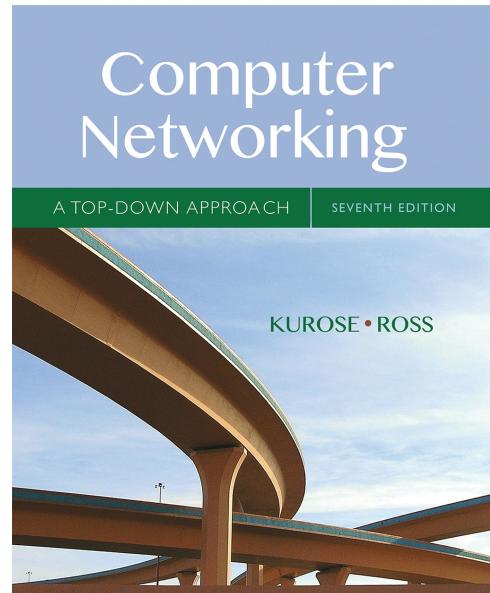


# Conceitos, Algoritmos e Protocolos de Encaminhamento

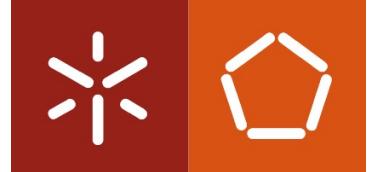


**Comunicações por Computador**  
Mestrado Integrado em Engenharia Informática

3º ano/2º semestre  
2021/2022



***Computer Networking: A Top Down Approach, Capítulo 4 e 5***  
***Jim Kurose, Keith Ross, Addison-Wesley ©2016 .***



# Sumário

## ● Encaminhamento

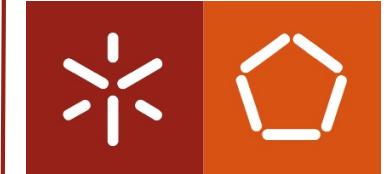
- Processo de Forwarding
- Processo de Routing

## ● Algoritmos de encaminhamento dinâmico

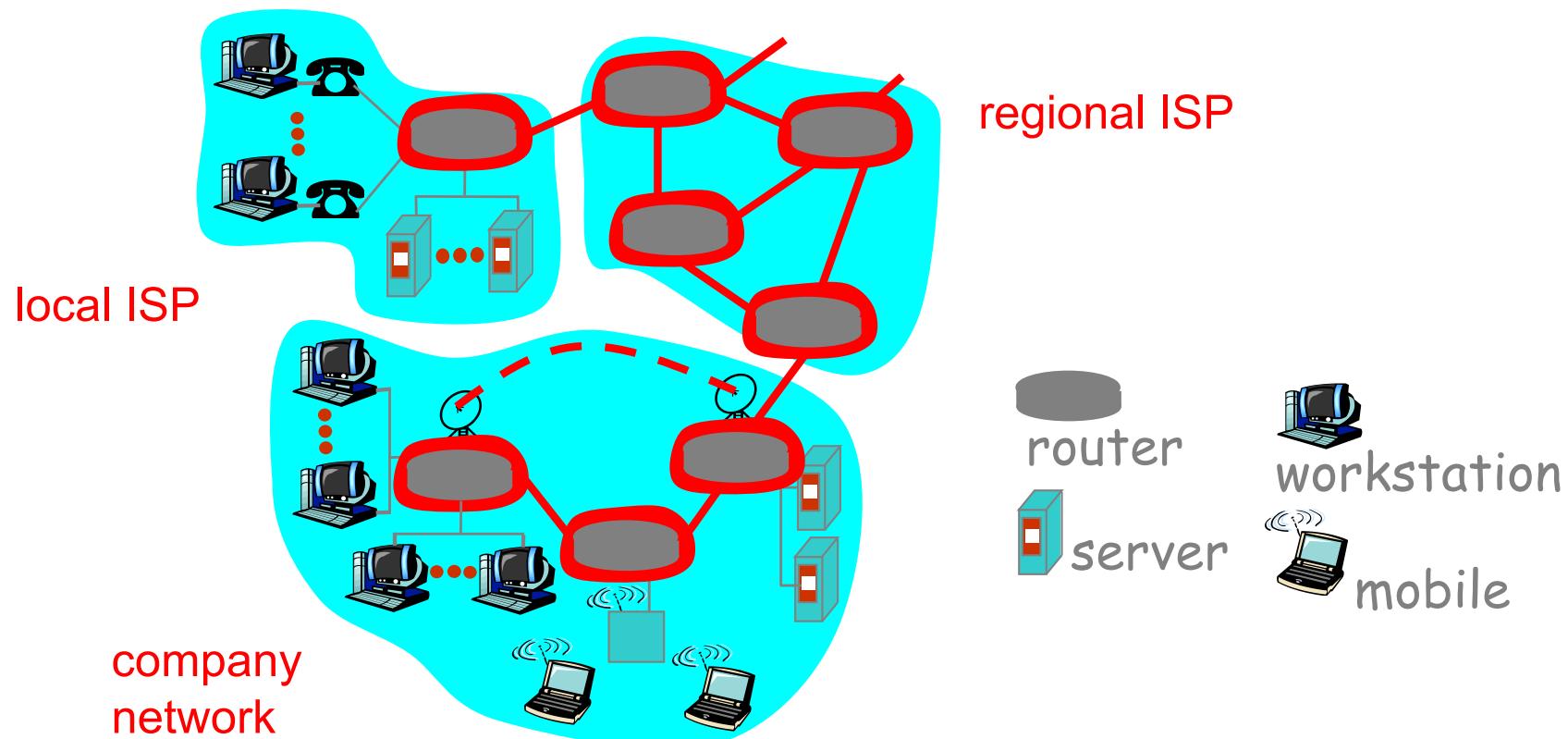
- Estado de Ligação (LS)
- Vectores de Distância (DV)
- Comparação entre DV e LS

## ● Protocolos de encaminhamento

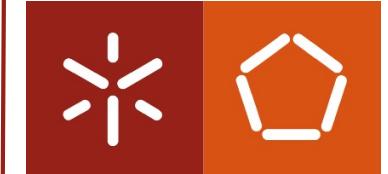
- Protocolos de encaminhamento interno (IGP)
- Protocolos de encaminhamento externo (EGP)



# Encaminhamento na Internet



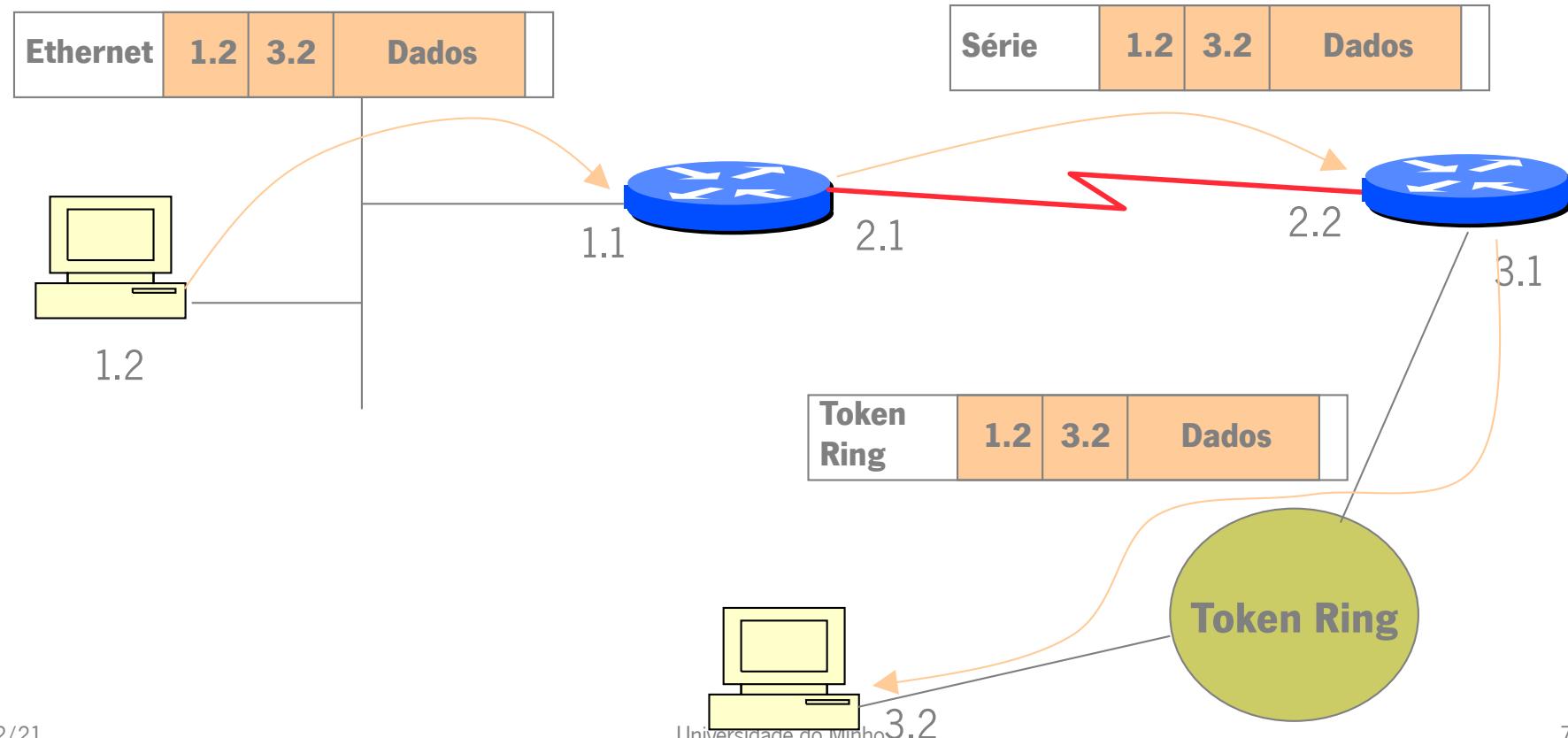
Fonte: Computer Networking: A Top-Down Approach  
Featuring the Internet, J. Kurose, Addison-Wesley

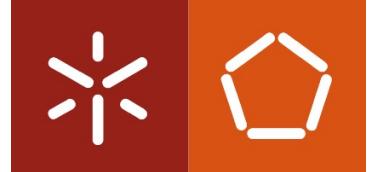


# Encaminhamento na Internet

- **Routers armazenam e reenviam datagramas IP**

- Desencapsula no interface de entrada, encapsula no interface de saída, de acordo com o tipo de interface...





# Encaminhamento na Internet

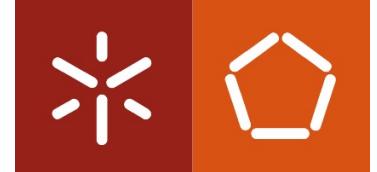
## ● Reenvio (forwarding) num encaminhador:

- Utiliza a tabela de reenvio previamente preenchida pelos protocolos de encaminhamento ou pelo administrador
- Procura na tabela, para um dado “destino”, o “próximo salto” e o “interface de saída”
- Comuta o pacote pelo interface respectivo, encapsulando-o numa trama de acordo com o tipo de interface

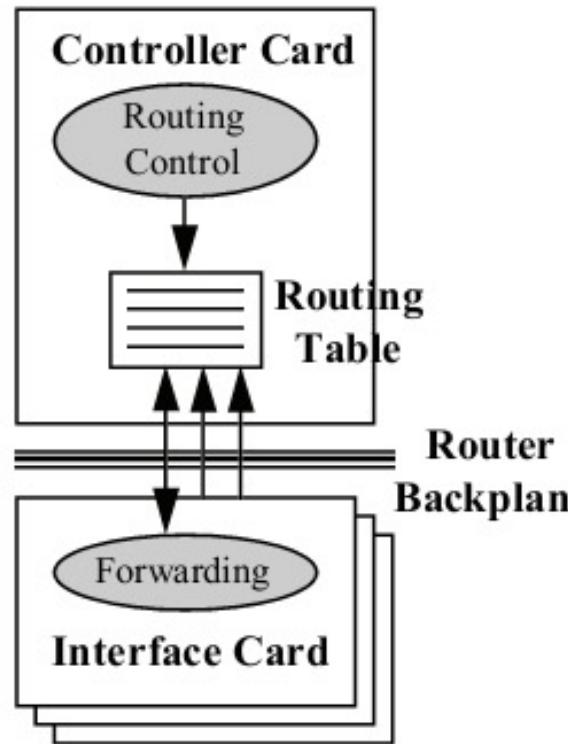
## ● Encaminhamento (routing):

- Preenche a tabela de encaminhamento com a(s) melhor(es) rotas para as redes de destino (*classfull*) ou para um conjunto de prefixos de endereços (*classless*)
- Pode ser um processo manual, feito pelo administrador – **encaminhamento estático**
- Ou um processo automático resultante da operação de um **protocolo de encaminhamento**, no caso mais comum de **encaminhamento dinâmico**

