritmo distance-vector: informações armazenadas pelos nós

- Considere um nó x
- Para cada nó v vizinho de x é armazenado $c(x,v)$, custo do enlace direto que liga x a v

Algoritmo distance-vector: informações armazenadas pelos nós

- ➡ Considere um nó x
- ➡ Para cada nó v vizinho de x é armazenado $c(x,v)$, custo do enlace direto que liga x a v
- ➡ Vetor de distâncias de x , $D_x = [D_x(y) : y \text{ é um nó da rede}]$, que contém as estimativas do custo de x até todos os outros nós da rede

Algoritmo distance-vector: informações armazenadas pelos nós

- ➡ Considere um nó x
- ➡ Para cada nó v vizinho de x é armazenado $c(x,v)$, custo do enlace direto que liga x a v
- ➡ Vetor de distâncias de x , $D_x = [D_x(y) : y \text{ é um nó da rede}]$, que contém as estimativas do custo de x até todos os outros nós da rede
- ➡ Vetor de distâncias de cada um dos vizinhos de x ,
 $D_v = [D_v(y) : y \text{ é um nó da rede}]$, que contém as estimativas do custo do vizinho v até todos os outros nós da rede

Algoritmo distance-vector: idéia principal

Algoritmo distance-vector: idéia principal

- Cada nó envia periodicamente seu vetor de distâncias para todos os vizinhos

Algoritmo distance-vector: idéia principal

- Cada nó envia periodicamente seu vetor de distâncias para todos os vizinhos
- Quando um nó recebe um novo vetor de distâncias, ele atualiza o próprio vetor usando a equação:

$$D_X(y) = \min_v \{c(x,v) + D_V(y)\}$$

Algoritmo distance-vector: idéia principal

- Cada nó envia periodicamente seu vetor de distâncias para todos os vizinhos
- Quando um nó recebe um novo vetor de distâncias, ele atualiza o próprio vetor usando a equação:

$$D_X(y) = \min_v \{c(x,v) + D_V(y)\}$$

- A estimativa $D_X(y)$, sob certas condições, vai convergir para o valor real do custo mínimo $d_X(y)$, calculado pelo algoritmo de Bellman-Ford

Algoritmo distance-vector: características

Algoritmo distance-vector: características

→ Iterativo, assíncrono

Algoritmo distance-vector: características

→ Iterativo, assíncrono

- cada iteração local é causada por:
 - (i) mudança no custo de em enlace do nó
 - (ii) recebimento de um novo vetor de distâncias de um vizinho

Algoritmo distance-vector: características

→ Iterativo, assíncrono

- cada iteração local é causada por:

- (i) mudança no custo de em enlace do nó
- (ii) recebimento de um novo vetor de distâncias de um vizinho

→ Distribuído

Algoritmo distance-vector: características

→ Iterativo, assíncrono

- cada iteração local é causada por:
 - (i) mudança no custo de em enlace do nó
 - (ii) recebimento de um novo vetor de distâncias de um vizinho

→ Distribuído

- um nó **x** notifica seus vizinhos somente quando seu vetor de distâncias é atualizado

Algoritmo distance-vector: características

→ Iterativo, assíncrono

- cada iteração local é causada por:
 - (i) mudança no custo de em enlace do nó
 - (ii) recebimento de um novo vetor de distâncias de um vizinho

→ Distribuído

- um nó **x** notifica seus vizinhos somente quando seu vetor de distâncias é atualizado
- os vizinhos de **x** irão notificar os seus vizinhos, se necessário

Algoritmo distance-vector: resumo

- 1 Inicialização
- 2 faça para todos os nós y em N :
 - 3 $D_x(y) = c(x,y)$, se y é vizinho de x
 - 4 $D_x(y) = \text{infinito}$, caso contrário
 - 5 para cada vizinho v
 - 6 envie o vetor de distâncias D_x calculado por x
 - 7
- 8 Loop
- 9 esperar (até que haja mudança no custo de um enlace de x ou até que seja recebido um vetor de distâncias de um vizinho v)
- 10 faça para cada y em N :
 - 11 $D_x(y) = \min_v \{c(x,v) + D_v(y)\}$
 - 12 Se $D_x(y)$ mudou para algum destino y
 - 13 envie o novo vetor de distâncias D_x para todos os vizinhos
 - 13 para sempre

Algoritmo distance-vector: resumo

- 1 Inicialização
- 2 faça para todos os nós y em N : conjunto de nós da rede
- 3 $D_x(y) = c(x,y)$, se y é vizinho de x
- 4 $D_x(y) = \text{infinito}$, caso contrário
- 5 para cada vizinho v
- 6 envie o vetor de distâncias D_x calculado por x
- 7
- 8 Loop
- 9 esperar (até que haja mudança no custo de um enlace de x ou
até que seja recebido um vetor de distâncias de um vizinho v)
- 10 faça para cada y em N :
- 11 $D_x(y) = \min_v \{c(x,v) + D_v(y)\}$
- 12 Se $D_x(y)$ mudou para algum destino y
- 13 envie o novo vetor de distâncias D_x para todos os vizinhos
- 13 para sempre

Algoritmo distance-vector: resumo

- 1 Inicialização
- 2 faça para todos os nós y em N : conjunto de nós da rede
- 3 $D_x(y) = c(x,y)$, se y é vizinho de x custo do enlace que liga x a y
- 4 $D_x(y) = \text{infinito}$, caso contrário
- 5 para cada vizinho v
- 6 envie o vetor de distâncias D_x calculado por x
- 7
- 8 Loop
- 9 esperar (até que haja mudança no custo de um enlace de x ou
até que seja recebido um vetor de distâncias de um vizinho v)
- 10 faça para cada y em N :
- 11 $D_x(y) = \min_v \{c(x,v) + D_v(y)\}$
- 12 Se $D_x(y)$ mudou para algum destino y
- 13 envie o novo vetor de distâncias D_x para todos os vizinhos
- 13 para sempre

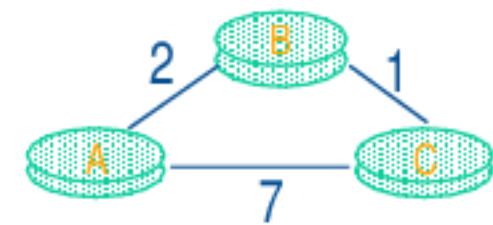
Algoritmo distance-vector: resumo

- 1 Inicialização
- 2 faça para todos os nós y em N : conjunto de nós da rede
- 3 $D_x(y) = c(x,y)$, se y é vizinho de x
- 4 $D_x(y) = \text{infinito}$, caso contrário custo do enlace que liga x a y
- 5 para cada vizinho v
- 6 envie o vetor de distâncias D_x calculado por x
- 7
- 8 Loop
- 9 esperar (até que haja mudança no custo de um enlace de x ou
até que seja recebido um vetor de distâncias de um vizinho v)
- 10 faça para cada y em N :
- 11 $D_x(y) = \min_v \{c(x,y) + D_v(y)\}$
- 12 Se $D_x(y)$ mudou para algum destino y
- 13 envie o novo vetor de distâncias D_x para todos os vizinhos
- 13 para sempre

Algoritmo distance-vector: resumo

- 1 Inicialização
- 2 faça para todos os nós y em N : conjunto de nós da rede
- 3 $D_x(y) = c(x,y)$, se y é vizinho de x
- 4 $D_x(y) = \text{infinito}$, caso contrário custo do enlace que liga x a y
- 5 para cada vizinho v
- 6 envie o vetor de distâncias D_x calculado por x
- 7
- 8 Loop
- 9 esperar (até que haja mudança no custo de um enlace de x ou
até que seja recebido um vetor de distâncias de um vizinho v)
- 10 faça para cada y em N :
- 11 $D_x(y) = \min_v \{c(x,y) + D_v(y)\}$
- 12 Se $D_x(y)$ mudou para algum destino y
- 13 envie o novo vetor de distâncias D_x para todos os vizinhos
- 13 para sempre

Algoritmo distance-vector: exemplo



Algoritmo distance-vector: exemplo

tabela do nó A

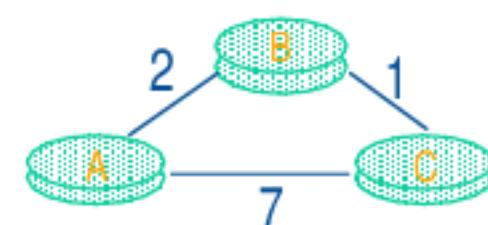
	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó C

	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0



Algoritmo distance-vector: exemplo

tabela do nó A

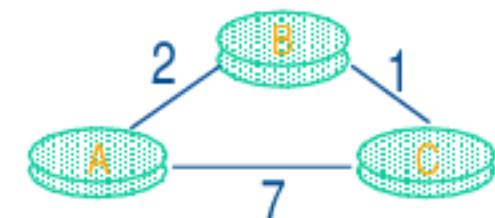
	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó C

	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0



Algoritmo distance-vector: exemplo

tabela do nó A

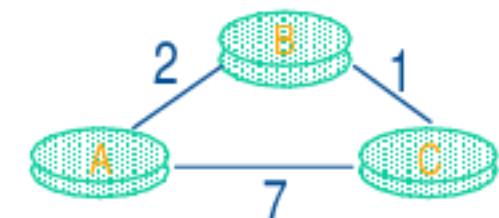
	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó C

	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0



Algoritmo distance-vector: exemplo

tabela do nó A

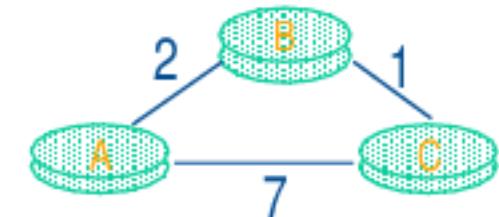
	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó C

	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0



Algoritmo distance-vector: exemplo

tabela do nó A

	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó A

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	7	1	0

tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó B

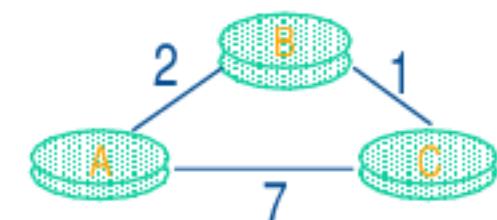
	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	7	1	0

tabela do nó C

	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0

tabela do nó C

	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	3	1	0



Algoritmo distance-vector: exemplo

tabela do nó A

	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó A

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	7	1	0

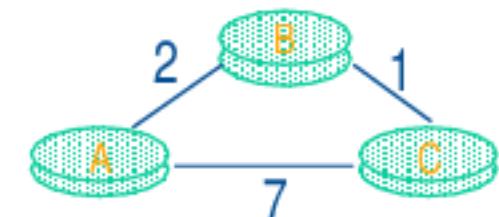
tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó C

	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0

$$\begin{aligned}
 D_A(B) &= \min\{c(A,B) + D_B(B), \\
 &\quad c(A,C) + D_C(B)\} \\
 &= \min\{2+0, 7+1\} = 2
 \end{aligned}$$



Algoritmo distance-vector: exemplo

tabela do nó A

	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó A

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	7	1	0

tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó C

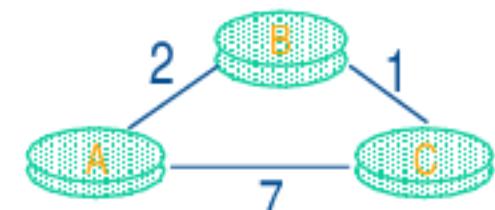
	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0

tabela do nó B

	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	7	1	0

tabela do nó C

	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	3	1	0



Algoritmo distance-vector: exemplo

tabela do nó A

	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó A

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	7	1	0

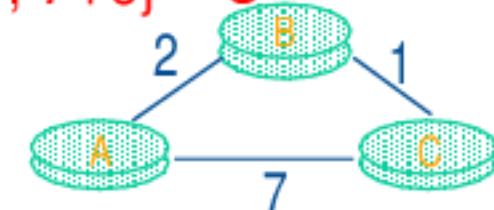
tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó C

	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0

$$\begin{aligned}
 DA(C) &= \min\{c(A,B) + DB(C), \\
 &\quad c(A,C) + DC(C)\} \\
 &= \min\{2+1, 7+0\} = 3
 \end{aligned}$$



Algoritmo distance-vector: exemplo

tabela do nó A

	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó A

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	7	1	0

tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó C

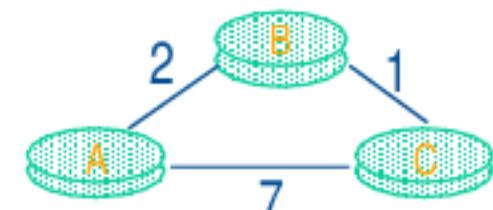
	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0

tabela do nó B

	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	7	1	0

tabela do nó C

	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	3	1	0



Algoritmo distance-vector: exemplo

tabela do nó A

	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó A

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	7	1	0

tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó C

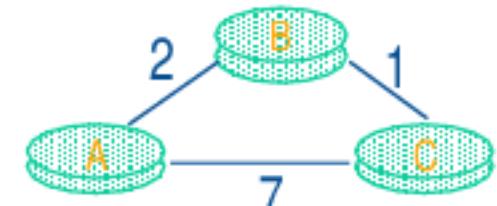
	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0

tabela do nó B

	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	7	1	0

tabela do nó C

	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	3	1	0



Algoritmo distance-vector: exemplo

tabela do nó A

	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó A

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	7	1	0

tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó C

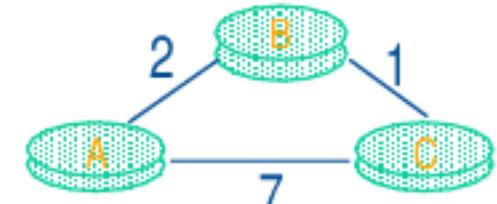
	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0

tabela do nó B

	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	7	1	0

tabela do nó C

	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	3	1	0



Algoritmo distance-vector: exemplo

tabela do nó A

	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó A

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	7	1	0

tabela do nó A

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	3	1	0

tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó B

	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	7	1	0

tabela do nó B

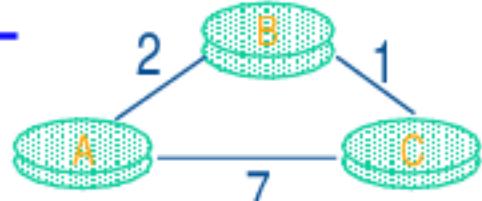
	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	3	1	0

tabela do nó C

	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0

	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	3	1	0

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	3	1	0



Algoritmo distance-vector: exemplo

tabela do nó A

	custo até		
	A	B	C
DA	0	2	7
DB	∞	∞	∞
DC	∞	∞	∞

tabela do nó A

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	7	1	0

tabela do nó A

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	3	1	0

tabela do nó B

	custo até		
	A	B	C
DA	∞	∞	∞
DB	2	0	1
DC	∞	∞	∞

tabela do nó B

	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	7	1	0

tabela do nó B

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	3	1	0

tabela do nó C

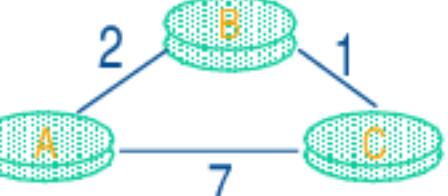
	custo até		
	A	B	C
DA	∞	∞	∞
DB	∞	∞	∞
DC	7	1	0

tabela do nó C

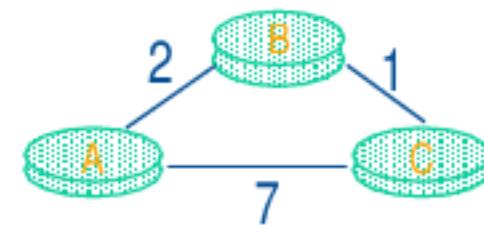
	custo até		
	A	B	C
DA	0	2	7
DB	2	0	1
DC	3	1	0

tabela do nó C

	custo até		
	A	B	C
DA	0	2	3
DB	2	0	1
DC	3	1	0



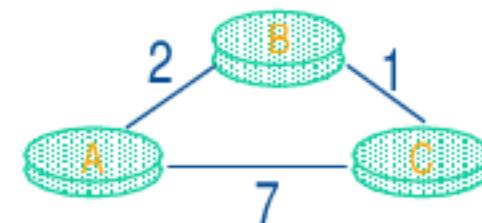
Algoritmo distance-vector: tabela de roteamento