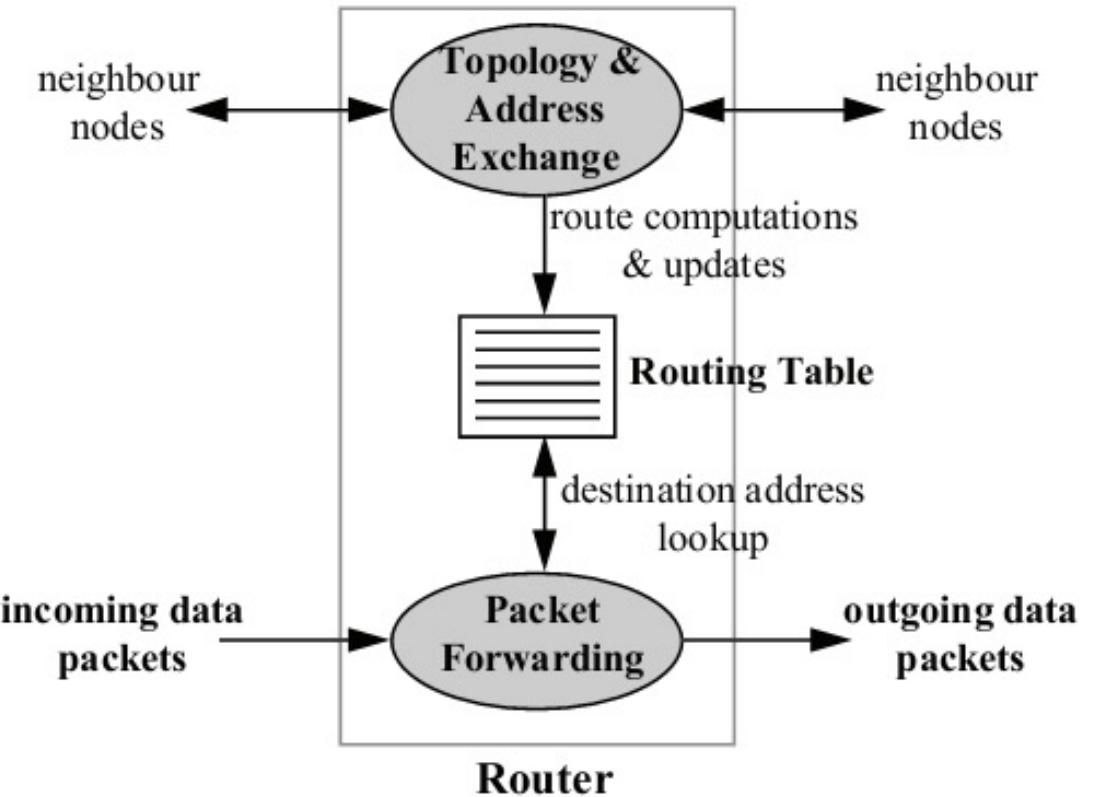
# Routers – Arquitectura Genérica



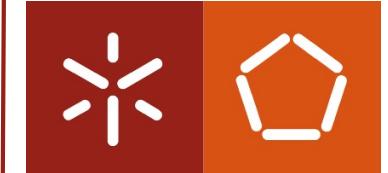
**Plano de Control**  
(Encaminhamento)



a) Arquitectura básica

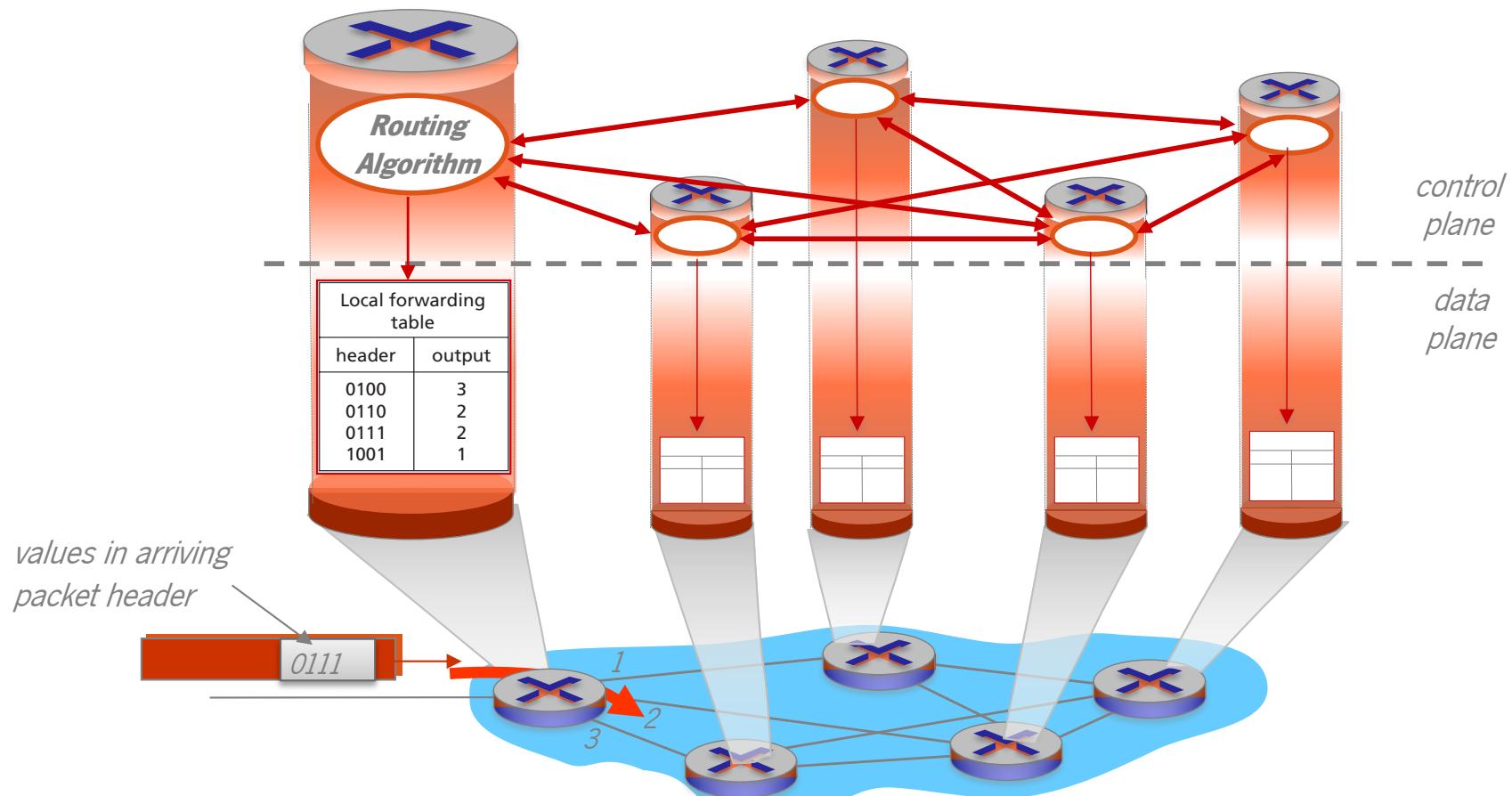


b) Componentes de routing

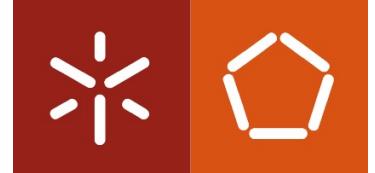


# Routers – *Plano de Controlo* por router

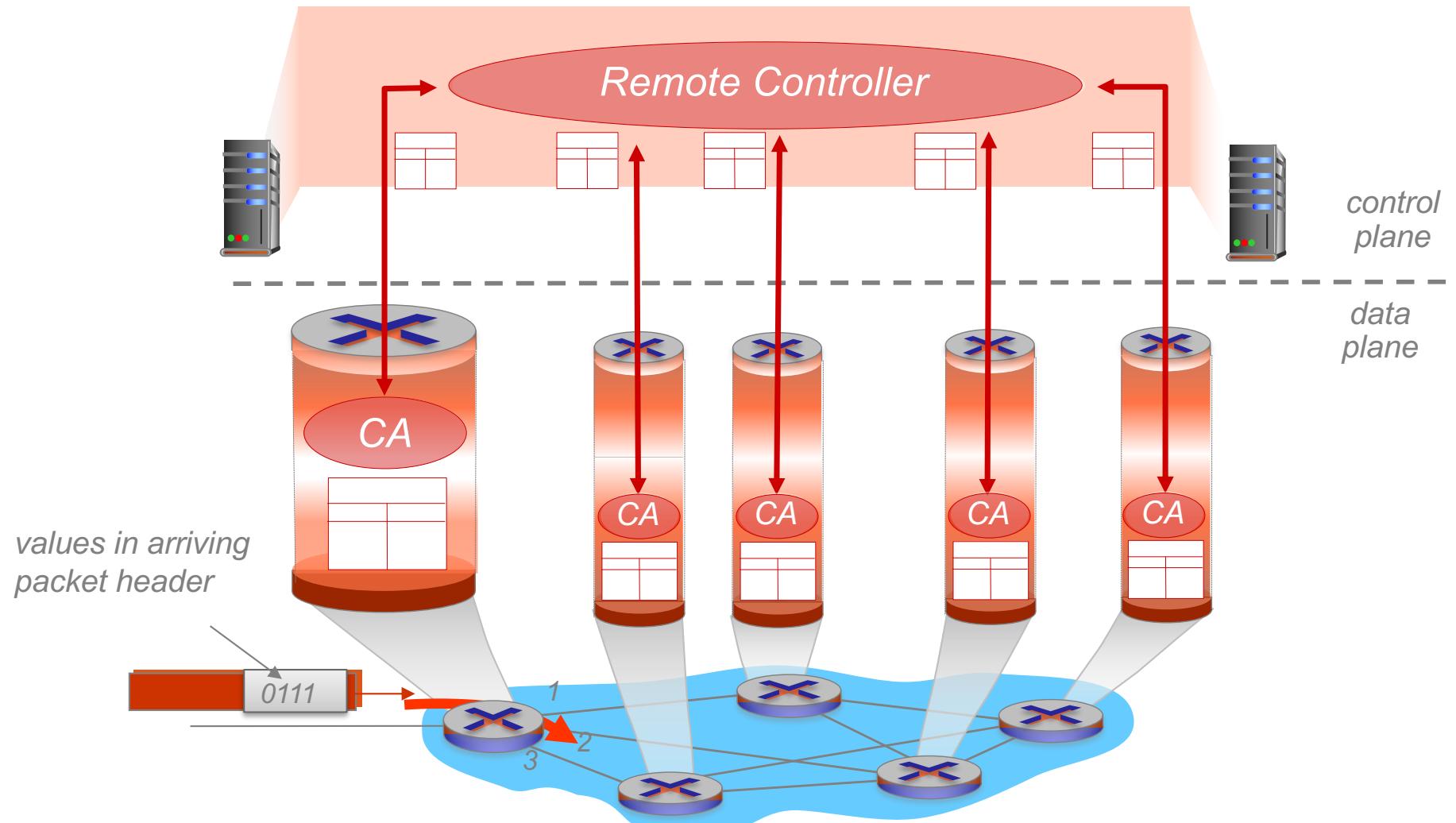
- Uma componente de controlo em **todos os routers** (algoritmos de routing distribuídos)



# Routers – *Plano de Controlo* centralizado

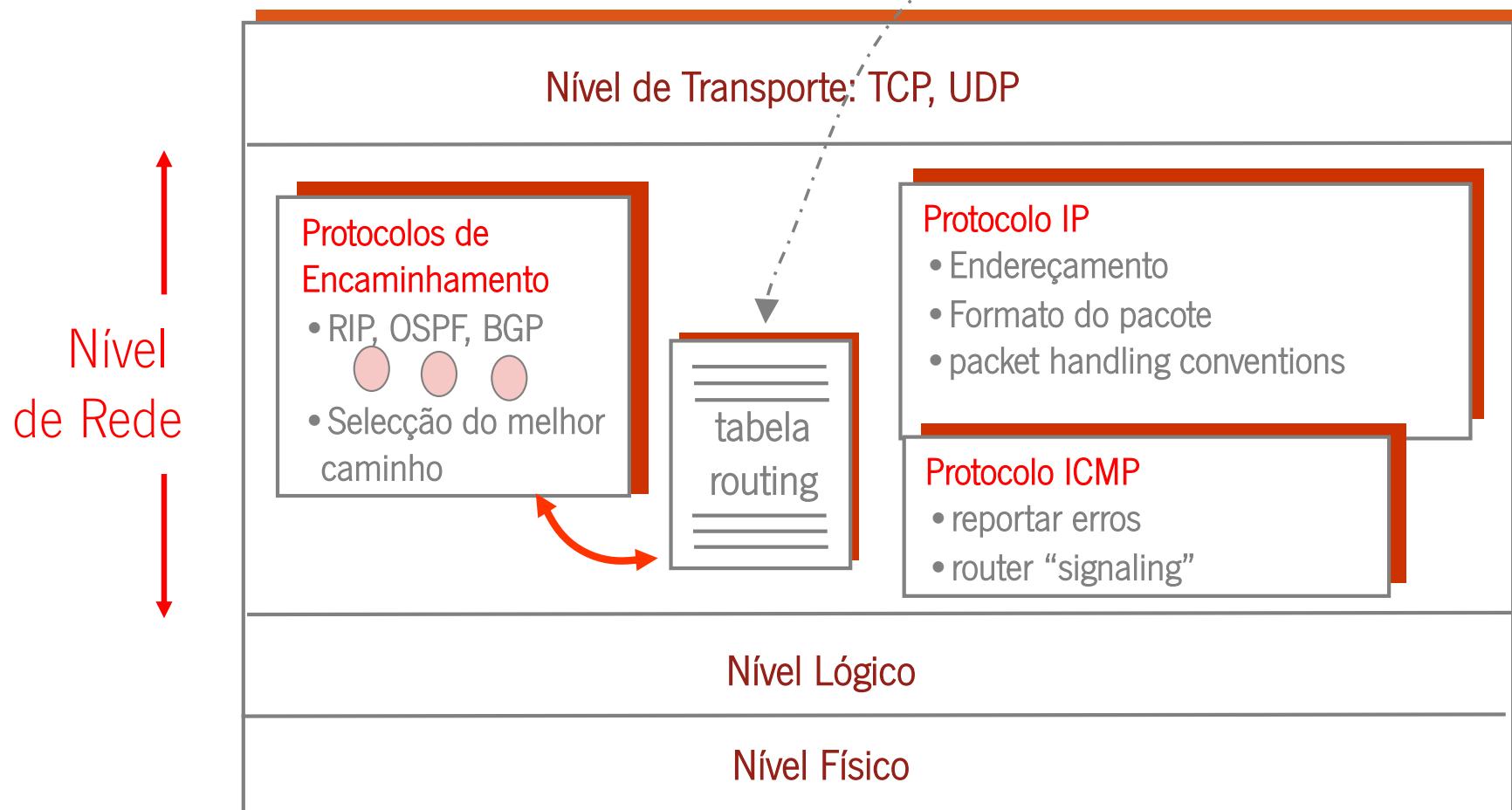


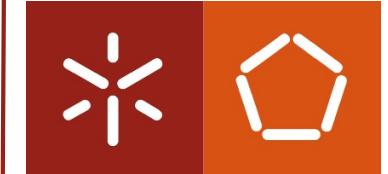
- Um controlador remoto que interage com agentes locais (CA)<sup>1</sup> (algoritmos de routing centralizados)



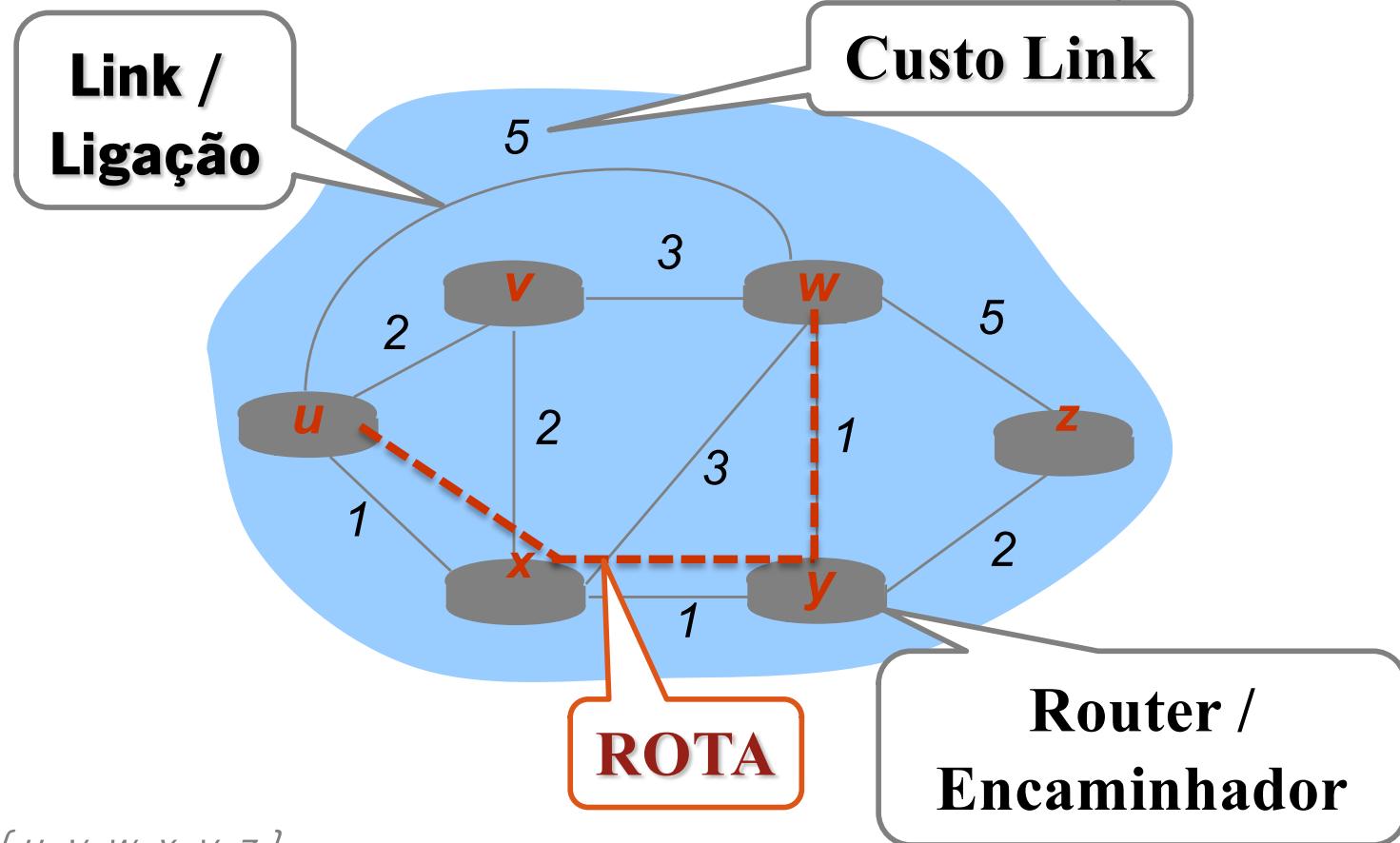
# Encaminhamento na Internet

ADMIIN: Rotas Estáticas!





# Abstracção: a rede como um grafo



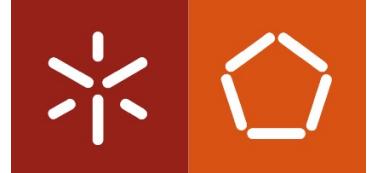
graph:  $G = (N, E)$

$N = \text{set of routers} = \{ u, v, w, x, y, z \}$

$E = \text{set of links} = \{ (u, v), (u, x), (v, x), (v, w), (x, w), (x, y), (w, y), (w, z), (y, z) \}$

Nota: Esta abstracção é útil noutros contextos de rede

Exemplo: P2P, onde N é o conjunto de *peers* e E o conjunto de conexões TCP



# Encaminhamento em redes IP

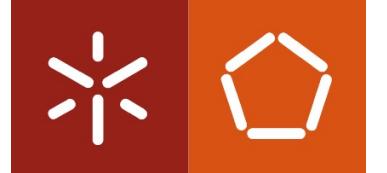
## Algoritmo Encaminhamento

Objectivo: dado um conjunto de encaminhadores com ligações de rede a interligá-los o objectivo do algoritmo de encaminhamento é determinar um “bom” caminho desde a fonte até ao destino

### Abstracção:

- **A topologia de rede é um Grafo:**

- Os nós do grafo são os encaminhadores
- Os arcos do grafo são as ligações da rede
- O custo das ligações pode ser estabelecido em função do atraso, da capacidade, do nível congestão, do custo, da distância, etc
- Um “bom” caminho significa tipicamente o caminho de “custo mínimo”, mas há outras possibilidades...



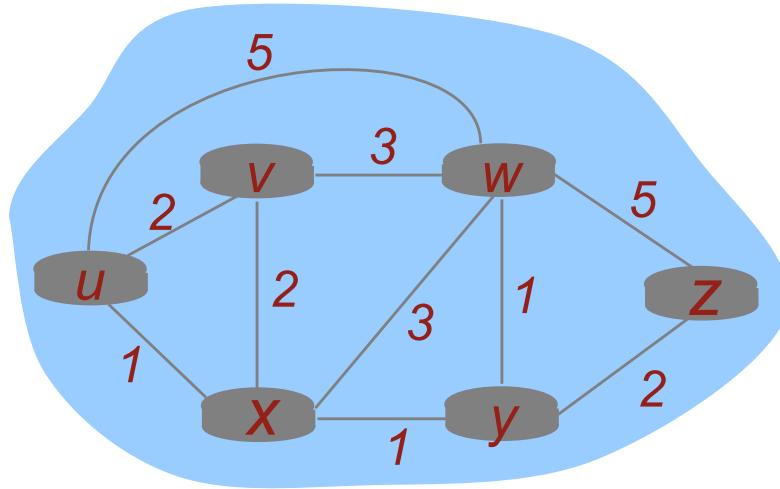
# Tabela de Encaminhamento

Nó  $u$

	Destino	Próximo Nó	Link	Custo
Nó $u$				
	$U$	$U$		$0$
	...	...	---	...
	$X$	$X$	$L_{UX}$	$1$
	...	...	---	...
	$W$	$X$	$L_{UX}$	$3$



# Abstracção: a rede como um grafo

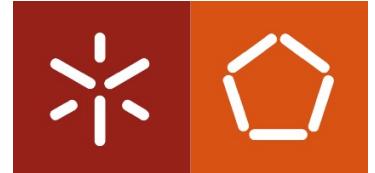


- $c(x,x') = \text{custo do link } (x,x')$ 
  - exemplo  **$c(u,w) = 5$**
- Custo poderia ser sempre 1, inversamente proporcional à largura de banda ou inversamente proporcional à congestão

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

Qual o caminho de custo mínimo entre u e z?

Algoritmos de encaminhamentos: procuram caminhos de custo mínimo



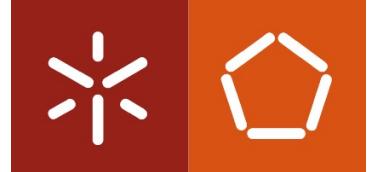
### Informação Global ou Descentralizada?

#### Global:

- Todos os encaminhadores têm um conhecimento completo da topologia e custo das ligações
- Algoritmos de estado das ligações (LS-“link state”)

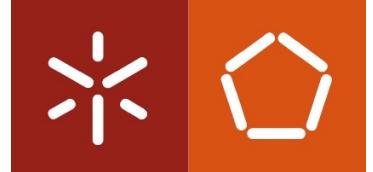
#### Descentralizada:

- Os encaminhadores só conhecem os vizinhos (fisicamente ligados) e o custo das ligações respectivas
- processo de computação é iterativo, troca de informação com os vizinhos
- Algoritmos de vector de distância (DV-“distance vector”)



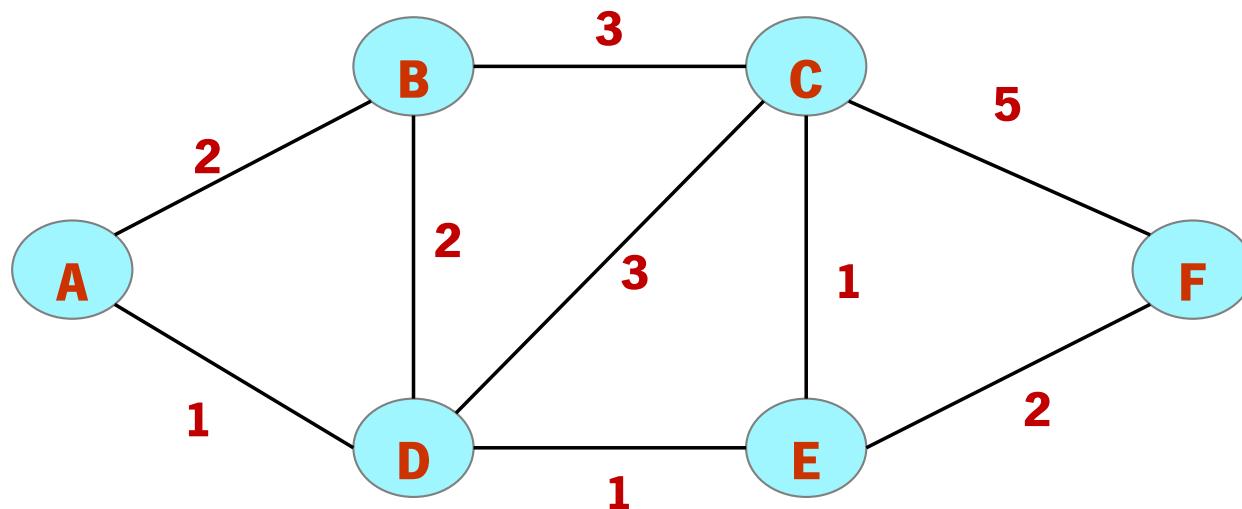
# Algoritmos do Estado das Ligações (LSA)

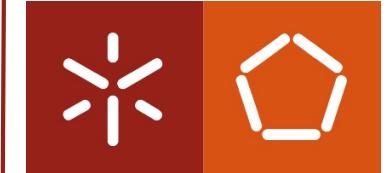
- 1. Todos os nós da topologia espalham pela rede o “estado das suas ligações” de forma a construírem a “base de dados topológica” – (usando inundação fiável - ***Reliable Flooding***)**
  - Inicialmente necessitam de conhecer apenas os seus vizinhos diretos, para quem enviam a identificação de todos os seus vizinhos bem como o custo das ligações que os separam deles
  - Um encaminhador ao receber esta informação atualiza a sua base de dados topológica e reenvia a informação para todos os seus vizinhos
  - Ao fim de algum tempo todos os nós possuem um conhecimento completo da topologia e dos custos de todas as ligações
- 2. Sobre esta informação, em cada encaminhador, é utilizado um algoritmo de descoberta dos caminhos de custo mínimo, tipicamente o algoritmo de Dijkstra.**
- 3. Com o resultado obtido da aplicação do algoritmo de Dijkstra, é preenchida a tabela de encaminhamento.**



# Exercício

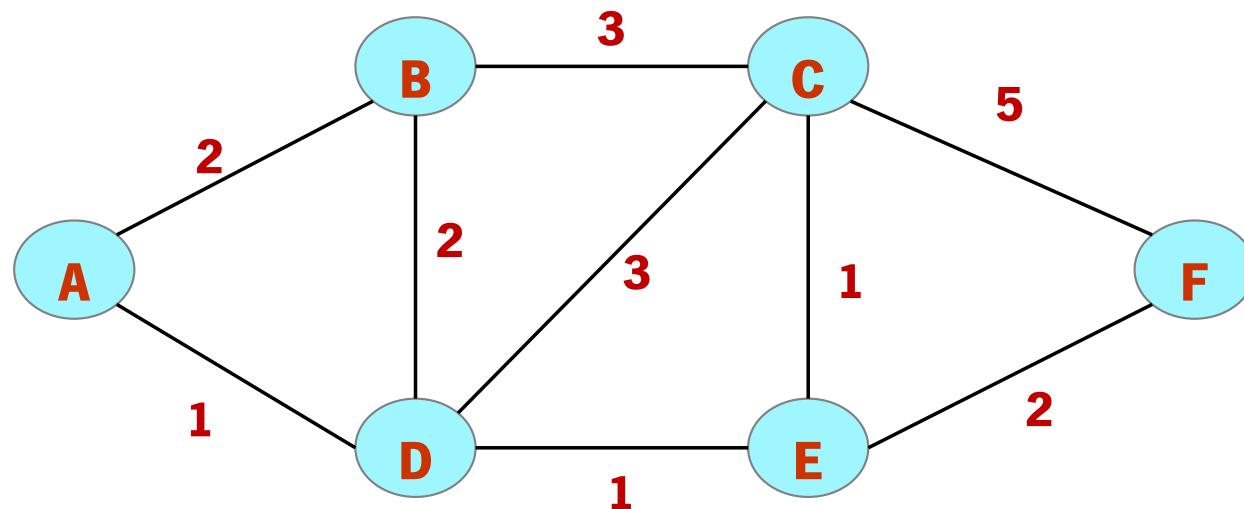
- Usando um algoritmo de estado da ligação, qual a visão topológica da rede que tem o nó A no final da primeira iteração de troca de LSAs?
- Seria correcto calcular as rotas nesta fase? E no final da segunda iteração?