Algoritmo distance-vector: tabela de roteamento

→ Tabela de roteamento do nó A

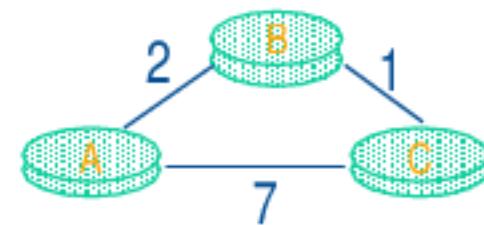
Destino	Enlace de saída	Custo
B	(A,B)	2
C	(A,B)	3



Algoritmo distance-vector: tabela de roteamento

→ Tabela de roteamento do nó A

Destino	Enlace de saída	Custo
B	(A,B)	2
C	(A,B)	3

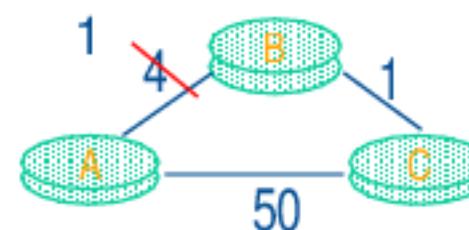


→ Tabela de roteamento do nó B

Destino	Enlace de saída	Custo
A	(B,A)	2
C	(B,C)	1

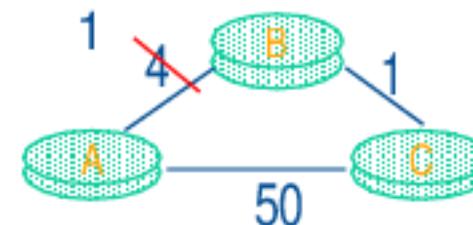
Algoritmo distance-vector: mudança nos custos dos enlaces

11



Algoritmo distance-vector: mudança nos custos dos enlaces

- = notícias boas são percebidas rapidamente



Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias boas são percebidas rapidamente

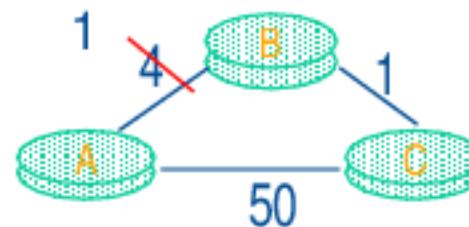


tabela do nó B

	custo até
via	A
A	4
C	6

tabela do nó C

	custo até
via	A
A	50
B	5

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias boas são percebidas rapidamente

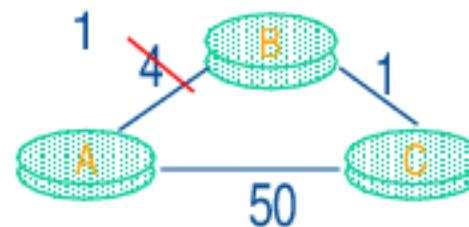


tabela do nó B

		custo até
		A
via	A	4
	C	6

tabela do nó C

		custo até
		A
via	A	50
	B	5

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias boas são percebidas rapidamente

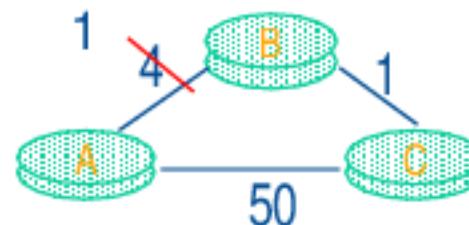


tabela do nó B

	custo até
	A
via	A
C	6

tabela do nó C

	custo até
	A
via	A
B	5

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias boas são percebidas rapidamente

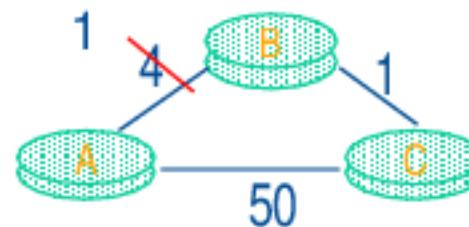


tabela do nó B

	custo até
via	A
A	4
C	6

tabela do nó C

	custo até
via	A
A	50
B	5

t_0

$c(A,B)=1$

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

notícias boas são percebidas rapidamente

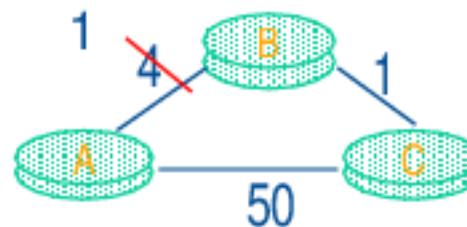


tabela do nó B

	custo até
via	A
A	4
C	6

tabela do nó B

	custo até
via	A
A	1
C	6

tabela do nó C

	custo até
via	A
A	50
B	5

tabela do nó C

	custo até
via	A
A	50
B	5



Algoritmo distance-vector: mudança nos custos dos enlaces

notícias boas são percebidas rapidamente

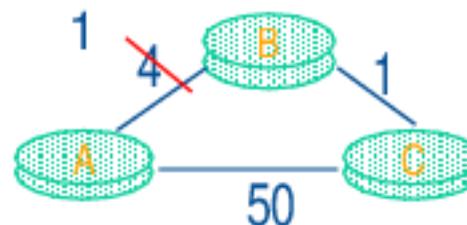


tabela do nó B

	custo até
via	A
A	4
C	6

tabela do nó B

	custo até
via	A
A	1
C	6

tabela do nó C

	custo até
via	A
B	5

tabela do nó C

	custo até
via	A
B	5

t_0
 $c(A,B)=1$

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

notícias boas são percebidas rapidamente

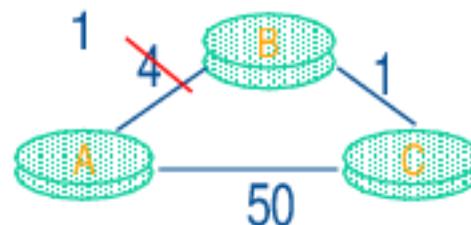


tabela do nó B

	custo até
via	A
A	4
C	6

tabela do nó B

	custo até
via	A
A	1
C	6

tabela do nó C

	custo até
via	A
B	5

tabela do nó C

	custo até
via	A
B	5

t_0
 $c(A,B)=1$

t_1
 $D_B(A)=1$

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

notícias boas são percebidas rapidamente

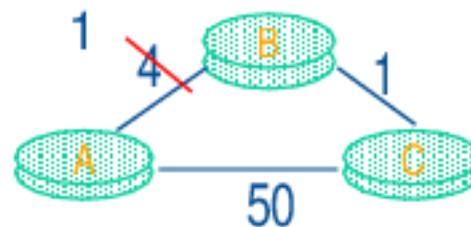


tabela do nó B

	custo até
via	A
A	4
C	6

tabela do nó B

	custo até
via	A
A	1
C	6

tabela do nó C

	custo até
via	A
B	5

tabela do nó C

	custo até
via	A
B	5

t_0
 $c(A,B)=1$

t_1
 $D_B(A)=1$

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

notícias boas são percebidas rapidamente

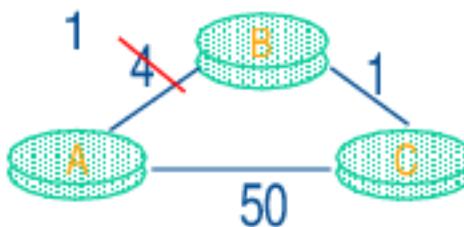


tabela do nó B

	custo até
via	A
A	4
C	6

tabela do nó B

	custo até
via	A
A	1
C	6

tabela do nó B

	custo até
via	A
A	1
C	6

tabela do nó C

	custo até
via	A
A	50
B	5

tabela do nó C

	custo até
via	A
A	50
B	5

tabela do nó C

	custo até
via	A
A	50
B	2

TEMPO

t_0
 $c(A,B)=1$

t_1
 $D_B(A)=1$

Algoritmo distance-vector: mudança nos custos dos enlaces

notícias boas são percebidas rapidamente

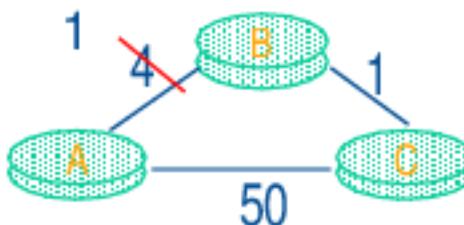


tabela do nó B

	custo até
via	A
A	4
C	6

tabela do nó B

	custo até
via	A
A	1
C	6

tabela do nó B

	custo até
via	A
A	1
C	6

tabela do nó C

	custo até
via	A
A	50
B	5

tabela do nó C

	custo até
via	A
A	50
B	5

tabela do nó C

	custo até
via	A
A	50
B	2

TEMPO

t_0
 $c(A,B)=1$

t_1
 $D_B(A)=1$

Algoritmo distance-vector: mudança nos custos dos enlaces

notícias boas são percebidas rapidamente

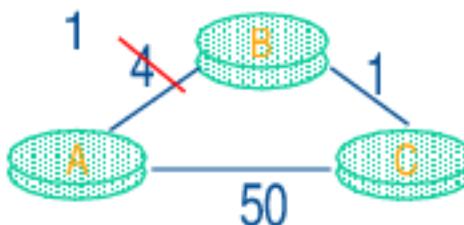


tabela do nó B

	custo até
via	A
A	4
C	6

tabela do nó B

	custo até
via	A
A	1
C	6

tabela do nó B

	custo até
via	A
A	1
C	6

tabela do nó C

	custo até
via	A
A	50
B	5

tabela do nó C

	custo até
via	A
A	50
B	5

tabela do nó C

	custo até
via	A
A	50
B	2

t_0
 $c(A,B)=1$

t_1
 $D_B(A)=1$

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

notícias boas são percebidas rapidamente

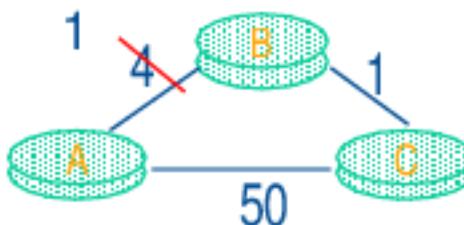


tabela do nó B

	custo até
via	A
A	4
C	6

tabela do nó B

	custo até
via	A
A	1
C	6

tabela do nó B

	custo até
via	A
A	1
C	6

tabela do nó C

	custo até
via	A
A	50
B	5

tabela do nó C

	custo até
via	A
A	50
B	5

tabela do nó C

	custo até
via	A
A	50
B	2

t_0
 $c(A,B)=1$

t_1
 $D_B(A)=1$

t_2
 $D_C(A)=2$

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

notícias boas são percebidas rapidamente

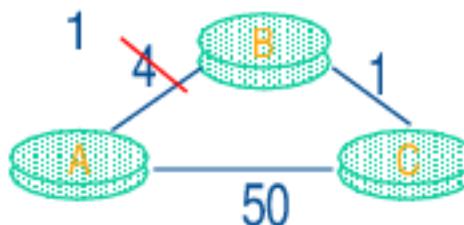


tabela do nó B

	custo até
A	A
via	C
	4
	6

tabela do nó B

	custo até
A	A
via	C
	1
	6

tabela do nó B

	custo até
A	A
via	C
	1
	6

tabela do nó B

	custo até
A	A
via	C
	1
	3

tabela do nó C

	custo até
A	A
via	B
	50
	5

tabela do nó C

	custo até
A	A
via	B
	50
	5

tabela do nó C

	custo até
A	A
via	B
	50
	2

tabela do nó C

	custo até
A	A
via	B
	50
	2

t_0
 $c(A,B)=1$

t_1
 $D_B(A)=1$

t_2
 $D_C(A)=2$

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

notícias boas são percebidas rapidamente

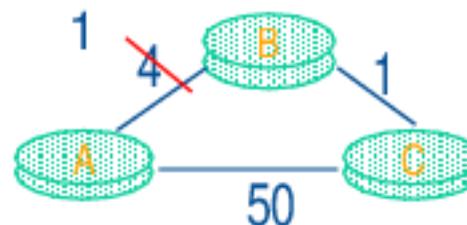


tabela do nó B

	custo até
A	A
via	C
	4
	6

tabela do nó B

	custo até
A	A
via	C
	1
	6

tabela do nó B

	custo até
A	A
via	C
	1
	6

tabela do nó B

	custo até
A	A
via	C
	1
	3

tabela do nó C

	custo até
A	A
via	B
	50
	5

tabela do nó C

	custo até
A	A
via	B
	50
	5

tabela do nó C

	custo até
A	A
via	B
	50
	2

tabela do nó C

	custo até
A	A
via	B
	50
	2

t_0
 $c(A,B)=1$

t_1
 $D_B(A)=1$

t_2
 $D_C(A)=2$

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

notícias boas são percebidas rapidamente

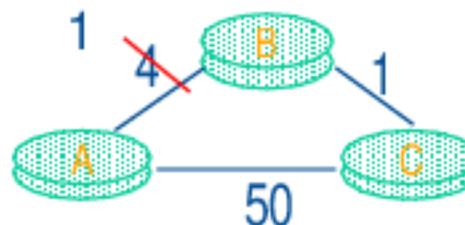


tabela do nó B

	custo até
A	A
via	C
	4
	6

tabela do nó B

	custo até
A	A
via	C
	1
	6

tabela do nó B

	custo até
A	A
via	C
	1
	6

tabela do nó B

	custo até
A	A
via	C
	1
	3

tabela do nó C

	custo até
A	A
via	B
	50
	5

tabela do nó C

	custo até
A	A
via	B
	50
	5

tabela do nó C

	custo até
A	A
via	B
	50
	2

tabela do nó C

	custo até
A	A
via	B
	50
	2

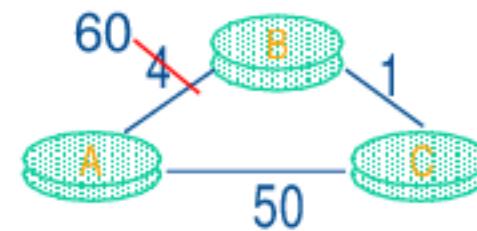
t_0
 $c(A,B)=1$

t_1
 $D_B(A)=1$

t_2
 $D_C(A)=2$

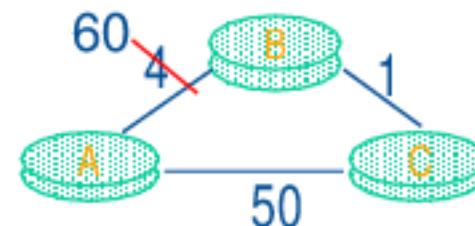
TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces



Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar



Algoritmo distance-vector: mudança nos custos dos enlaces

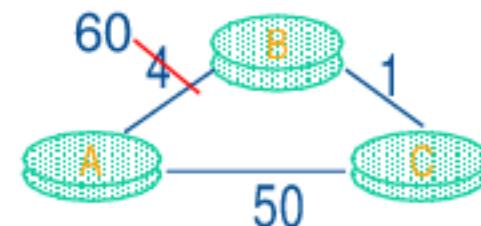
- notícias ruins demoram a se propagar

tabela do nó B

	custo até
	A
via A	4
via C	6

tabela do nó C

	custo até
	A
via A	50
via B	5



TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

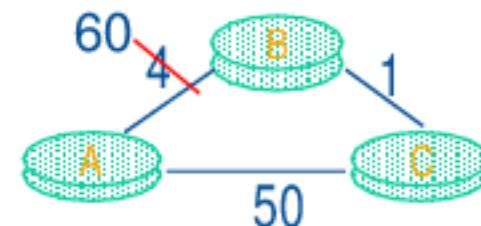
- notícias ruins demoram a se propagar

tabela do nó B

	custo até
	A
via A	4
via C	6

tabela do nó C

	custo até
	A
via A	50
via B	5



TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

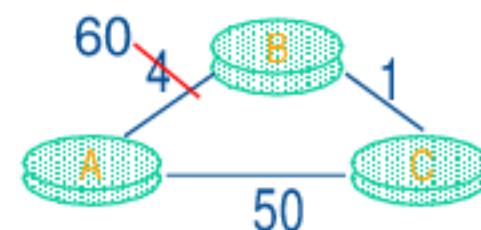
- notícias ruins demoram a se propagar

tabela do nó B

	custo até
	A
via A	4
via C	6

tabela do nó C

	custo até
	A
via A	50
via B	5



TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar

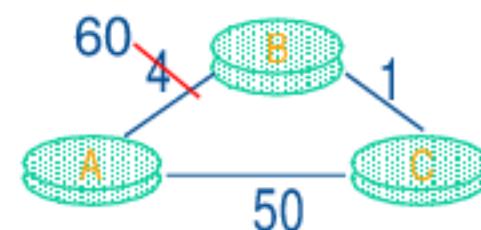
tabela do nó B

	custo até
A	
via A	4
via C	6

tabela do nó C

	custo até
A	
via A	50
via B	5

t_0
 $c(A,B)=60$



TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar

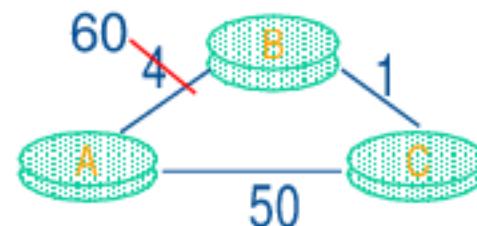


tabela do nó B tabela do nó B

	custo até		custo até
via	A	via	A
via	A C	via	C
	4 6		

tabela do nó C tabela do nó C

	custo até		custo até
via	A	via	A
via	A B	via	B
	50 5		50 5

t0

$c(A,B)=60$

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar

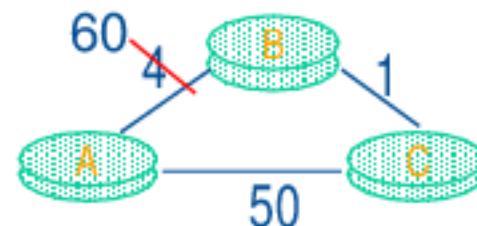


tabela do nó B tabela do nó B

	custo até		custo até
	A		A
via	A	4	60
C	6		

tabela do nó C tabela do nó C

	custo até		custo até
	A		A
via	A	50	50
B	5		5

t0

$c(A,B)=60$

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar

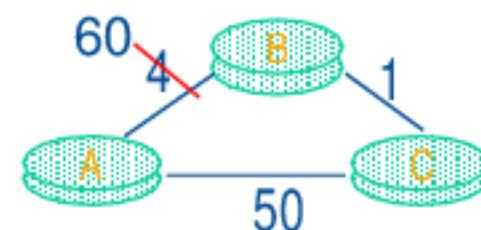
tabela do nó B tabela do nó B

	custo até		custo até
	A		A
via	4	via	60
C	6	C	6

tabela do nó C tabela do nó C

	custo até		custo até
	A		A
via	50	via	50
B	5	B	5

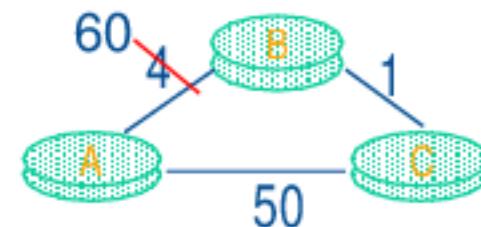
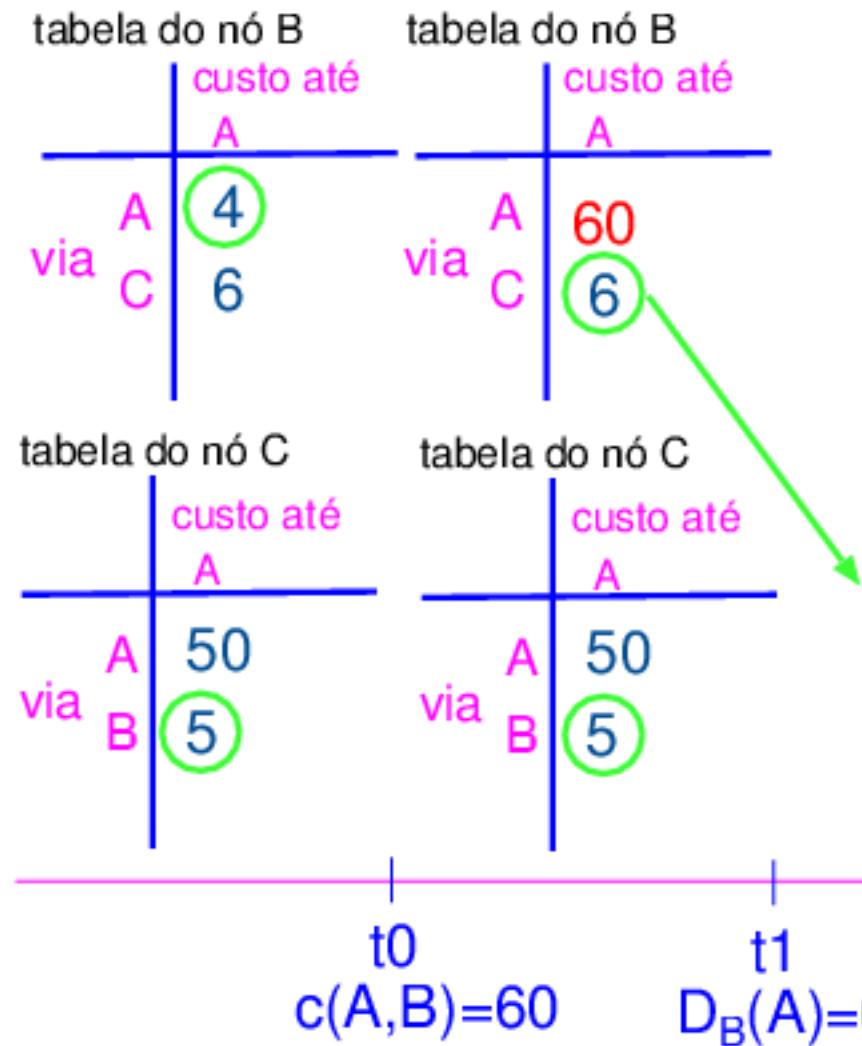
t0 c(A,B)=60



TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar



Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar

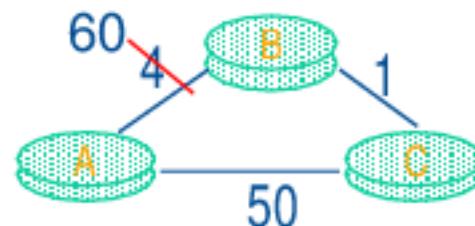


tabela do nó B		tabela do nó B		tabela do nó B	
	custo até		custo até		custo até
	A		A		A
via A	4	via A	60	via A	60
via C	6	via C	6	via C	6

tabela do nó C tabela do nó C tabela do nó C

	custo até	
	A	A
via A	50	50
via B	5	5

	custo até	
	A	A
via A	50	50
via B	5	5

	custo até	
	A	A
via A	50	50
via B	7	7

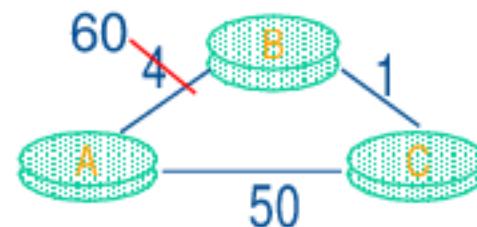
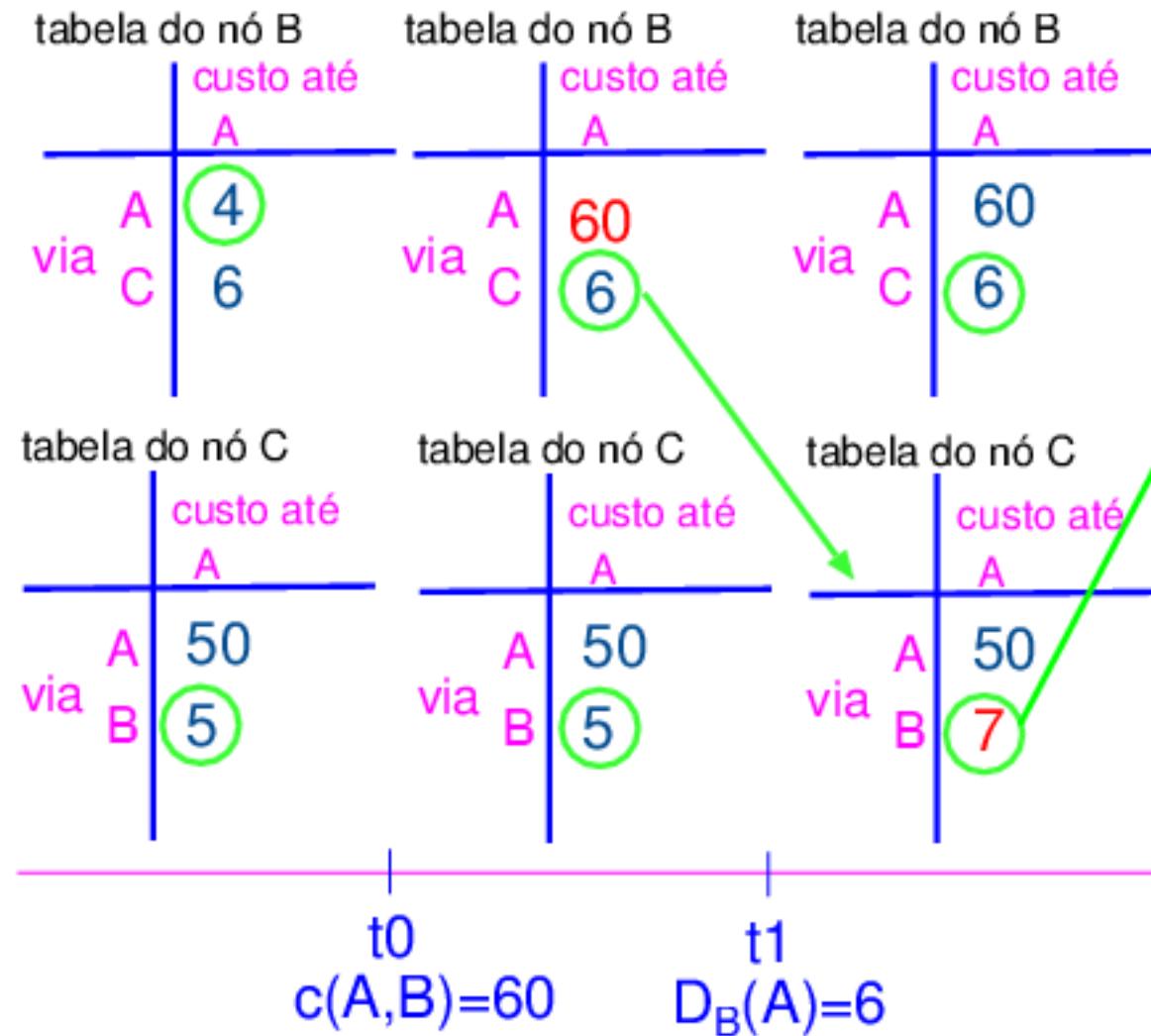
$$c(A,B)=60$$