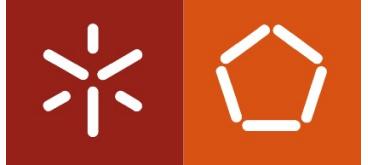


# Exercício

1. Cada nó gera uma mensagem LSA com o estado dos seus links



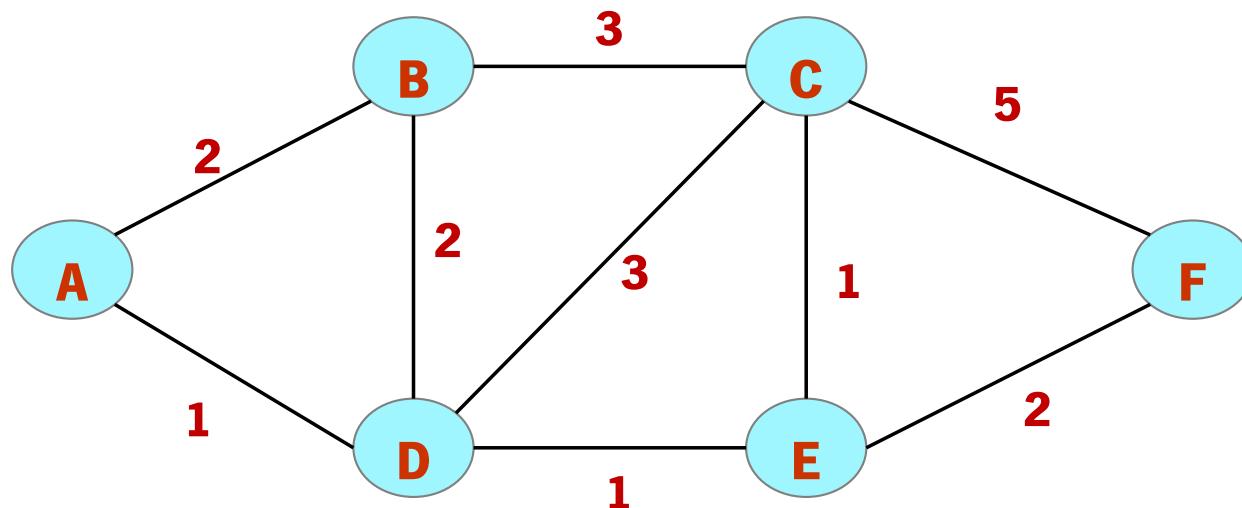
# Exercício

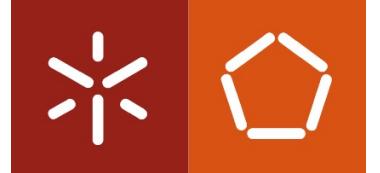


1. Cada nó gera uma mensagem LSA com o estado dos seus links

$$\text{LSA}(A) = \{ (A \rightarrow B, 2); (A \rightarrow D, 1) \}, \dots, \text{LSA}(F) = \{ (F \rightarrow C, 5); (F \rightarrow E, 2) \}$$

Grafo da rede = { todos os LSAs de todos os nós }



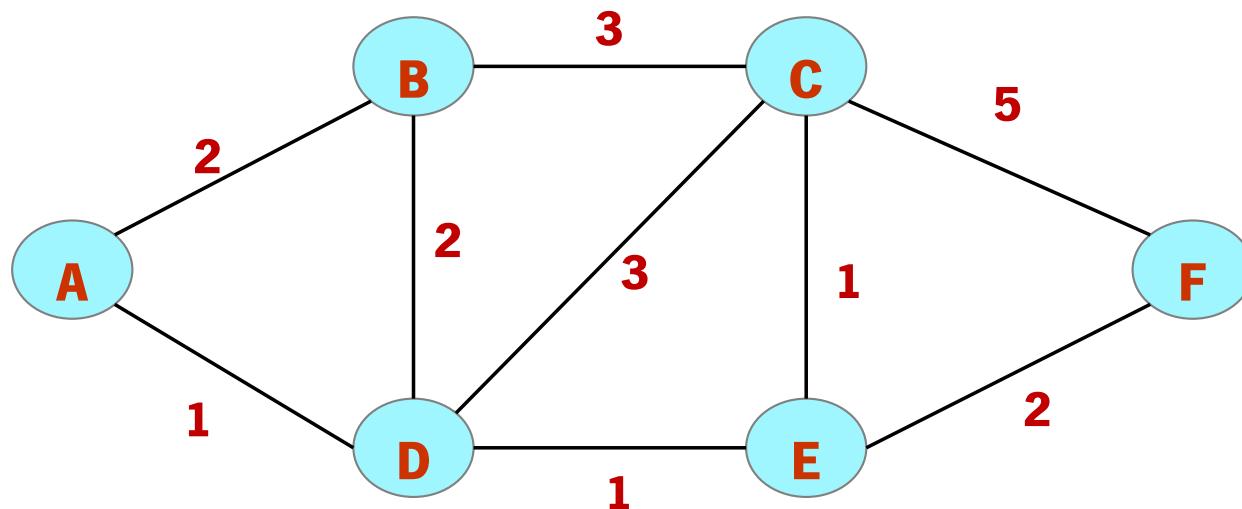


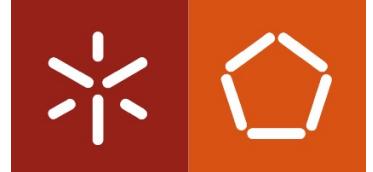
# Exercício

1. Cada nó gera uma mensagem LSA com o estado dos seus links

$$\text{LSA(A)} = \{ (\text{A} \rightarrow \text{B}, 2); (\text{A} \rightarrow \text{D}, 1) \}, \dots, \text{LSA(F)} = \{ (\text{F} \rightarrow \text{C}, 5); (\text{F} \rightarrow \text{E}, 2) \}$$

2. Cada nó tem de enviar a sua msg LSA a todos os outros (Flooding)





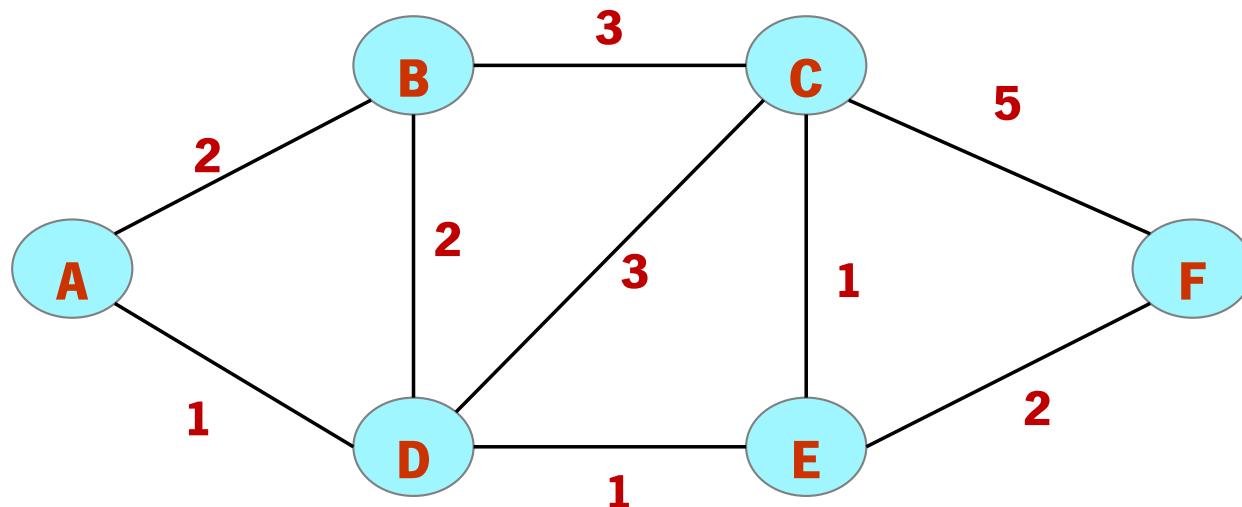
# Exercício

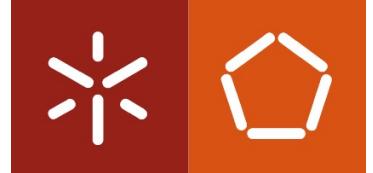
1. Cada nó gera uma mensagem LSA com o estado dos seus links

$$\text{LSA}(A) = \{ (A \rightarrow B, 2); (A \rightarrow D, 1) \}, \dots, \text{LSA}(F) = \{ (F \rightarrow C, 5); (F \rightarrow E, 2) \}$$

2. Cada nó tem de enviar a sua msg LSA a todos os outros (Flooding)

Exemplo só para as mensagens enviadas pelo A





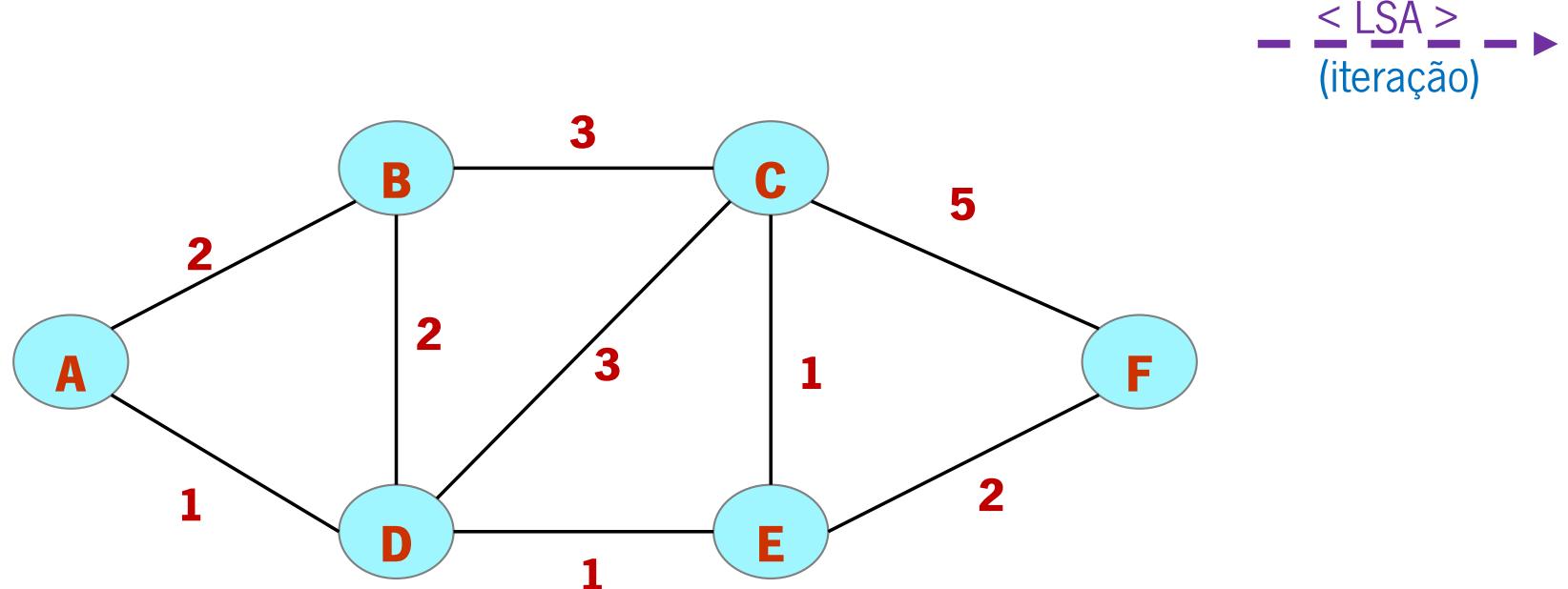
# Exercício

1. Cada nó gera uma mensagem LSA com o estado dos seus links

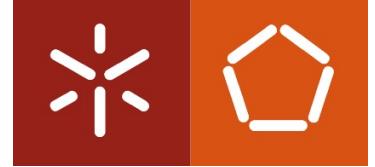
$$\text{LSA}(A) = \{ (A \rightarrow B, 2); (A \rightarrow D, 1) \}, \dots, \text{LSA}(F) = \{ (F \rightarrow C, 5); (F \rightarrow E, 2) \}$$

2. Cada nó tem de enviar a sua msg LSA a todos os outros (Flooding)

Exemplo só para as mensagens enviadas pelo A



# Exercício

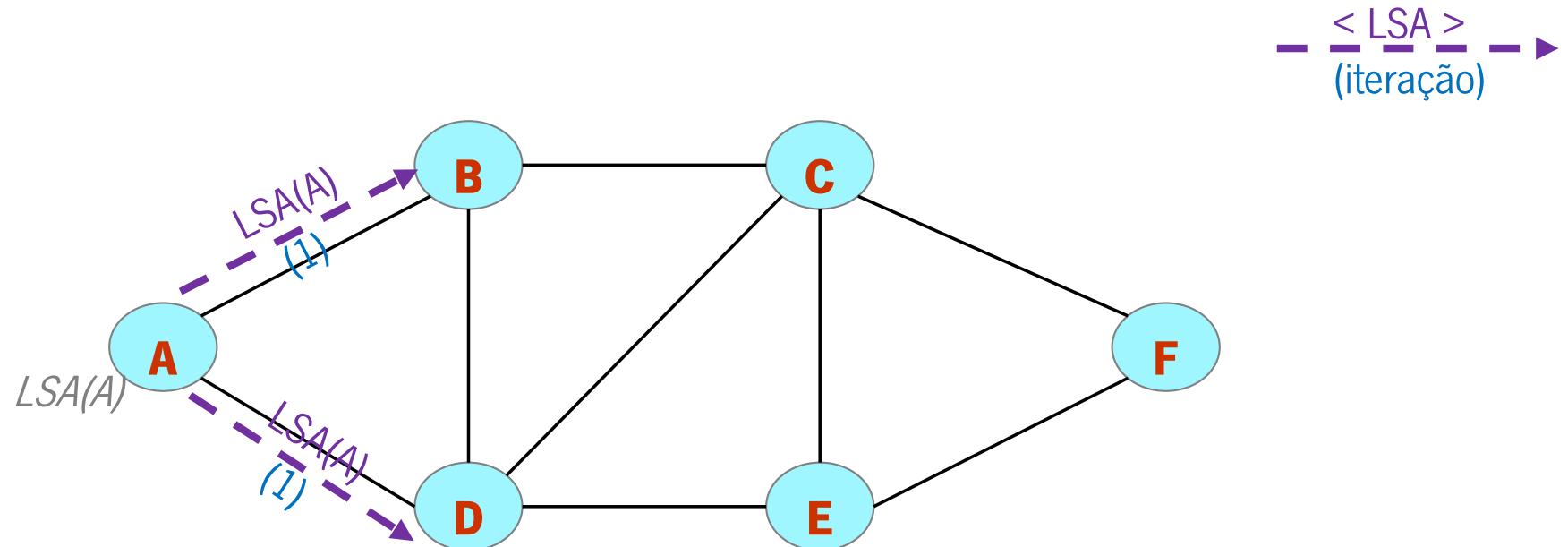


1. Cada nó gera uma mensagem LSA com o estado dos seus links

$$\text{LSA}(A) = \{ (A \rightarrow B, 2); (A \rightarrow D, 1) \}, \dots, \text{LSA}(F) = \{ (F \rightarrow C, 5); (F \rightarrow E, 2) \}$$

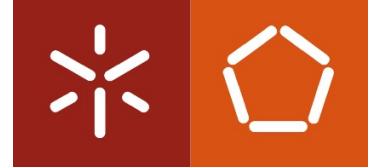
2. Cada nó tem de enviar a sua msg LSA a todos os outros (Flooding)

Exemplo só para as mensagens enviadas pelo A



-→ O nó A vai enviar a todos os vizinhos...

# Exercício

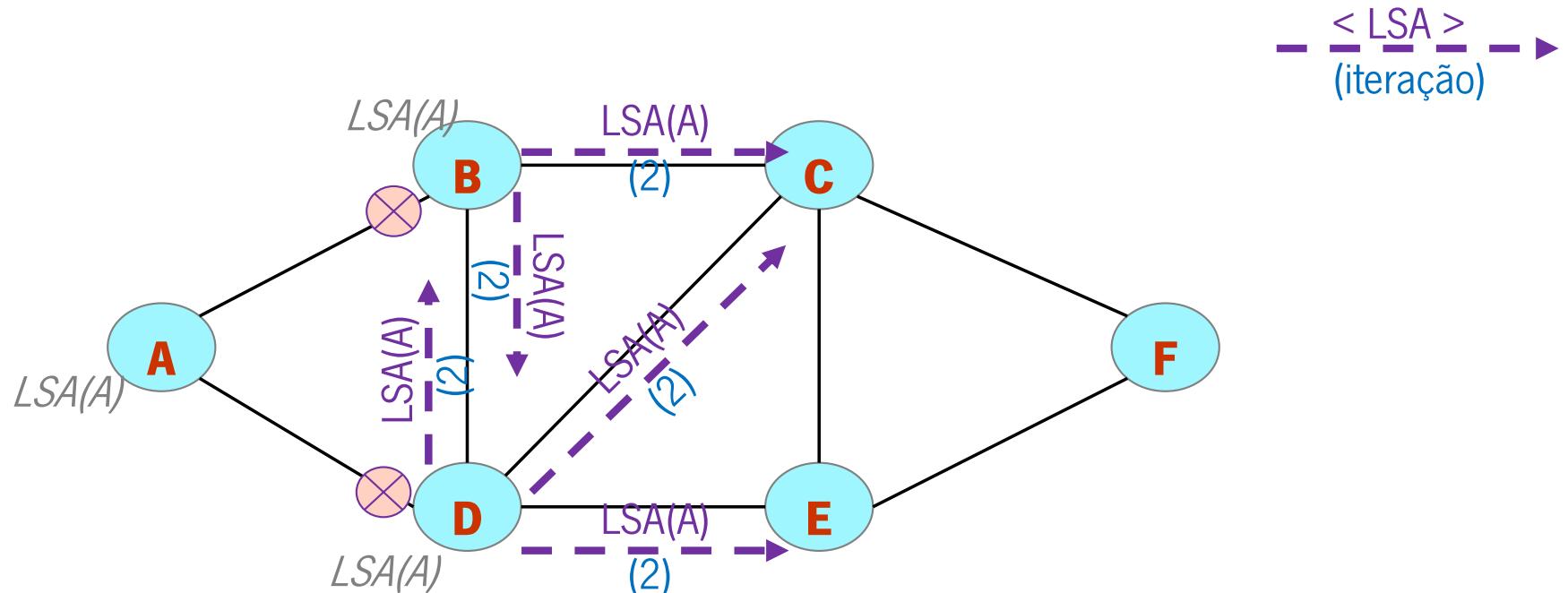


1. Cada nó gera uma mensagem LSA com o estado dos seus links

$$\text{LSA}(A) = \{ (A \rightarrow B, 2); (A \rightarrow D, 1) \}, \dots, \text{LSA}(F) = \{ (F \rightarrow C, 5); (F \rightarrow E, 2) \}$$

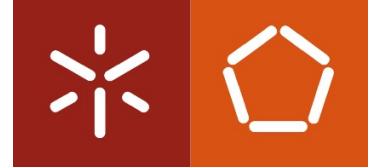
2. Cada nó tem de enviar a sua msg LSA a todos os outros (Flooding)

Exemplo só para as mensagens enviadas pelo A



Os vizinhos reenviam também a todos, exceto de onde receberam! (RPF)

# Exercício

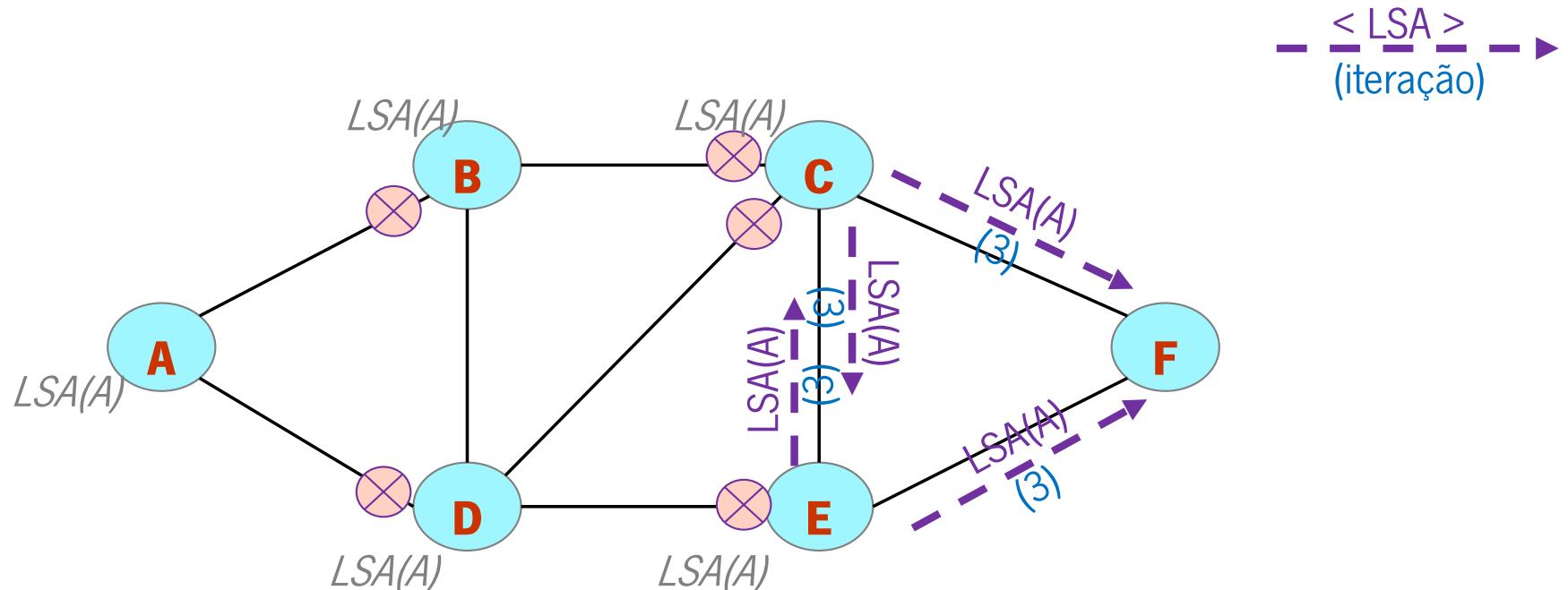


1. Cada nó gera uma mensagem LSA com o estado dos seus links

$$\text{LSA}(A) = \{ (A \rightarrow B, 2); (A \rightarrow D, 1) \}, \dots, \text{LSA}(F) = \{ (F \rightarrow C, 5); (F \rightarrow E, 2) \}$$

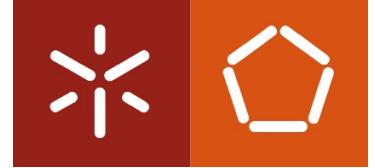
2. Cada nó tem de enviar a sua msg LSA a todos os outros (Flooding)

Exemplo só para as mensagens enviadas pelo A



Os vizinhos reenviam também a todos, exceto de onde receberam! (RPF)

# Exercício

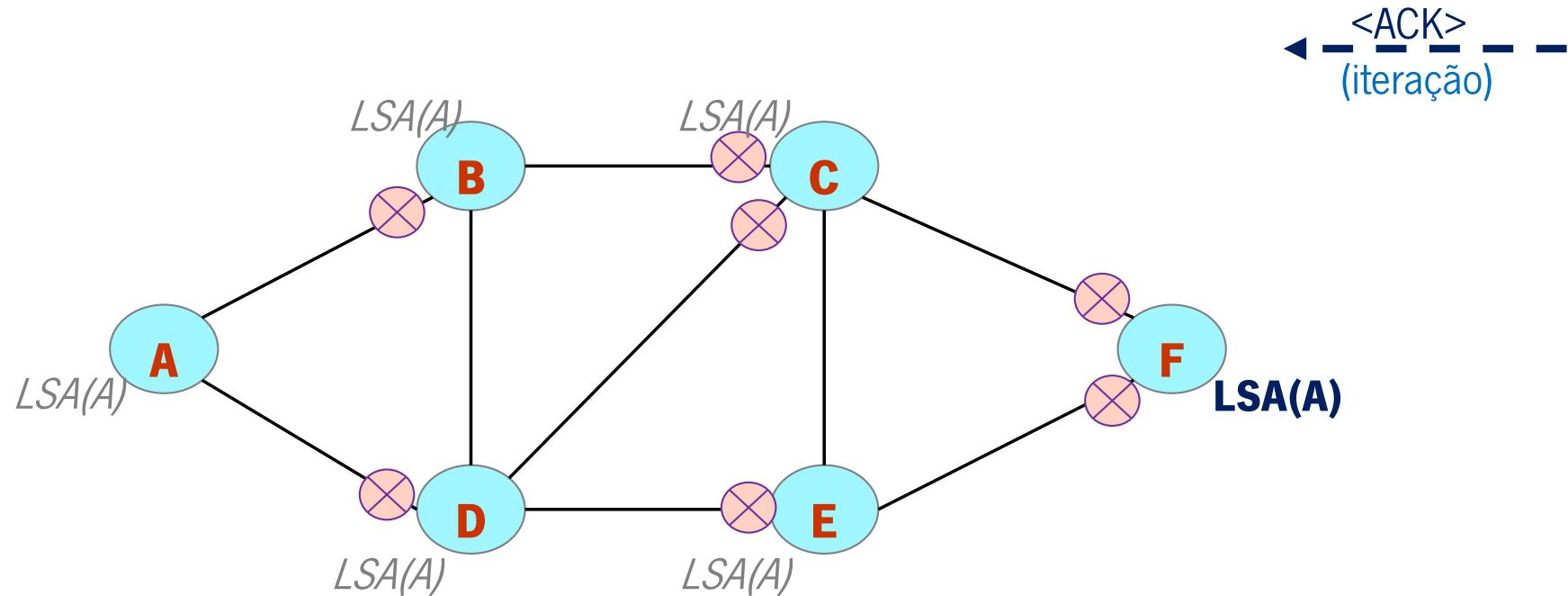


1. Cada nó gera uma mensagem LSA com o estado dos seus links

$$\text{LSA}(A) = \{ (A \rightarrow B, 2); (A \rightarrow D, 1) \}, \dots, \text{LSA}(F) = \{ (F \rightarrow C, 5); (F \rightarrow E, 2) \}$$