$$D_B(A)=6$$

TEMPO

Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar



Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar

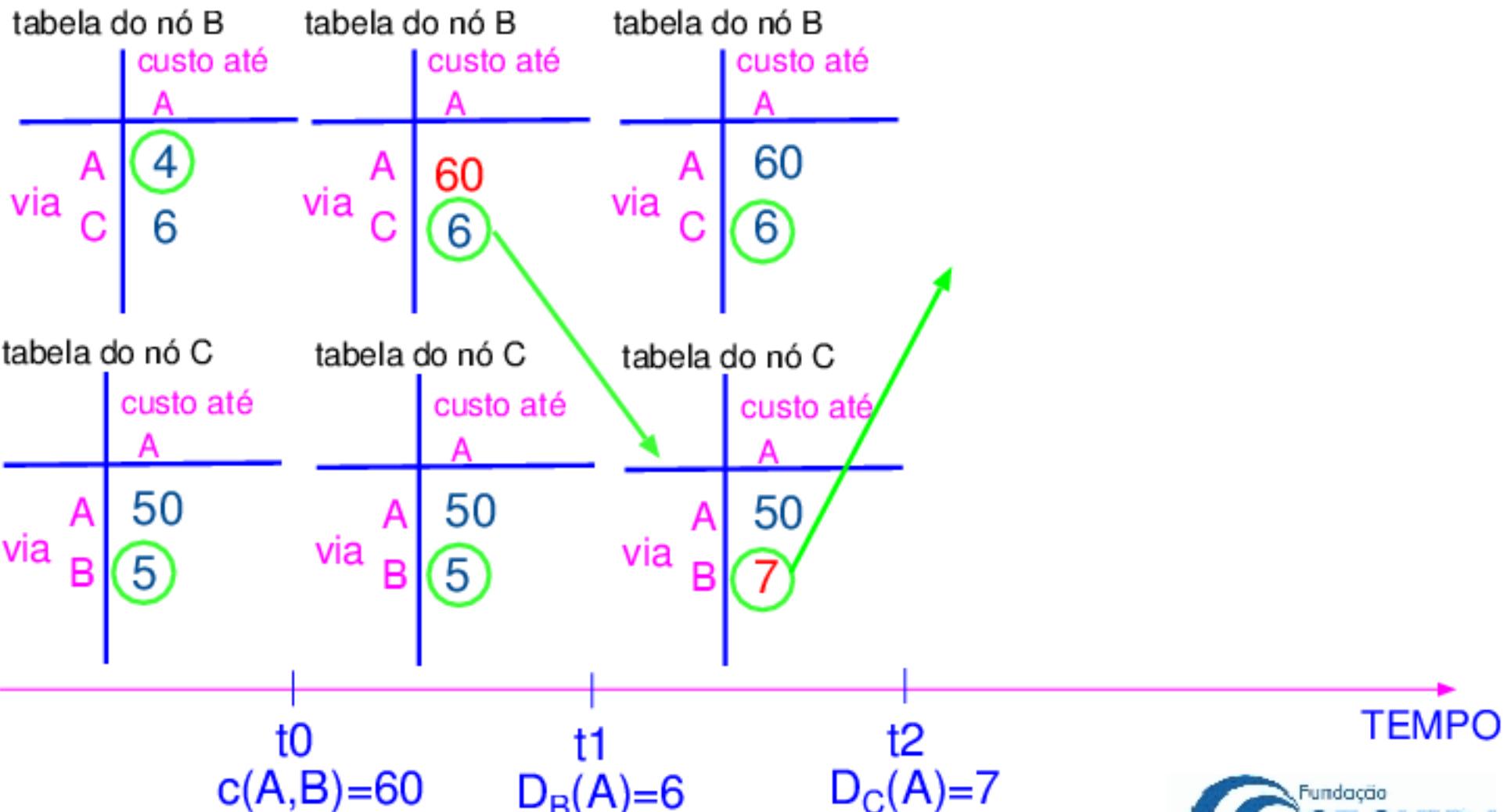
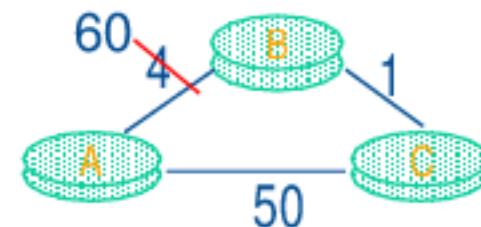
tabela do nó B		tabela do nó B		tabela do nó B	
	custo até		custo até		custo até
via	A	via	A	via	A
A	4	C	60	C	60
C	6		6		6

tabela do nó C		tabela do nó C		tabela do nó C	
	custo até		custo até		custo até
via	A	via	A	via	A
A	50	B	50	B	50
B	5		5		7

t0
 $c(A,B)=60$

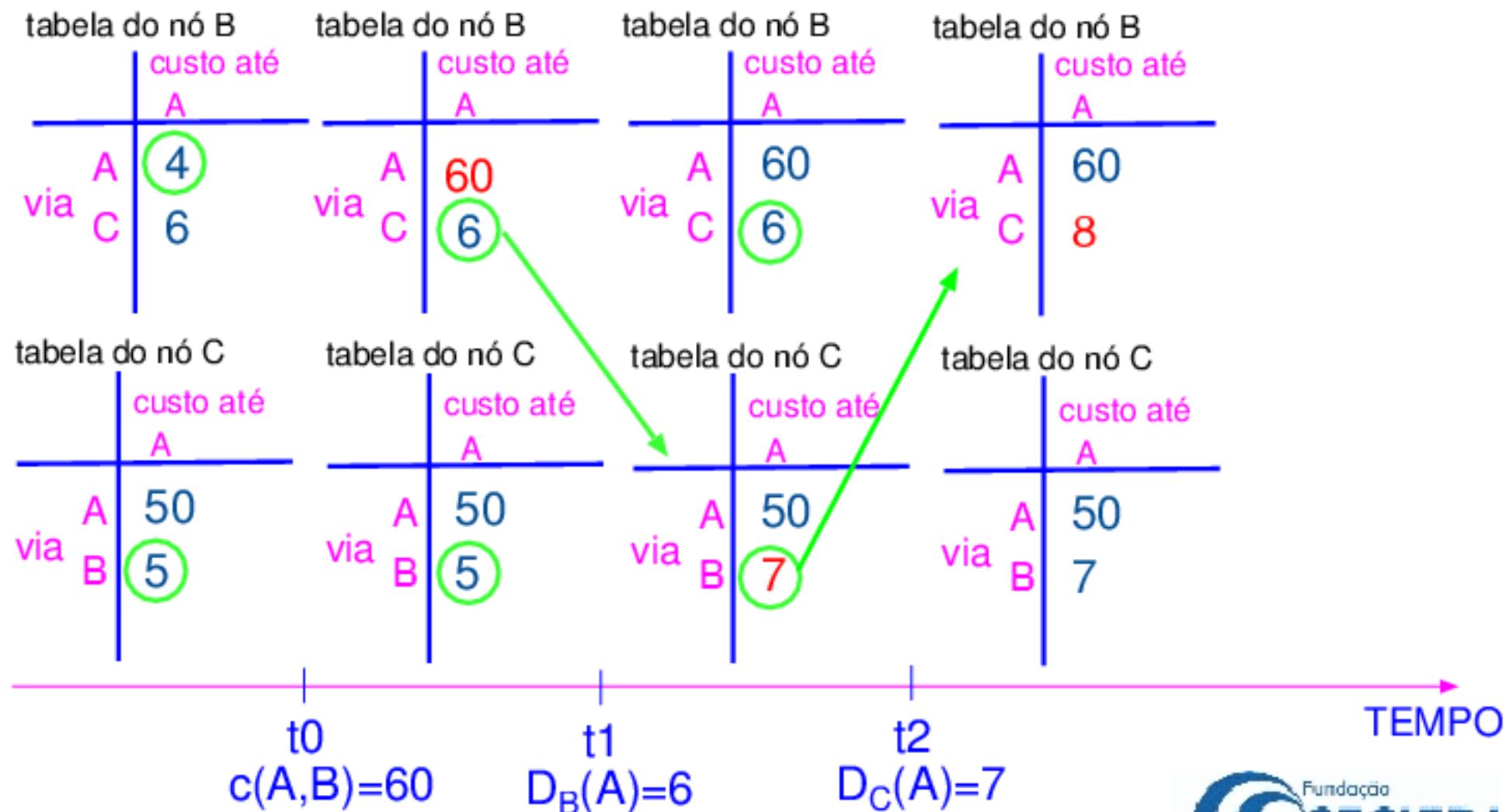
t1
 $D_B(A)=6$

t2
 $D_C(A)=7$



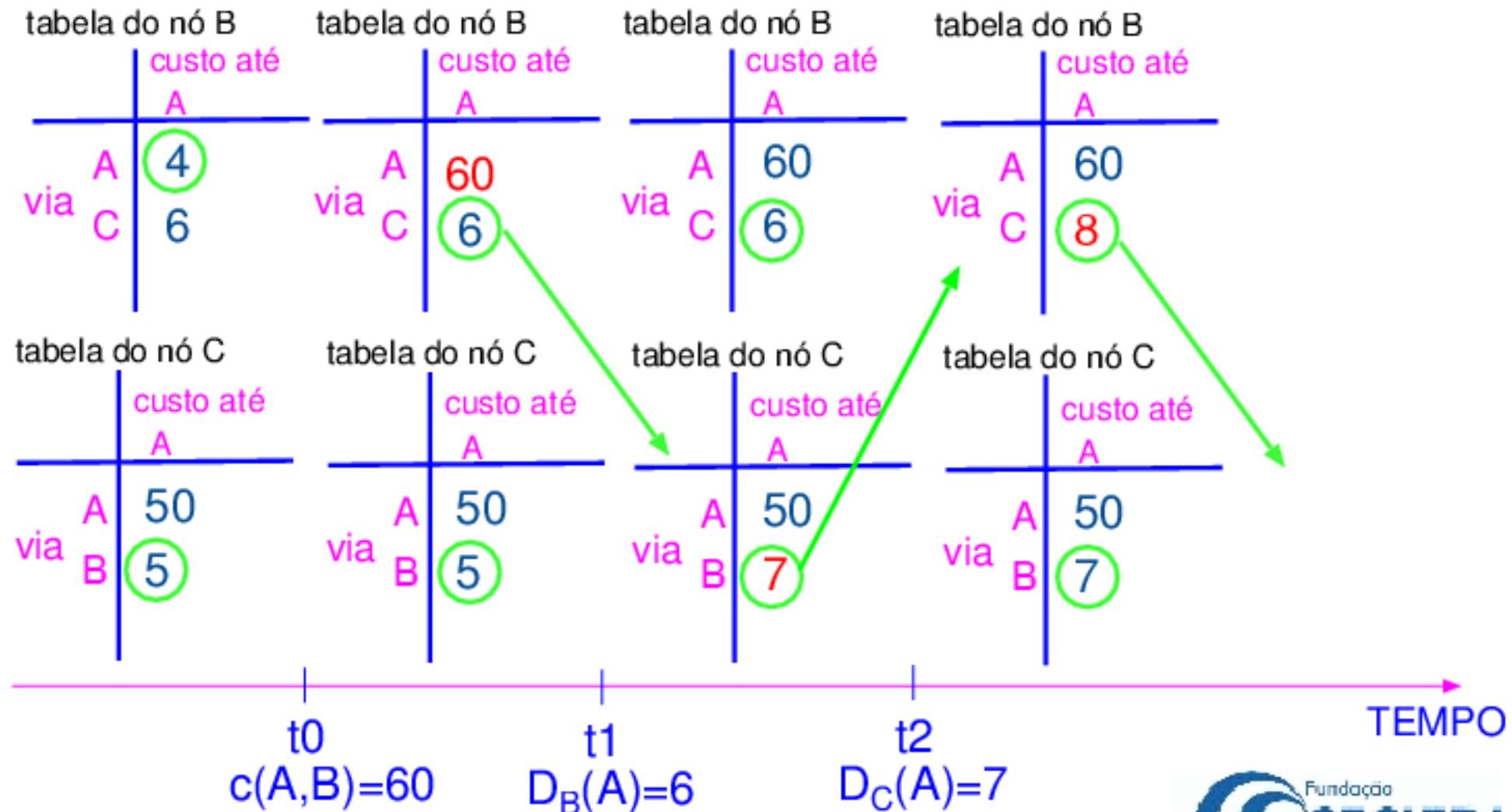
Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar



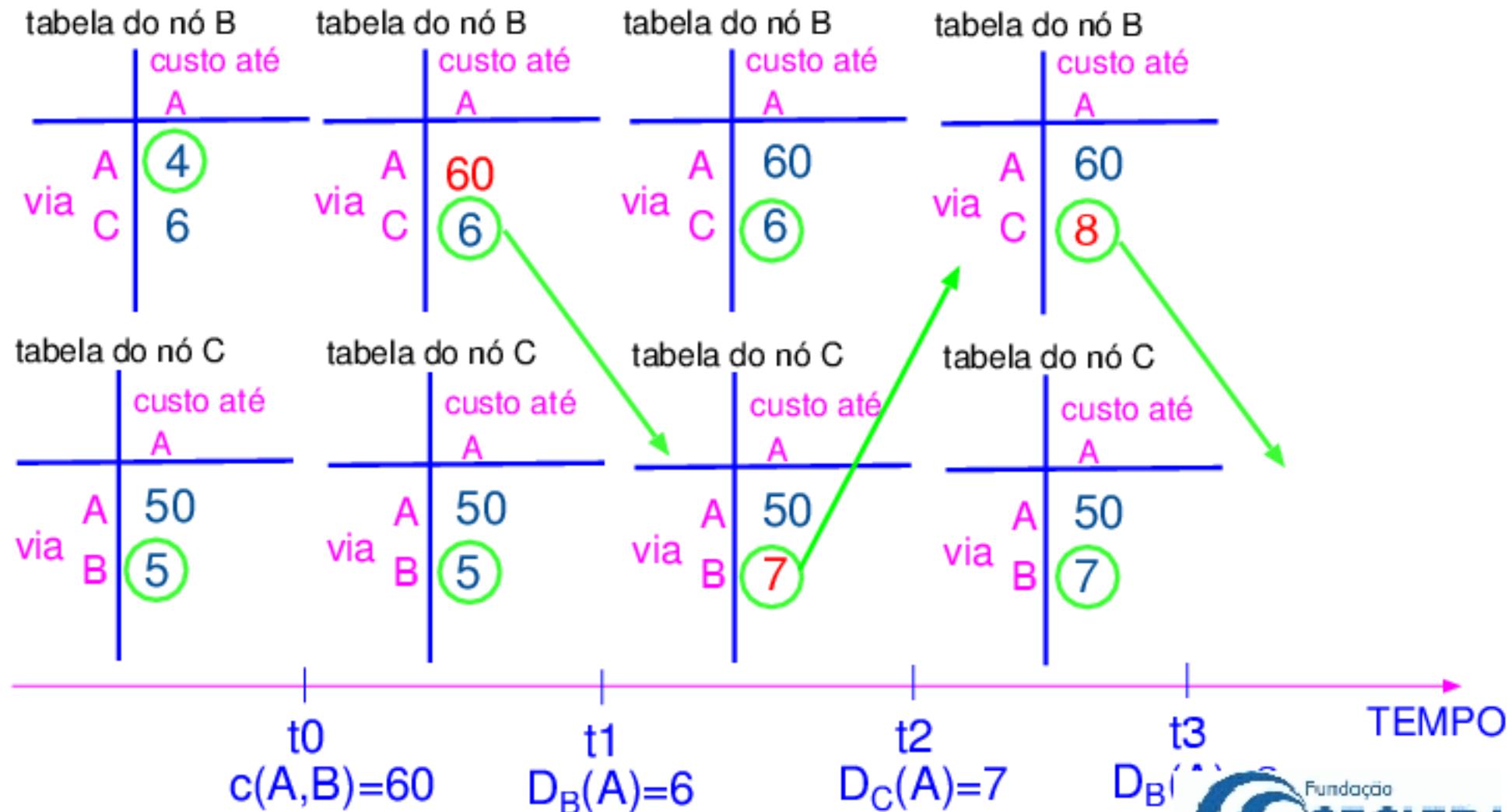
Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar