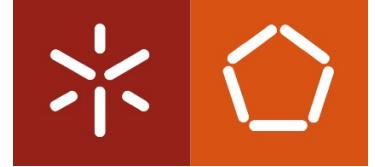
2. Cada nó tem de enviar a sua msg LSA a todos os outros (Flooding)

Exemplo só para as mensagens enviadas pelo A



3. A receção tem de ser confirmada de volta até ao A (certeza de que chegou)

# Exercício

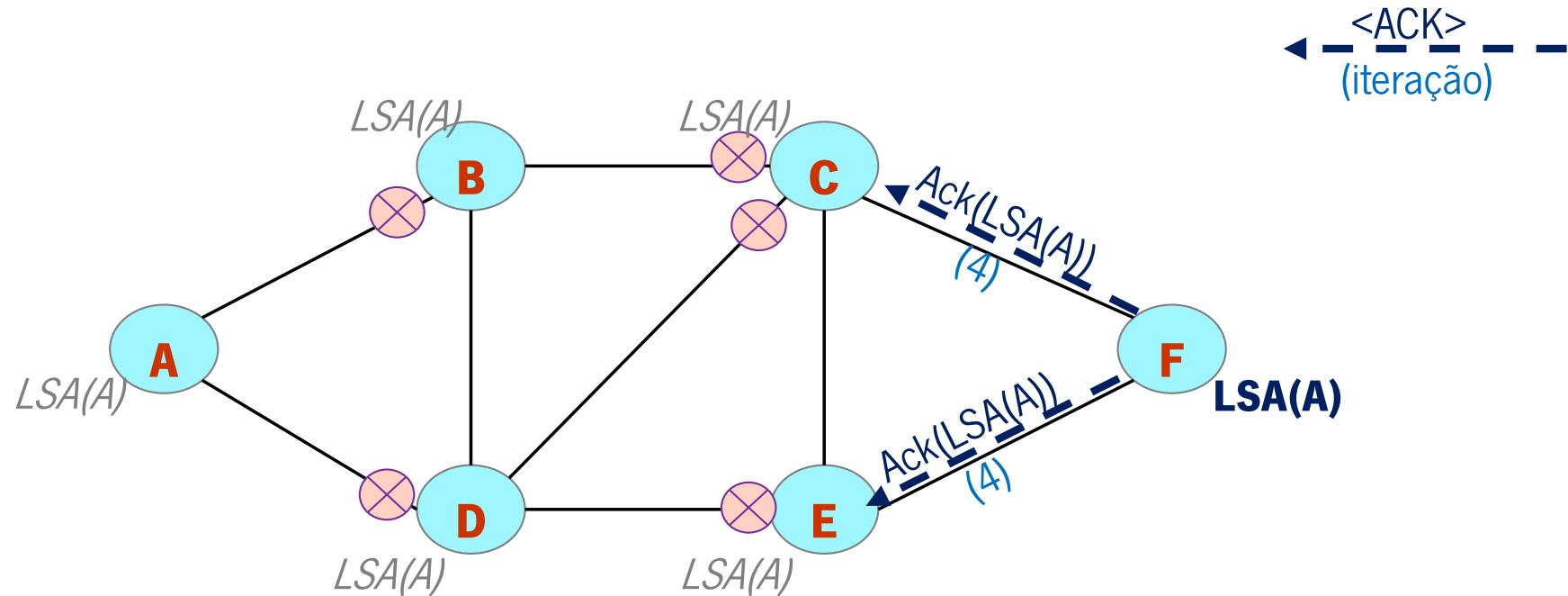


1. Cada nó gera uma mensagem LSA com o estado dos seus links

$$\text{LSA}(A) = \{ (A \rightarrow B, 2); (A \rightarrow D, 1) \}, \dots, \text{LSA}(F) = \{ (F \rightarrow C, 5); (F \rightarrow E, 2) \}$$

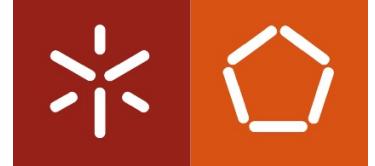
2. Cada nó tem de enviar a sua msg LSA a todos os outros (Flooding)

Exemplo só para as mensagens enviadas pelo A



3. A receção tem de ser confirmada de volta até ao A (certeza de que chegou)

# Exercício

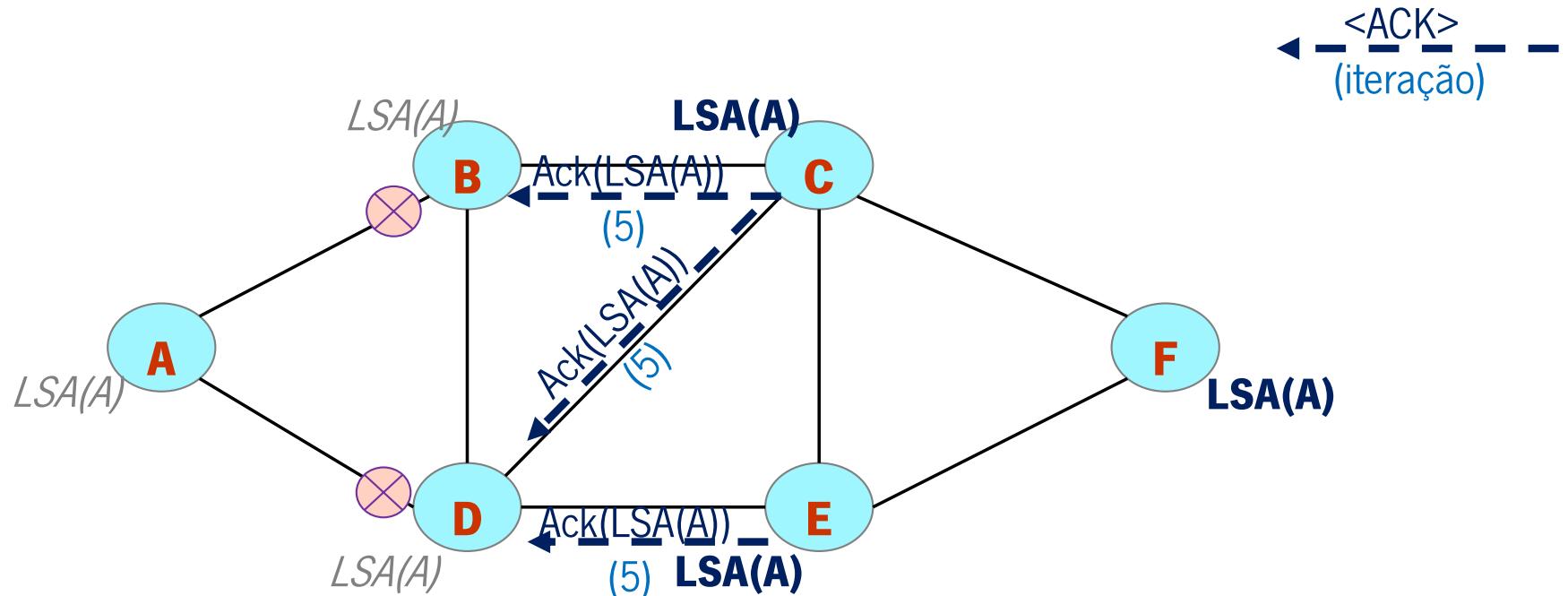


1. Cada nó gera uma mensagem LSA com o estado dos seus links

$$\text{LSA}(A) = \{ (A \rightarrow B, 2); (A \rightarrow D, 1) \}, \dots, \text{LSA}(F) = \{ (F \rightarrow C, 5); (F \rightarrow E, 2) \}$$

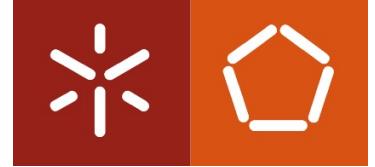
2. Cada nó tem de enviar a sua msg LSA a todos os outros (Flooding)

Exemplo só para as mensagens enviadas pelo A



3. A receção tem de ser confirmada de volta até ao A (certeza de que chegou)

# Exercício

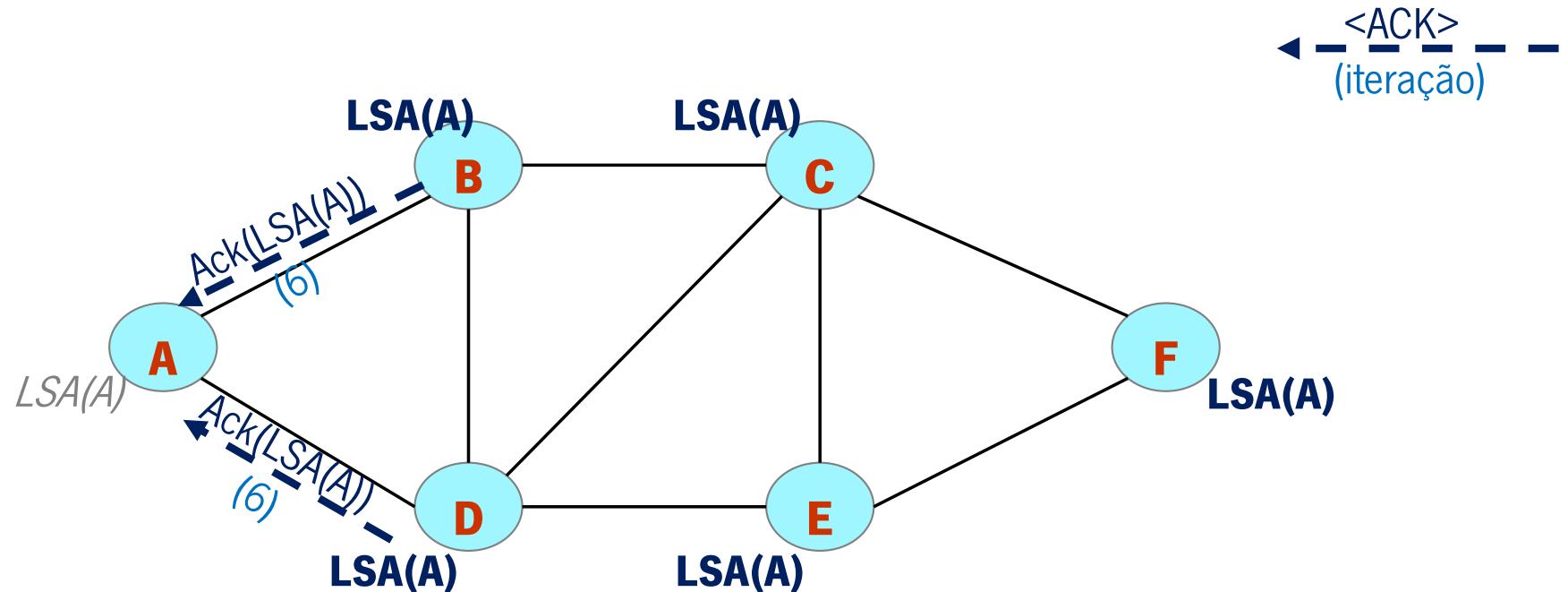


1. Cada nó gera uma mensagem LSA com o estado dos seus links

$$\text{LSA}(A) = \{ (A \rightarrow B, 2); (A \rightarrow D, 1) \}, \dots, \text{LSA}(F) = \{ (F \rightarrow C, 5); (F \rightarrow E, 2) \}$$

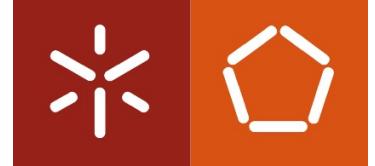
2. Cada nó tem de enviar a sua msg LSA a todos os outros (Flooding)