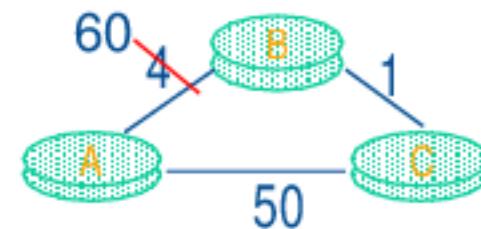
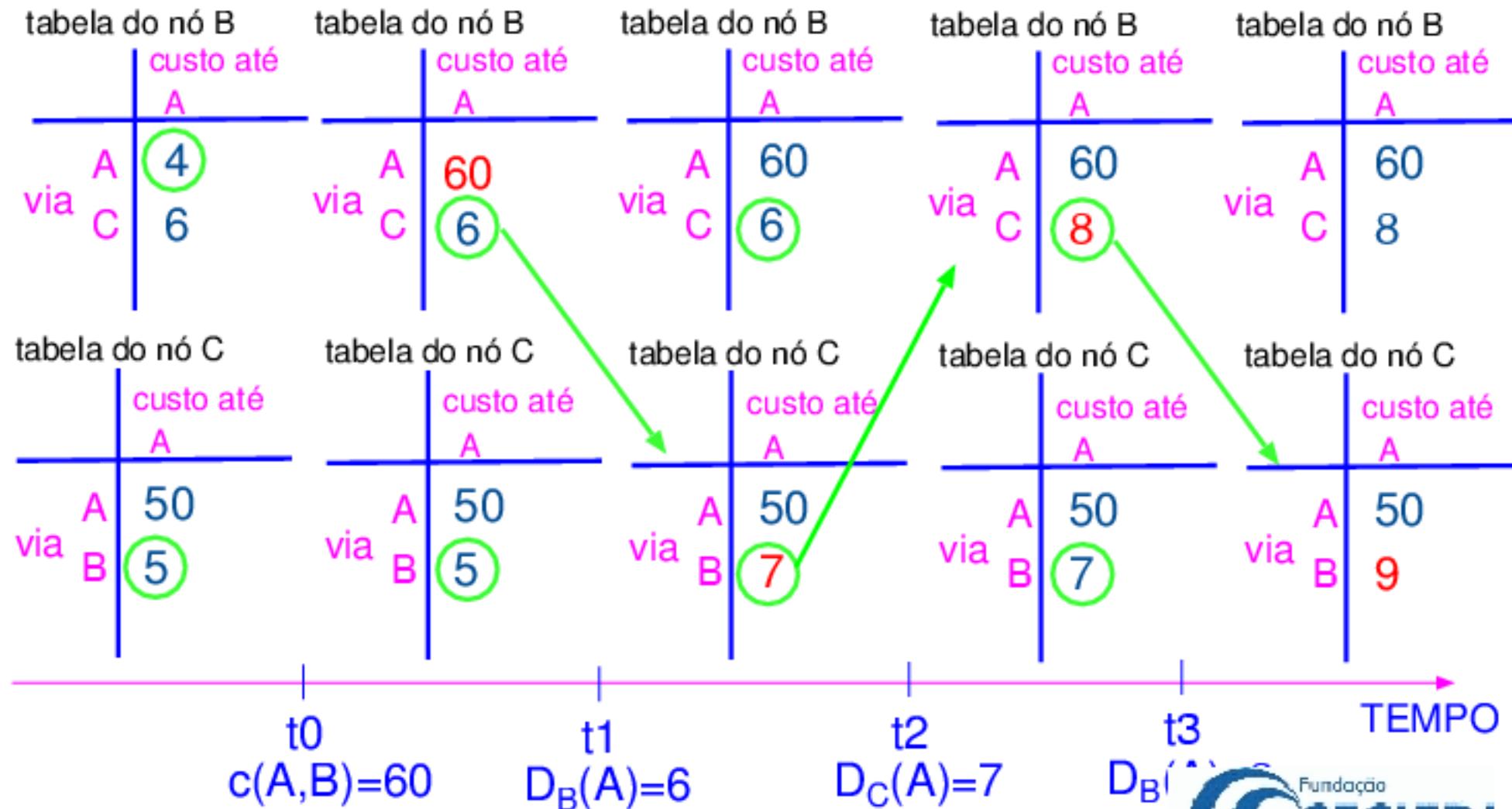
Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar



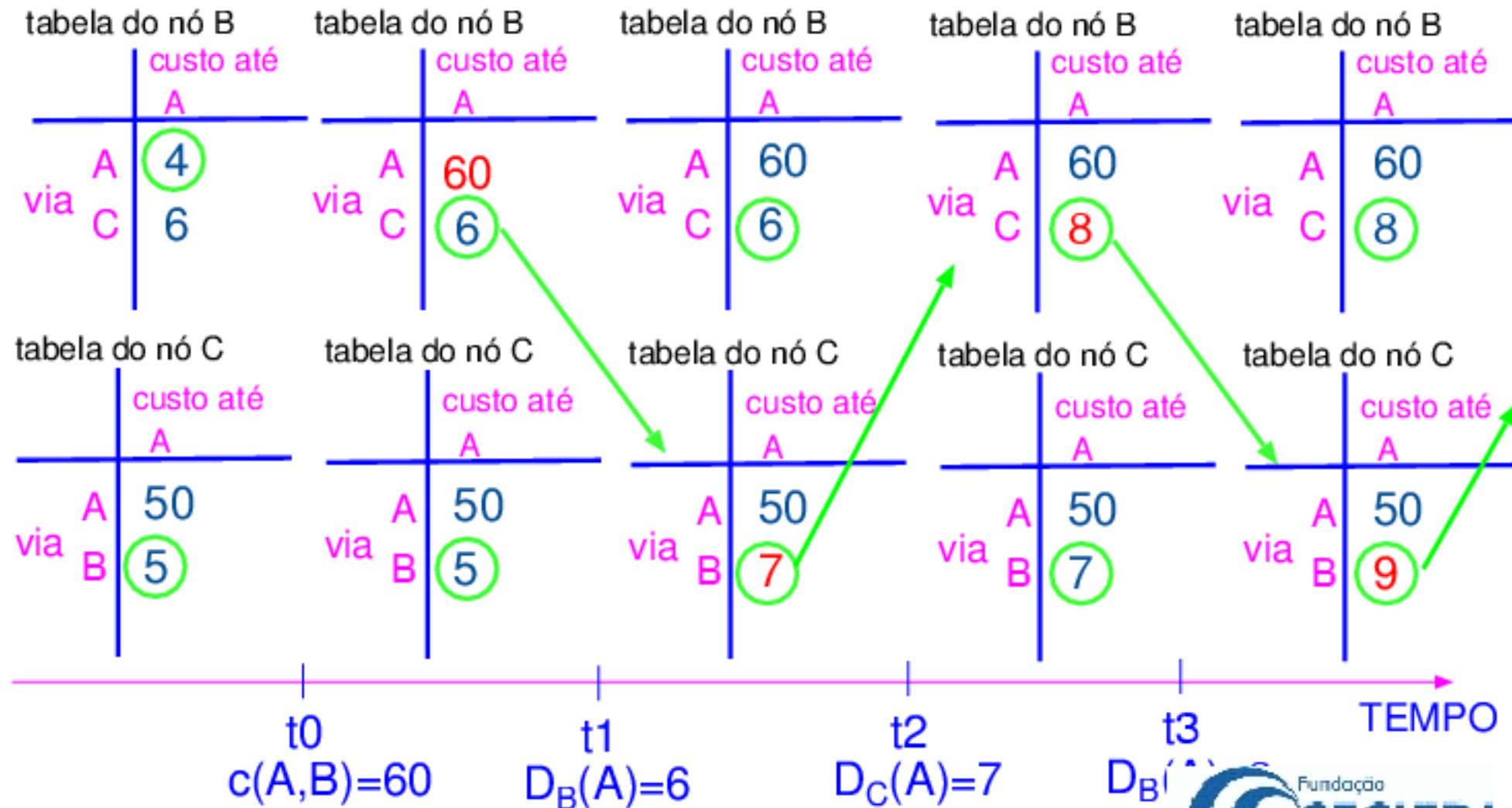
Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar

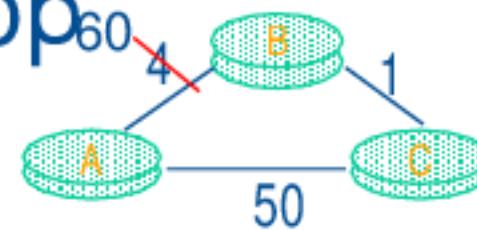


Algoritmo distance-vector: mudança nos custos dos enlaces

- notícias ruins demoram a se propagar



Problema do algoritmo distance-vector: pacotes podem ficar em loop



Problema do algoritmo distance-vector: pacotes podem ficar em loop

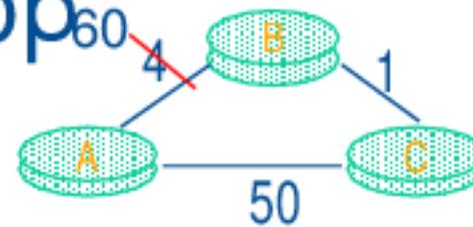


tabela do nó B

		custo até	
		A	
via	A	4	
	C	6	

tabela do nó B

		custo até	
		A	
via	A	60	
	C	6	

tabela do nó C

		custo até	
		A	
via	A	50	
	B	5	

t_0
 $c(A,B)=60$

TEMPO

Problema do algoritmo distance-vector: pacotes podem ficar em loop

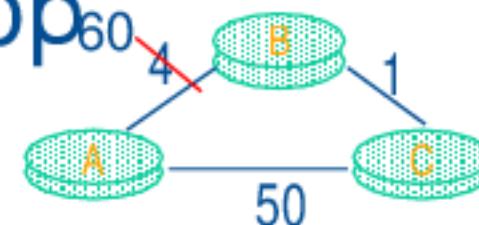
tabela do nó B tabela do nó B

		custo até	
		A	
via	A	4	
	C	6	

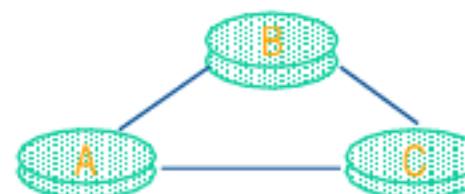
tabela do nó C tabela do nó C

		custo até	
		A	
via	A	50	
	B	5	

$\xrightarrow{t_0}$
 $c(A,B)=60$



Problema: B não pode escolher esta rota pois ela passa pelo próprio B !
--> pacotes podem ficar em loop !



TEMPO

Problema do algoritmo distance-vector: pacotes podem ficar em loop

tabela do nó B tabela do nó B

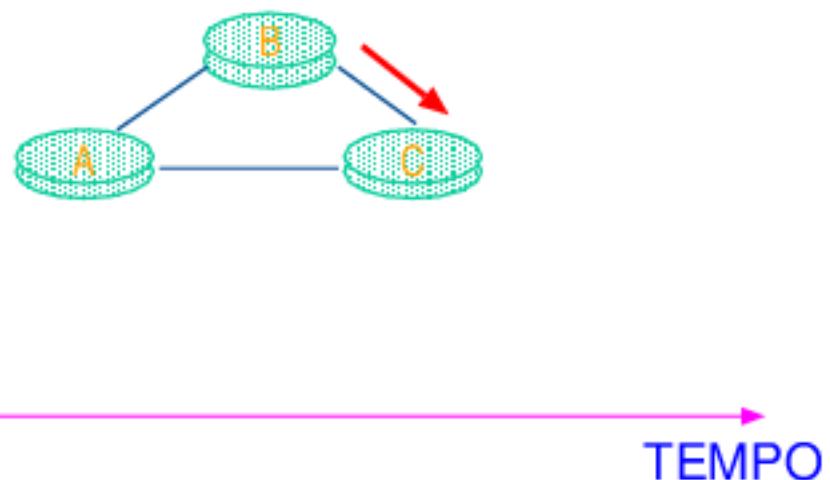
		custo até	
		A	
via	A	4	
	C	6	

tabela do nó C tabela do nó C

		custo até	
		A	
via	A	50	
	B	5	

$\xrightarrow{t_0}$
 $c(A,B)=60$

Problema: B não pode escolher esta rota pois ela passa pelo próprio B !
--> pacotes podem ficar em loop !



Problema do algoritmo distance-vector: pacotes podem ficar em loop

tabela do nó B tabela do nó B

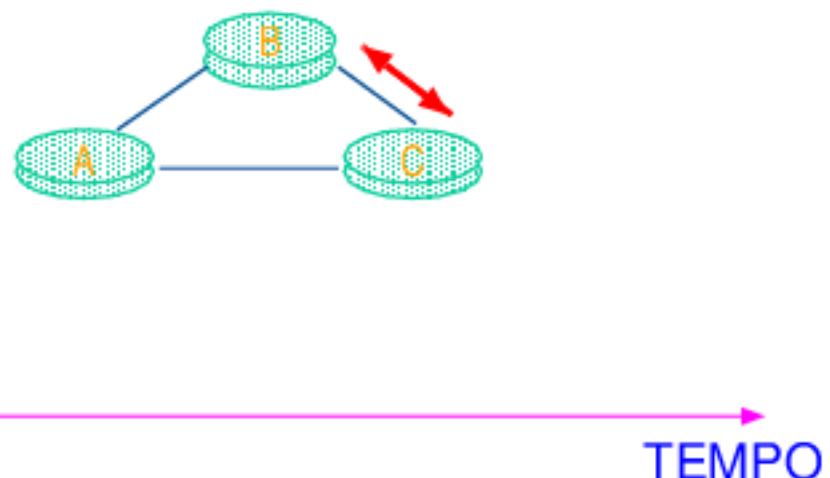
		custo até	
		A	
via	A	4	
	C	6	

tabela do nó C tabela do nó C

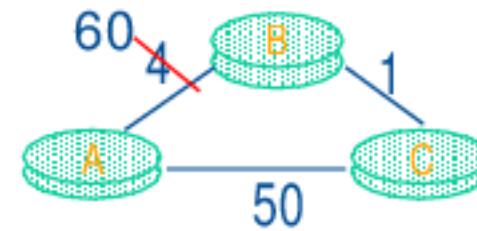
		custo até	
		A	
via	A	50	
	B	5	

$\xrightarrow{t_0}$
 $c(A,B)=60$

Problema: B não pode escolher esta rota pois ela passa pelo próprio B !
--> pacotes podem ficar em loop !

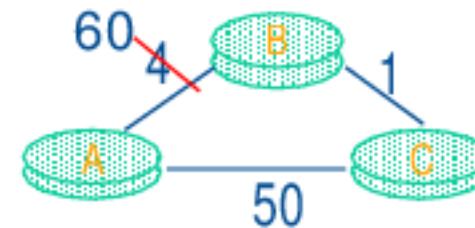


Algoritmo distance-vector: problema contagem até o infinito



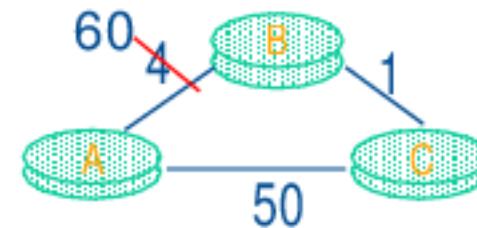
Algoritmo distance-vector: problema contagem até o infinito

- Problema relativo a notícias ruins demorarem a se propagar é conhecido como problema de contagem até o infinito (counting to infinity problem)



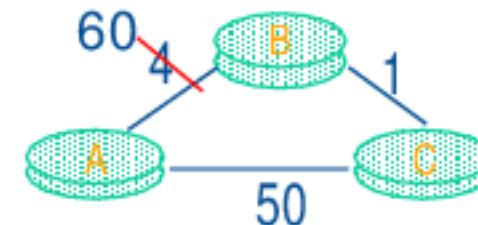
Algoritmo distance-vector: problema contagem até o infinito

- Problema relativo a notícias ruins demorarem a se propagar é conhecido como problema de contagem até o infinito (counting to infinity problem)
- Nestes casos o algoritmo pode demorar muito a convergir



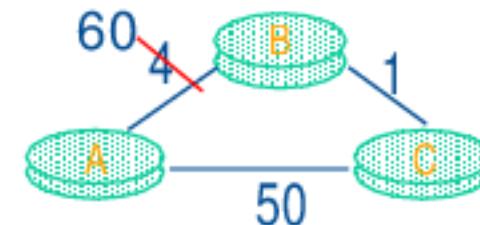
Algoritmo distance-vector: problema contagem até o infinito

- Problema relativo a notícias ruins demorarem a se propagar é conhecido como problema de contagem até o infinito (counting to infinity problem)
- Nestes casos o algoritmo pode demorar muito a convergir
- Uma solução para o problema:



Algoritmo distance-vector: problema contagem até o infinito

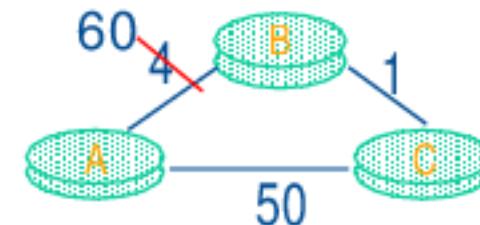
- Problema relativo a notícias ruins demorarem a se propagar é conhecido como problema de contagem até o infinito (counting to infinity problem)
- Nestes casos o algoritmo pode demorar muito a convergir
- Uma solução para o problema:



- O nó C notifica B que sua distância até A é infinita pois B está na rota de C para alcançar A

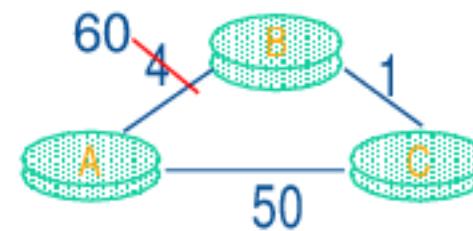
Algoritmo distance-vector: problema contagem até o infinito

- Problema relativo a notícias ruins demorarem a se propagar é conhecido como problema de contagem até o infinito (counting to infinity problem)
- Nestes casos o algoritmo pode demorar muito a convergir
- Uma solução para o problema:



- O nó C notifica B que sua distância até A é infinita pois B está na rota de C para alcançar A
- Esta solução é conhecida como inversão envenenada (poisoned reverse)

Algoritmo distance-vector: inversão envenenada



Algoritmo distance-vector: inversão envenenada

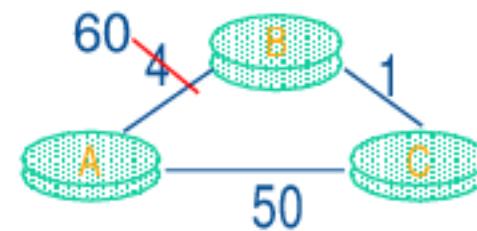


tabela do nó B

		custo até
		A
via	A	4
	C	∞

tabela do nó C

		custo até
		A
via	A	50
	B	5

TEMPO

Algoritmo distance-vector: inversão envenenada

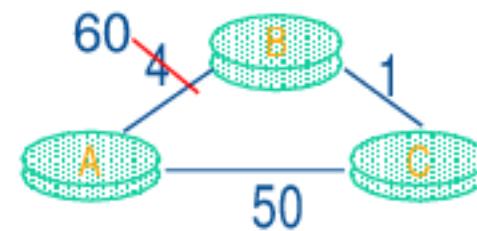


tabela do nó B

		custo até
		A
via	A	4
	C	∞

tabela do nó C

		custo até
		A
via	A	50
	B	5

TEMPO

Algoritmo distance-vector: inversão envenenada

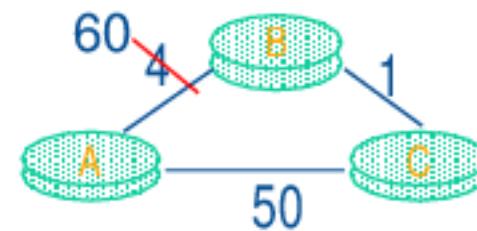


tabela do nó B

		custo até
		A
via	A	4
	C	∞

tabela do nó C

		custo até
		A
via	A	50
	B	5

TEMPO

Algoritmo distance-vector: inversão envenenada

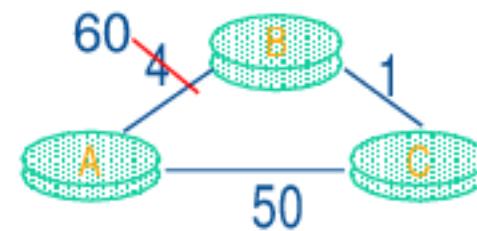


tabela do nó B

		custo até
		A
via	A	4
	C	∞

tabela do nó C

		custo até
		A
via	A	50
	B	5

t_0
 $c(A,B)=60$

TEMPO

Algoritmo distance-vector: inversão envenenada

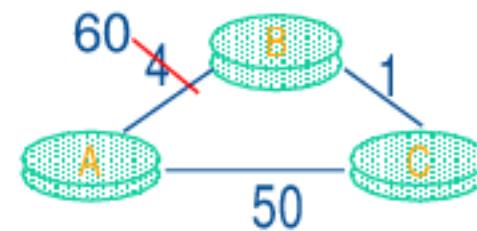


tabela do nó B

	custo até		custo até
	A		A
via	A		C
A	4		∞
C	∞		

tabela do nó B

	custo até		custo até
	A		A
via	A		C
A	∞		
C			∞

tabela do nó C

	custo até		custo até
	A		A
via	A		B
A	50		
B	5		

t_0
 $c(A,B)=60$

TEMPO

Algoritmo distance-vector: inversão envenenada

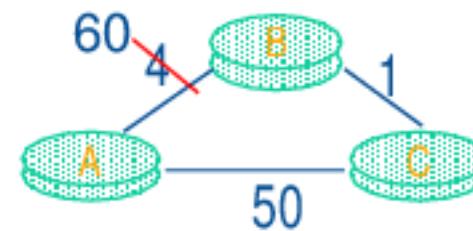


tabela do nó B

	custo até	
	A	C
via	4	∞

tabela do nó B

	custo até	
	A	C
via	60	∞

tabela do nó C

	custo até	
	A	B
via	50	5

t_0
 $c(A,B)=60$

TEMPO

Algoritmo distance-vector: inversão envenenada

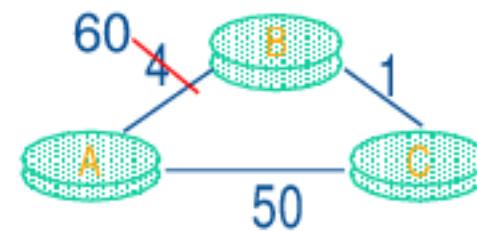


tabela do nó B

	custo até	A
via	A	4
C	∞	

tabela do nó B

	custo até	A
via	A	60
C	∞	

tabela do nó C

	custo até	A
via	B	50
A	∞	

tabela do nó C

	custo até	A
via	B	5
A	∞	

t_0
 $c(A,B)=60$

TEMPO

Algoritmo distance-vector: inversão envenenada

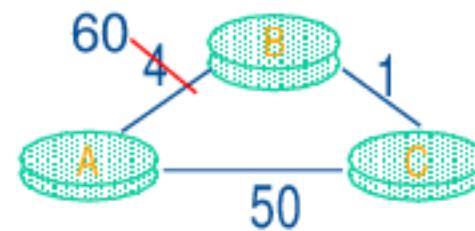


tabela do nó B

	custo até	A
via	A	4
C	∞	

tabela do nó B

	custo até	A
via	A	60
C	∞	

tabela do nó C

	custo até	A
via	A	50
B	∞	

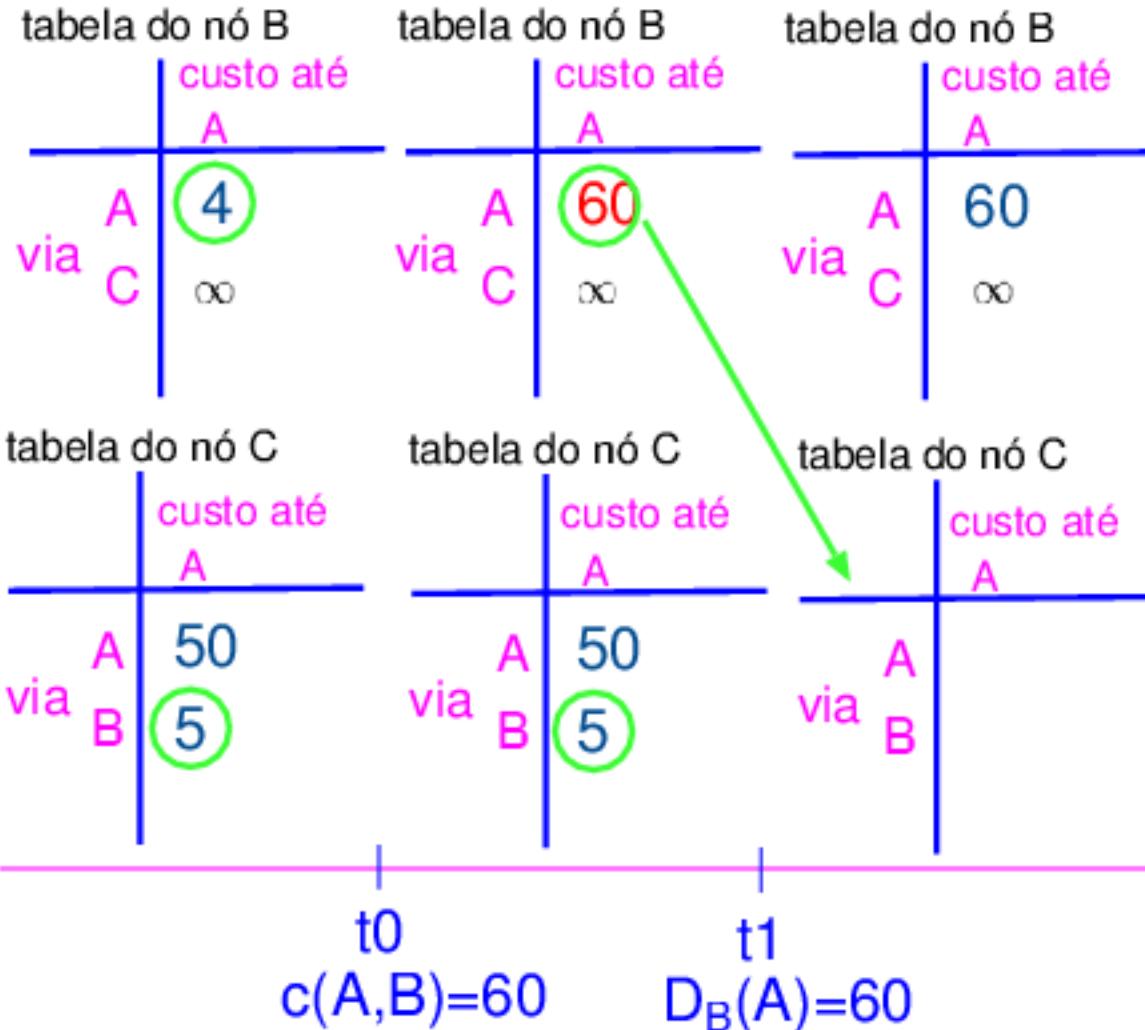
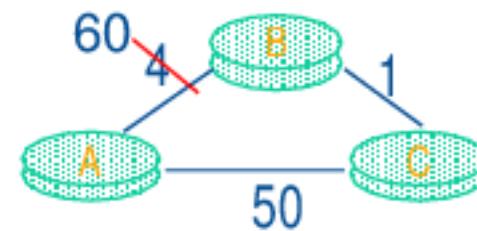
tabela do nó C