Exemplo só para as mensagens enviadas pelo A



3. A receção tem de ser confirmada de volta até ao A (certeza de que chegou)

# Exercício

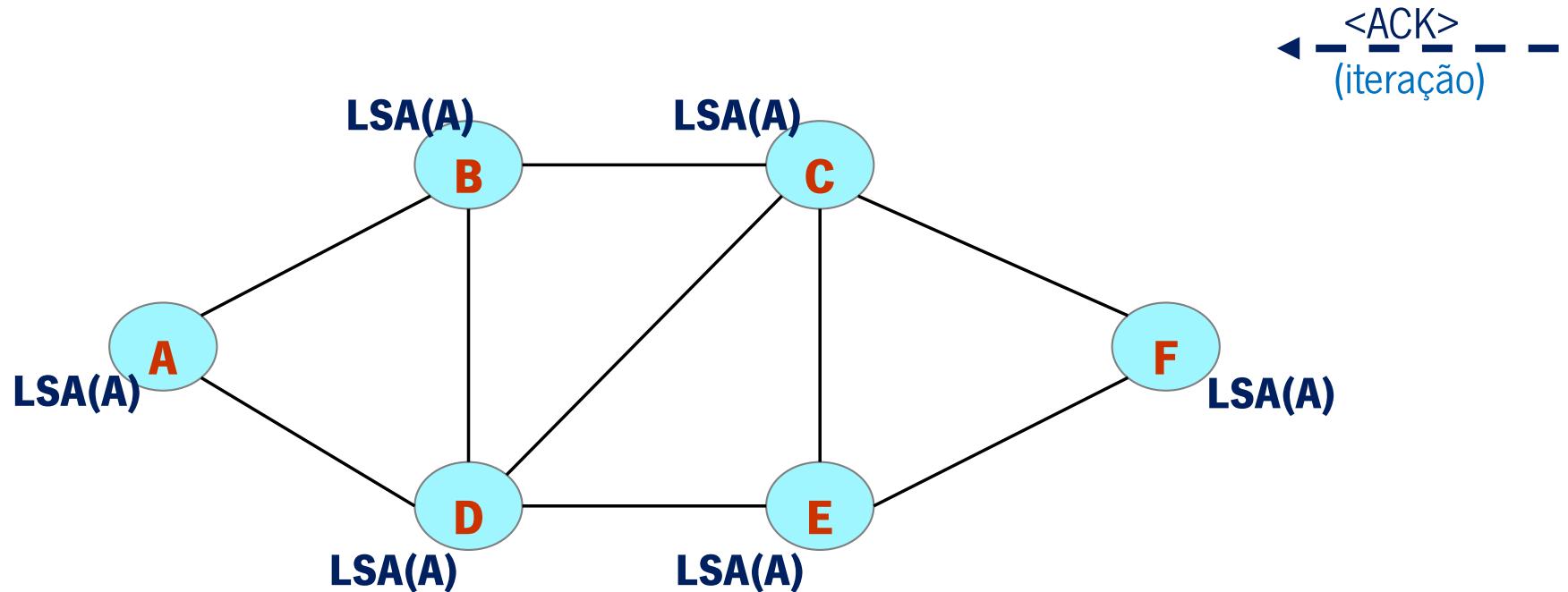


1. Cada nó gera uma mensagem LSA com o estado dos seus links

$$\text{LSA}(A) = \{ (A \rightarrow B, 2); (A \rightarrow D, 1) \}, \dots, \text{LSA}(F) = \{ (F \rightarrow C, 5); (F \rightarrow E, 2) \}$$

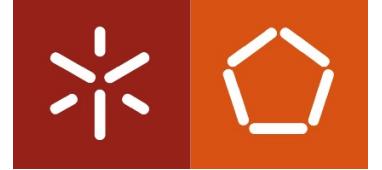
2. Cada nó tem de enviar a sua msg LSA a todos os outros (Flooding)

Exemplo só para as mensagens enviadas pelo A



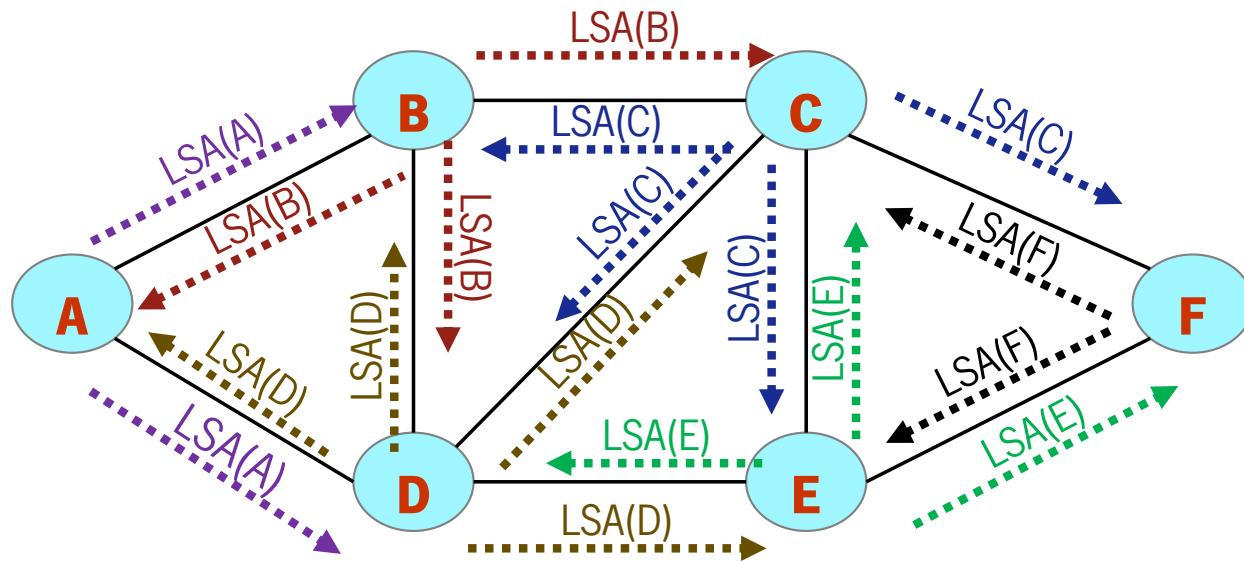
3. A receção tem de ser confirmada de volta até ao A (certeza de que chegou)

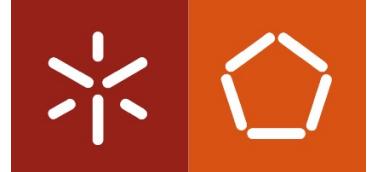
# Exercício



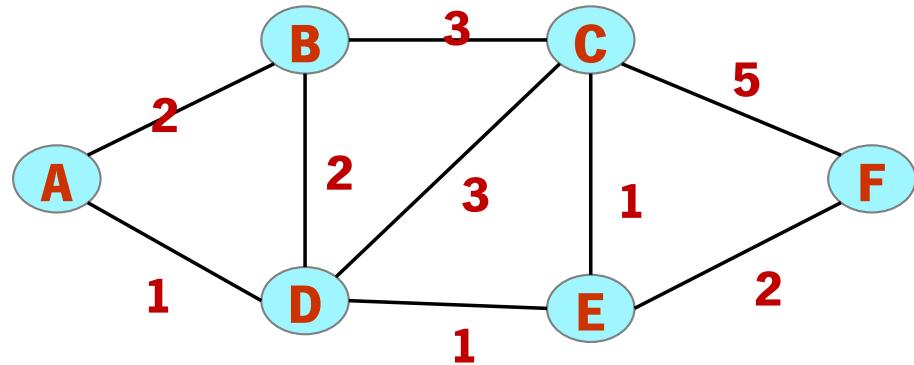
Conseguem imaginar todos a enviar ao mesmo tempo???

Sincronizado? (pouco provável!) Só na primeira iteração:

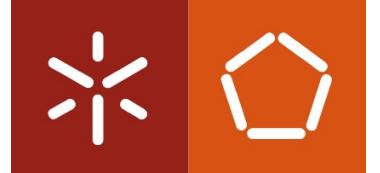




# Exercício

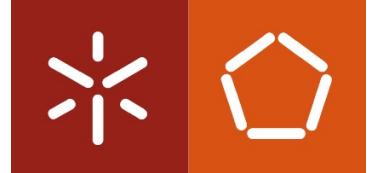


- Usando um algoritmo de estado da ligação, qual a visão topológica da rede que tem o nó A no final da primeira iteração de troca de LSAs?
- Seria correcto calcular as rotas nesta fase? E no final da segunda iteração?



# Algoritmo de Dijkstra

- **Algoritmo Iterativo que ao fim de k iterações consegue descobrir os caminhos de custo mínimo de um determinado nó para k destinos**
  - Seja  $c(i,j)$  o custo da ligação do nó  $i$  para o nó  $j$ . Se o nó  $i$  e o nó  $j$  não estão directamente ligados  $c(i,j)=\infty$ ;
  - Seja  $D(v)$  o custo do caminho desde o nó origem até ao nó  $v$
  - Seja  $p(v)$  o nó que antecede  $v$  no caminho desde o nó origem até ao nó  $v$
  - Seja  $N$  o conjunto de todos os nós para os quais já se conhece o caminho de custo mínimo



# Algoritmo de Dijkstra

1 **Initialization:**

2     $N' = \{u\}$

3    *for all nodes v*

4      *if v adjacent to u*

5        *then  $D(v) = c(u,v)$*

6        *else  $D(v) = \infty$*

7

8 **Loop**

9    *find w not in  $N'$  such that  $D(w)$  is a minimum*

10    *add w to  $N'$*

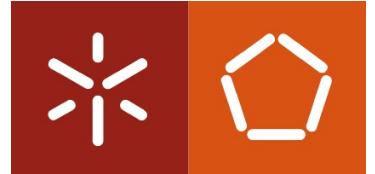
11    *update  $D(v)$  for all v adjacent to w and not in  $N'$ :*

12     **$D(v) = \min( D(v), D(w) + c(w,v) )$**

13    /\* new cost to v is either old cost to v or known

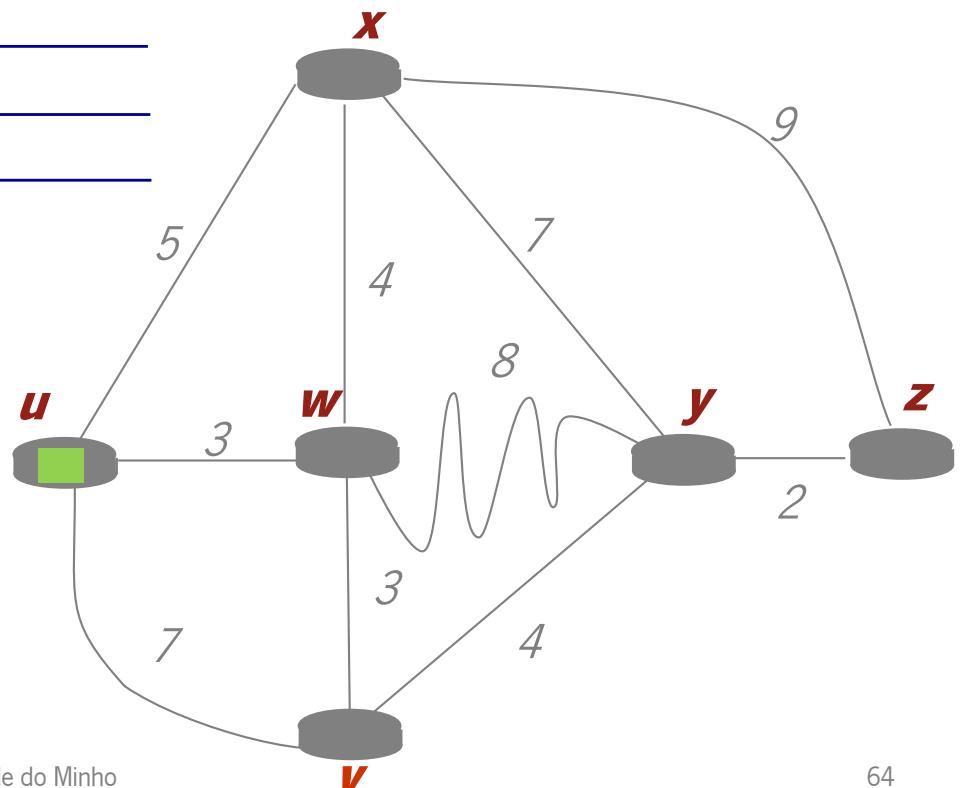
14    shortest path cost to w plus cost from w to v \*/

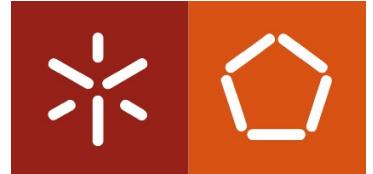
15 **until all nodes in  $N'$**



# LSA - Dijkstra: exemplo

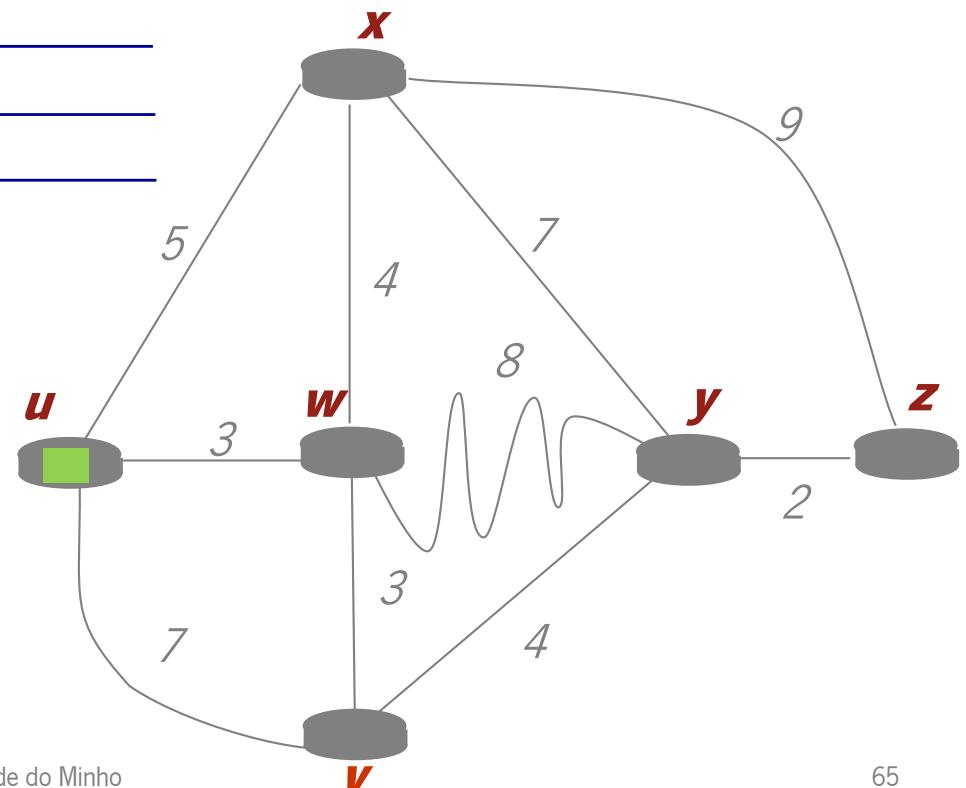
Step	$N'$	$D(v)$ $p(v)$	$D(w)$ $p(w)$	$D(x)$ $p(x)$	$D(y)$ $p(y)$	$D(z)$ $p(z)$
0	$u$					
1						
2						
3						
4						
5						
6						
7						
8						
9						