	custo até	A
via	A	50
B	∞	

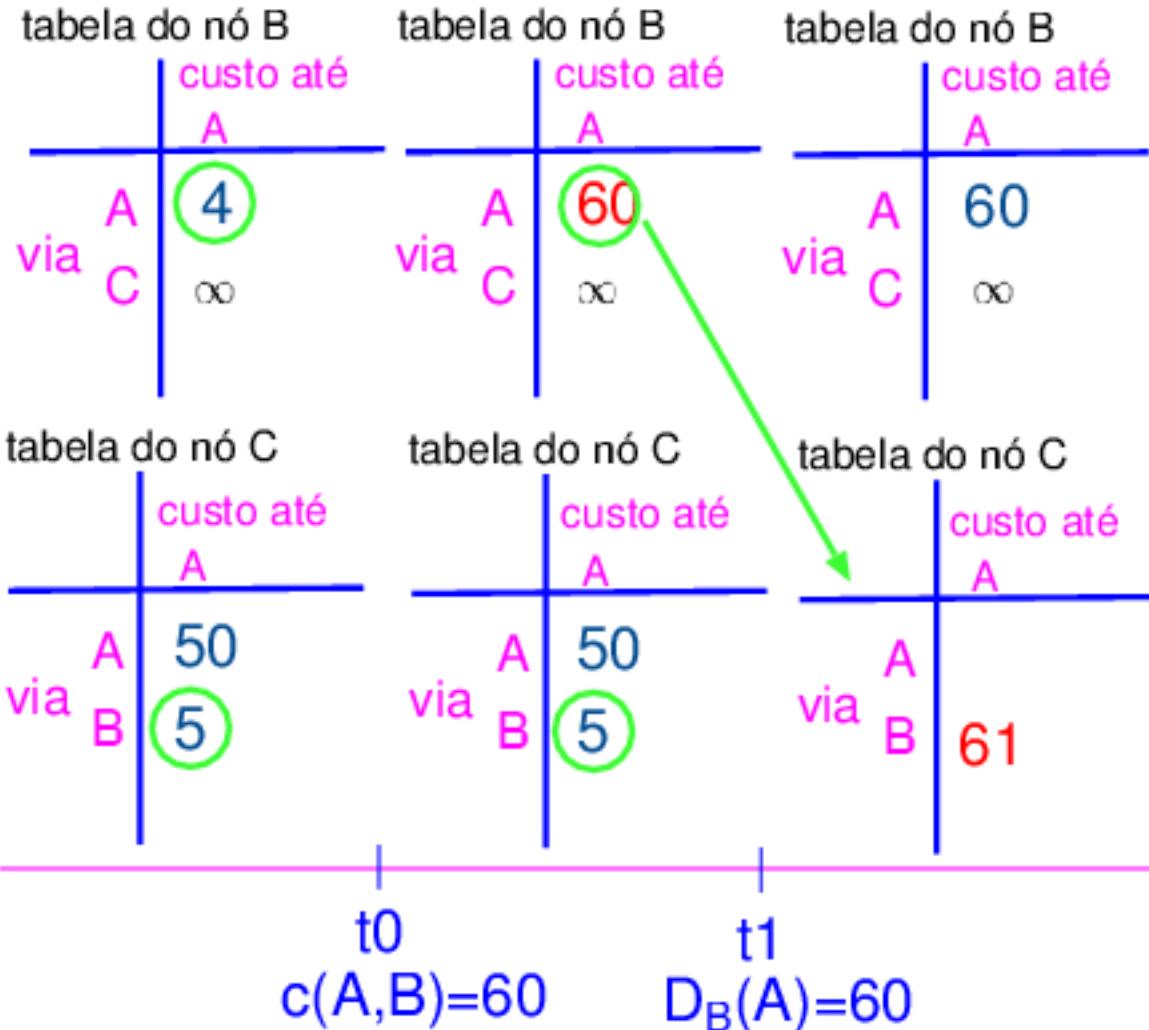
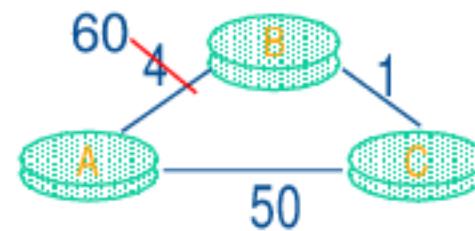
t_0 t_1
 $c(A,B)=60$ $D_B(A)=60$

TEMPO

Algoritmo distance-vector: inversão envenenada

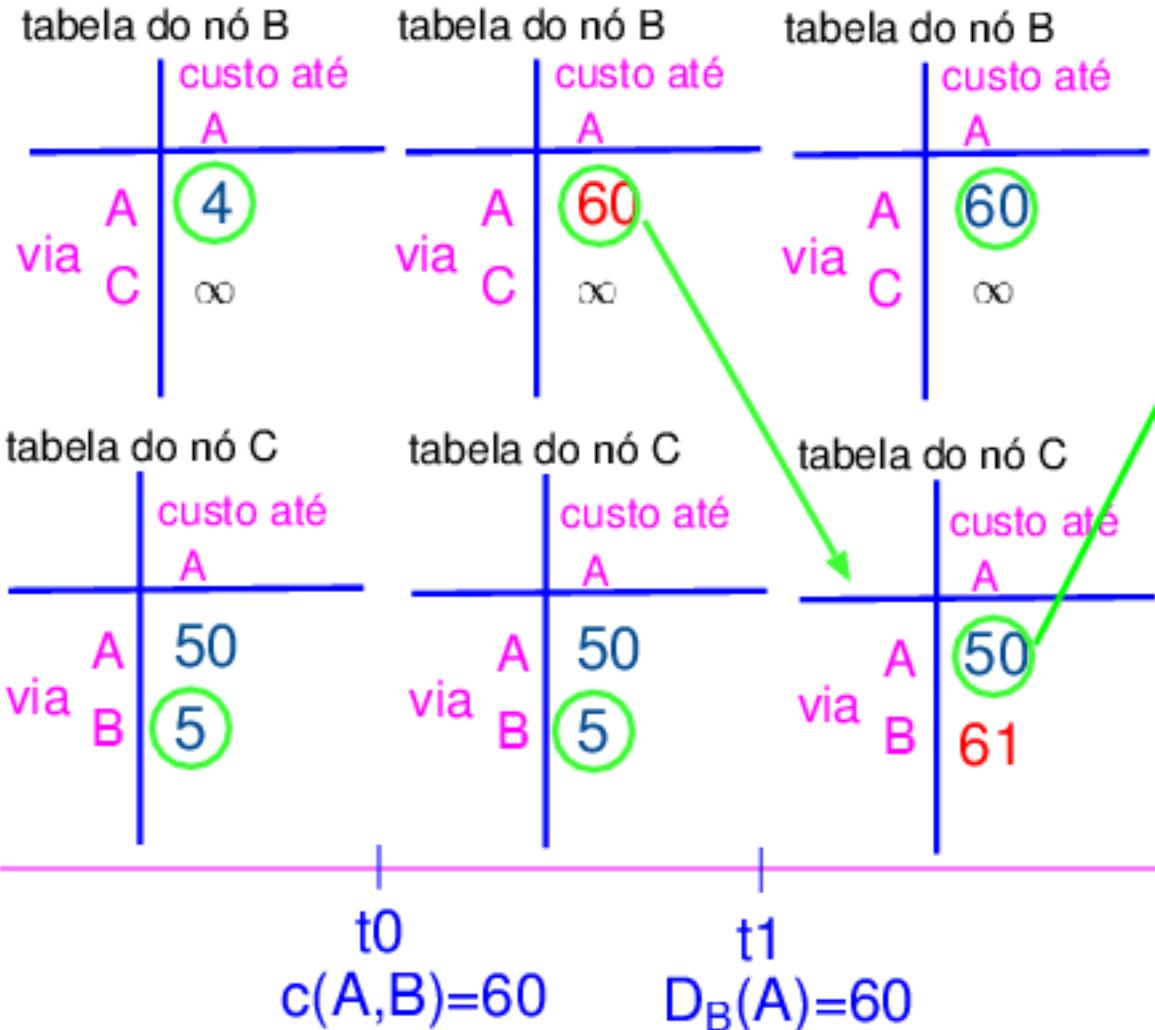
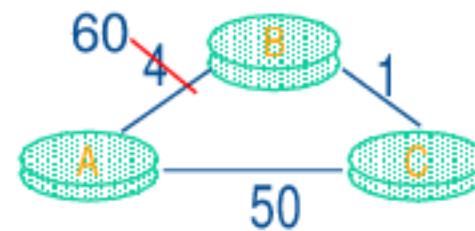


Algoritmo distance-vector: inversão envenenada

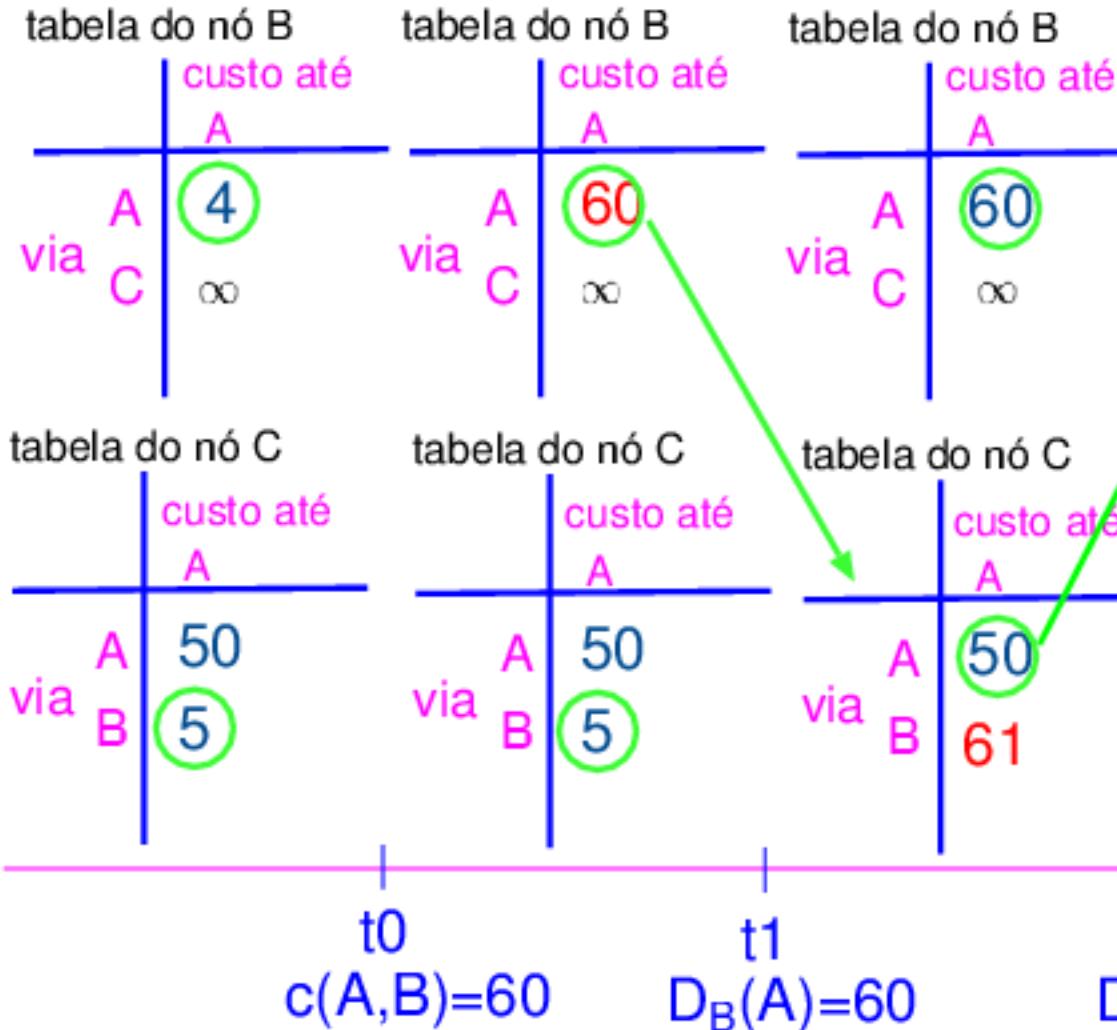
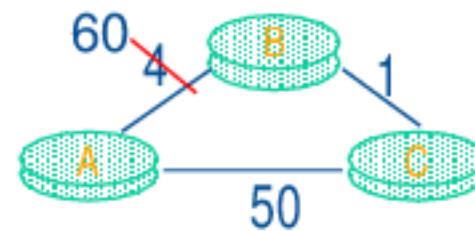


TEMPO

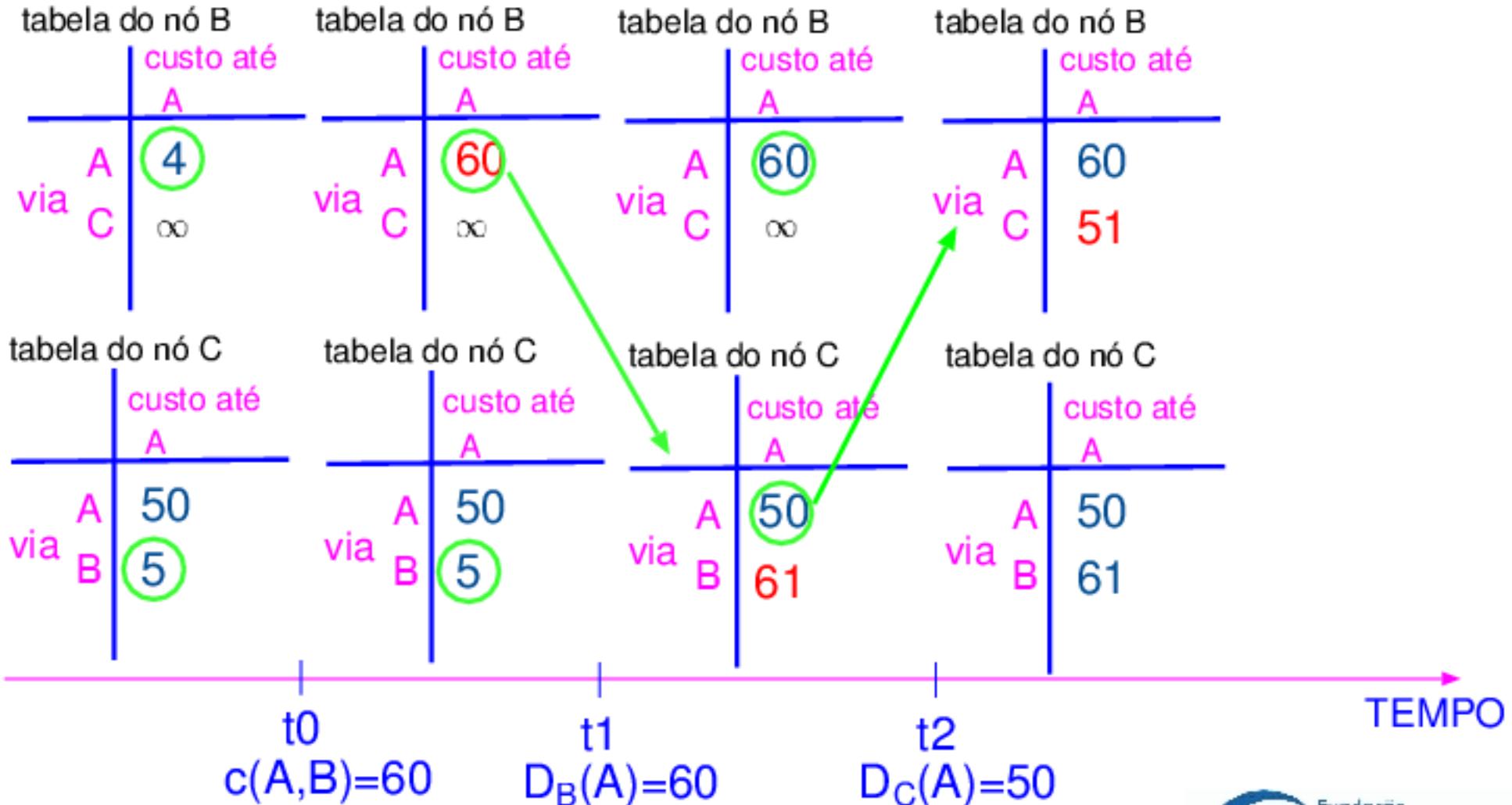
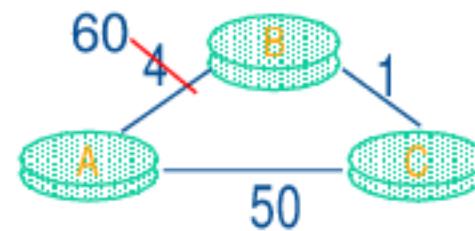
Algoritmo distance-vector: inversão envenenada



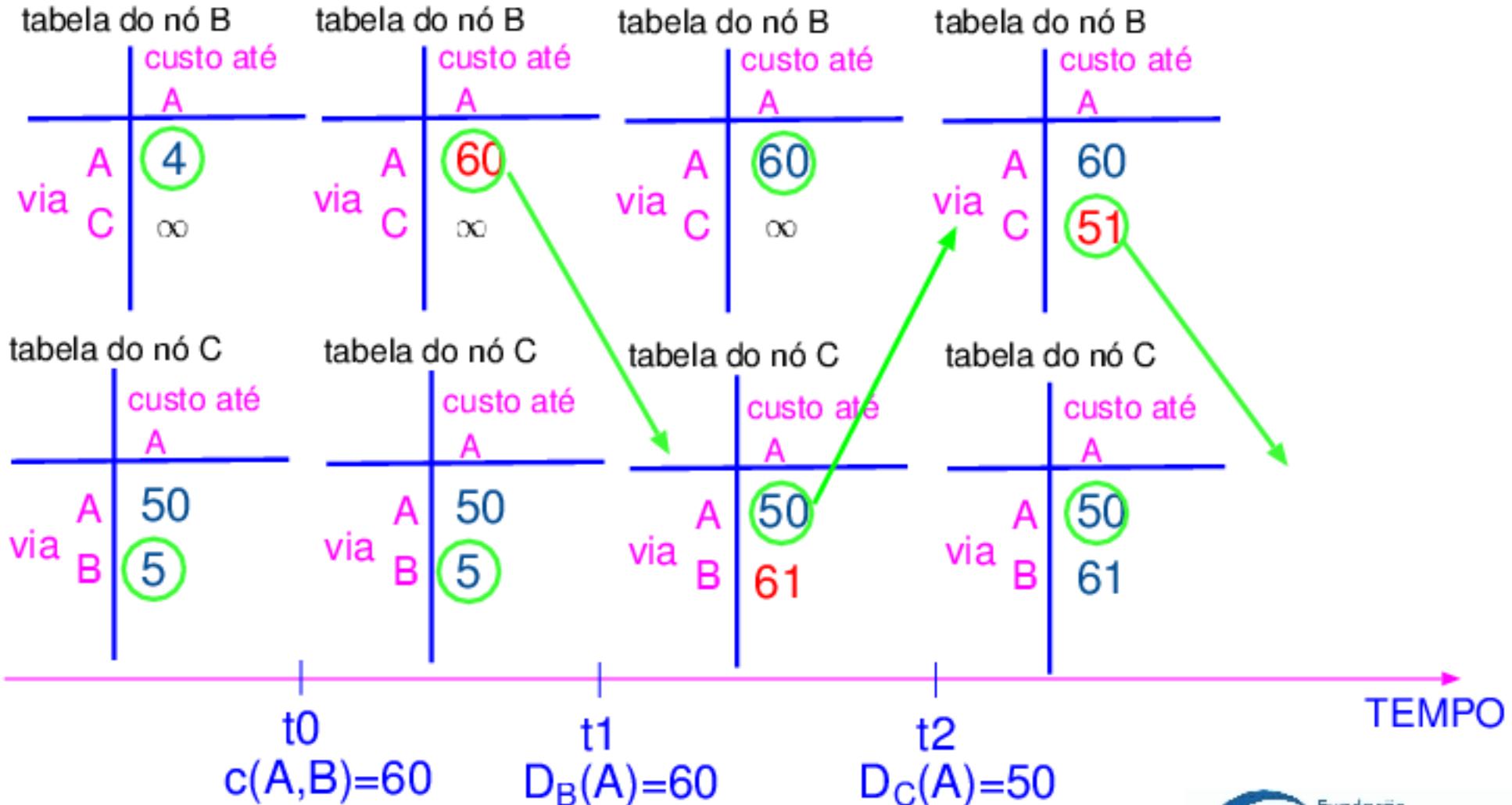
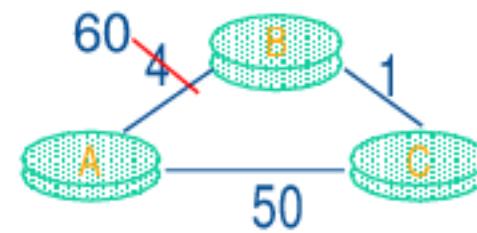
Algoritmo distance-vector: inversão envenenada



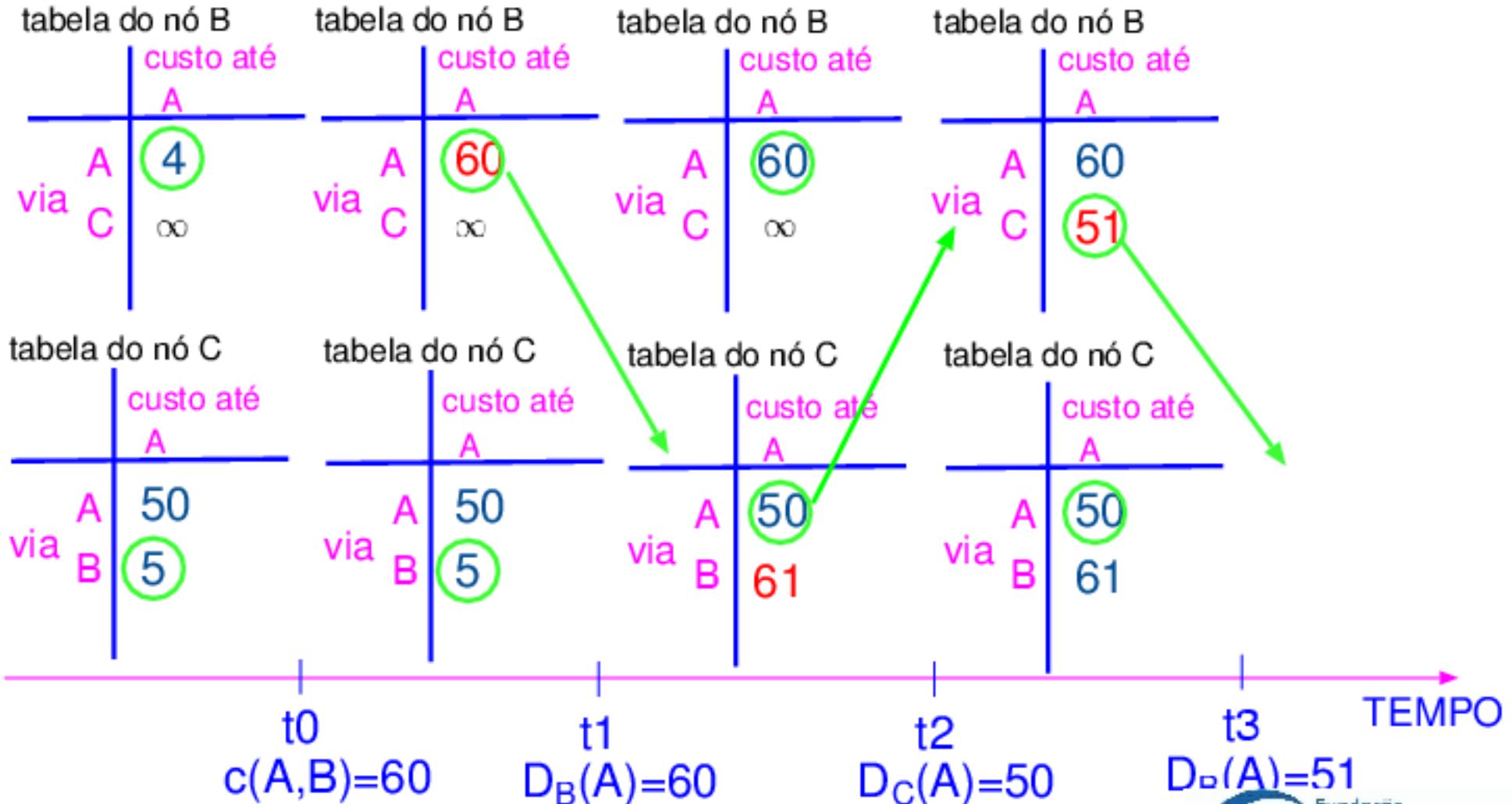
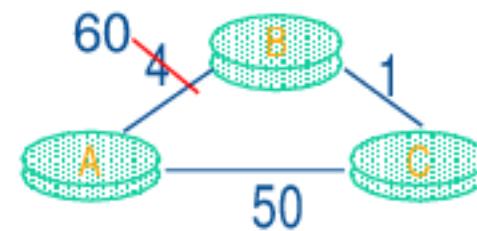
Algoritmo distance-vector: inversão envenenada



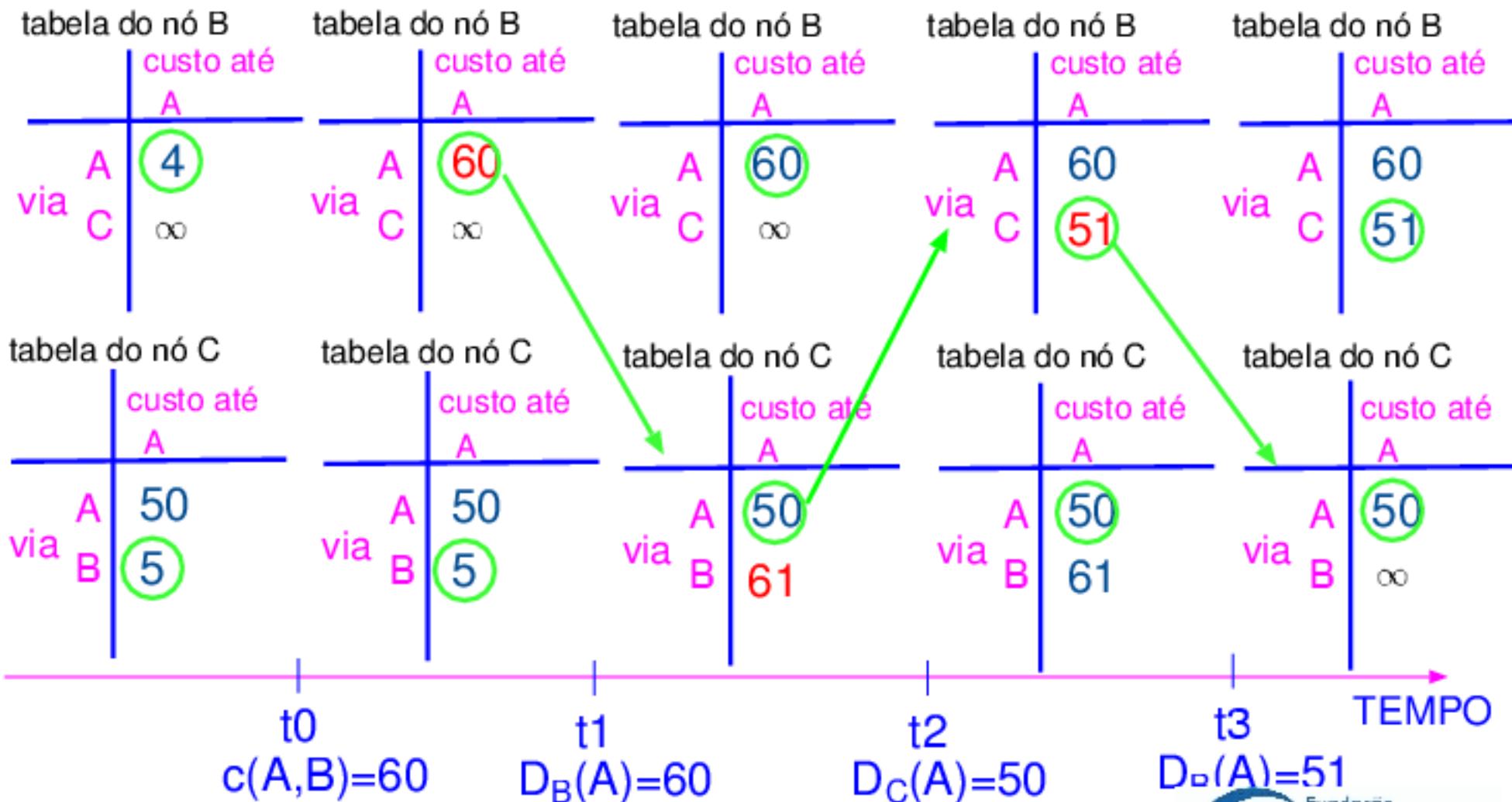
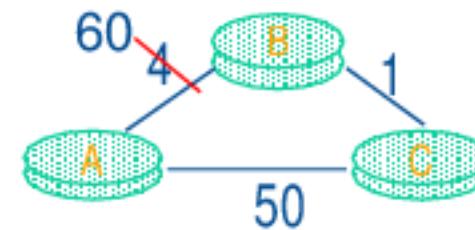
Algoritmo distance-vector: inversão envenenada



Algoritmo distance-vector: inversão envenenada



Algoritmo distance-vector: inversão envenenada



Comparação entre algoritmos link-state (LS) e distance-vector (DV)

Comparação entre algoritmos link-state (LS) e distance-vector (DV)

→ Complexidade do número de mensagens

Comparação entre algoritmos link-state (LS) e distance-vector (DV)

→ Complexidade do número de mensagens

- LS - com N nós, E enlaces, $O(nE)$ mensagens são enviadas

Comparação entre algoritmos link-state (LS) e distance-vector (DV)

→ Complexidade do número de mensagens

- LS - com N nós, E enlaces, $O(nE)$ mensagens são enviadas
- DV - troca somente entre vizinhos, depende da velocidade de convergência

Comparação entre algoritmos link-state (LS) e distance-vector (DV)

→ Complexidade do número de mensagens

- LS - com N nós, E enlaces, $O(nE)$ mensagens são enviadas
- DV - troca somente entre vizinhos, depende da velocidade de convergência

→ Robustez: o que ocorre se um nó funciona de forma incorreta ?

Comparação entre algoritmos link-state (LS) e distance-vector (DV)

→ Complexidade do número de mensagens

- LS - com N nós, E enlaces, $O(nE)$ mensagens são enviadas
- DV - troca somente entre vizinhos, depende da velocidade de convergência

→ Robustez: o que ocorre se um nó funciona de forma incorreta ?

- LS - nó pode informar custo de **enlace** incorreto
 - cada nó computa a sua própria tabela de roteamento
 - erro que se propaga é somente do custo de enlace

Comparação entre algoritmos link-state (LS) e distance-vector (DV)

→ Complexidade do número de mensagens

- LS - com N nós, E enlaces, $O(nE)$ mensagens são enviadas
- DV - troca somente entre vizinhos, depende da velocidade de convergência

→ Robustez: o que ocorre se um nó funciona de forma incorreta ?

- LS - nó pode informar custo de **enlace** incorreto
 - cada nó computa a sua própria tabela de roteamento
 - erro que se propaga é somente do custo de enlace
- DV - nó pode informar custo de **caminhos** incorreto (tabela de roteamento incorreta)
 - os nós usam as tabelas enviadas pelos vizinhos para calcular a sua própria tabela de roteamento
 - erro se propaga pela rede, podendo levar a tabelas de roteamento totalmente erradas