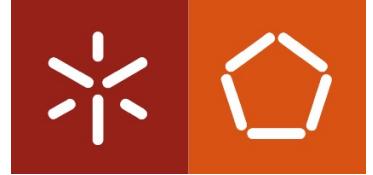




# LSA - Dijkstra: exemplo

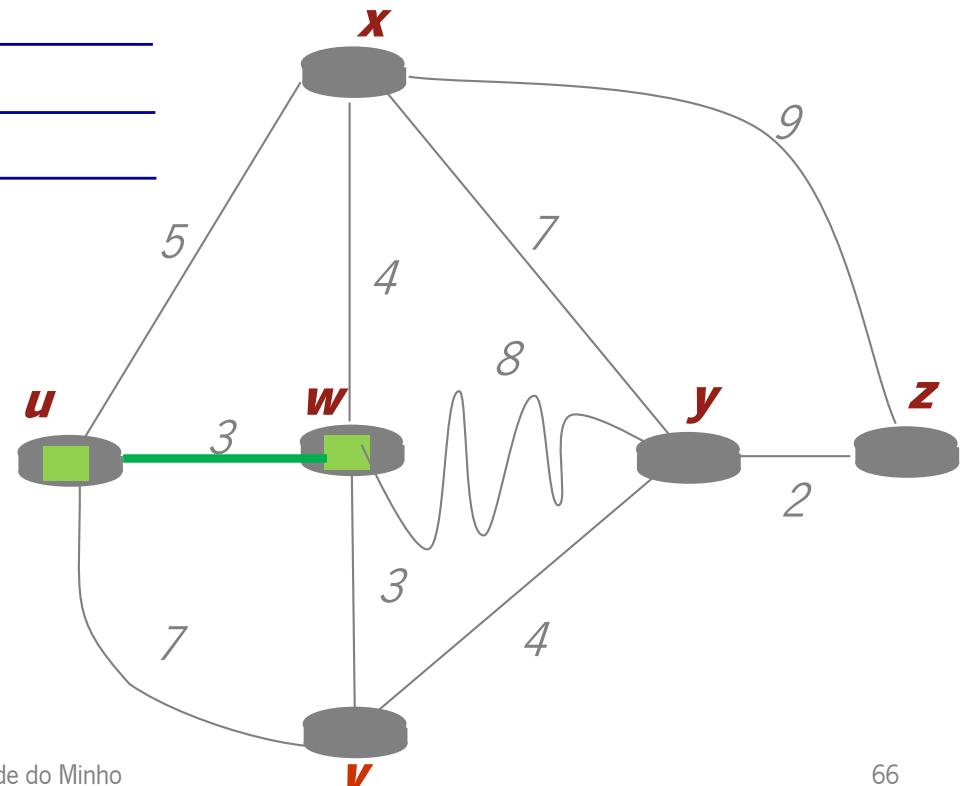
Step	$N'$	$D(v)$ $p(v)$	$D(w)$ $p(w)$	$D(x)$ $p(x)$	$D(y)$ $p(y)$	$D(z)$ $p(z)$
0	$u$	$7, u$	$3, u$	$5, u$	$\infty$	$\infty$

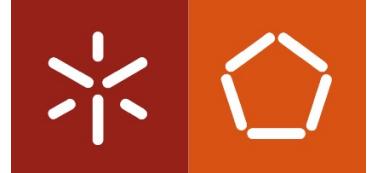




# LSA - Dijkstra: exemplo

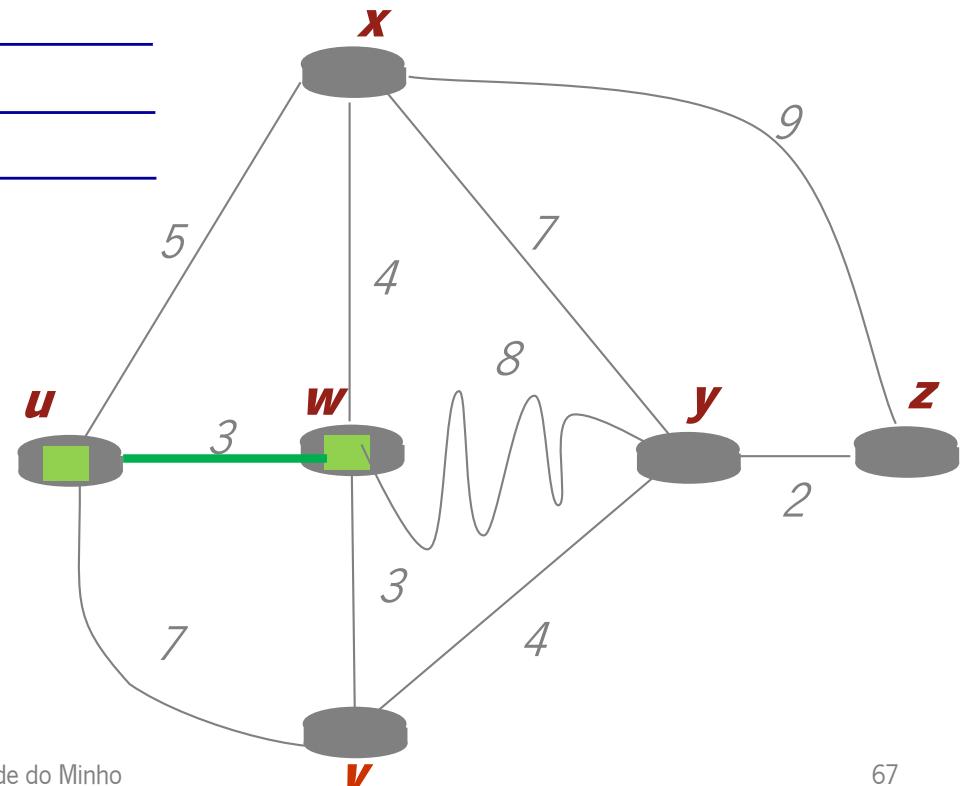
<i>Step</i>	$N'$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
		$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
0	$u$	$7, u$	$3, u$	$5, u$	$\infty$	$\infty$
1	$uw$					

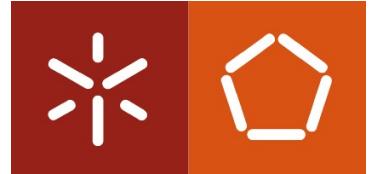




# LSA - Dijkstra: exemplo

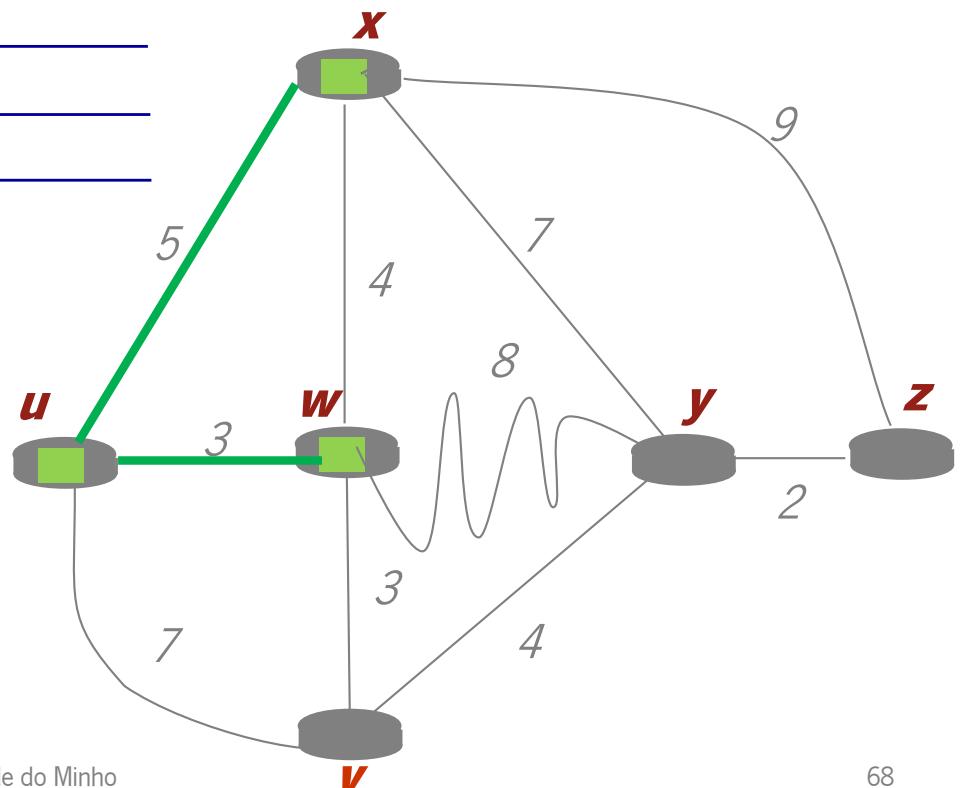
<i>Step</i>	$N'$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
		$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
0	$u$	$7, u$	$3, u$	$5, u$	$\infty$	$\infty$
1	$uw$	$6, w$		$5, u$	$11, w$	$\infty$

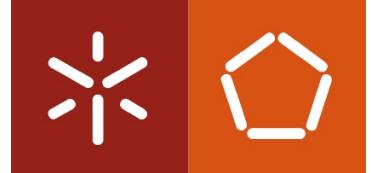




# LSA - Dijkstra: exemplo

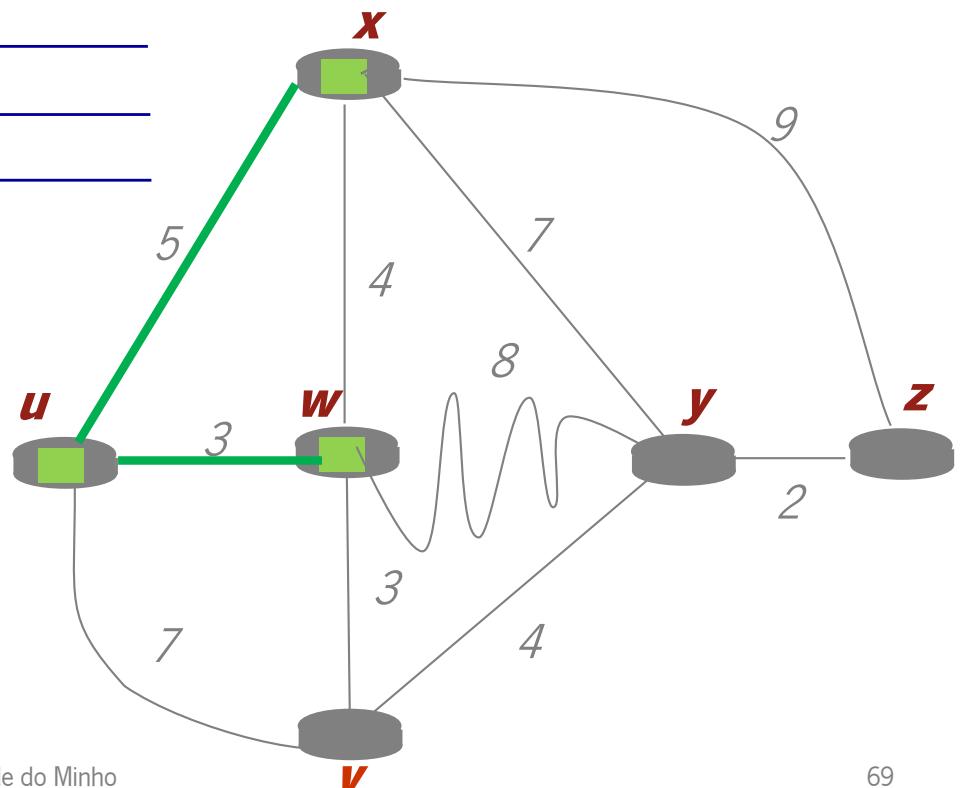
<i>Step</i>	$N'$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
		$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
0	$u$	$7, u$	$3, u$	$5, u$	$\infty$	$\infty$
1	$uw$	$6, w$		$5, u$	$11, w$	$\infty$
2	$uwx$					

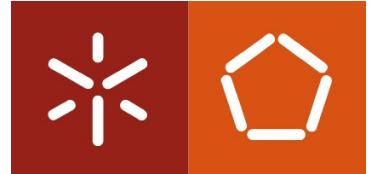




# LSA - Dijkstra: exemplo

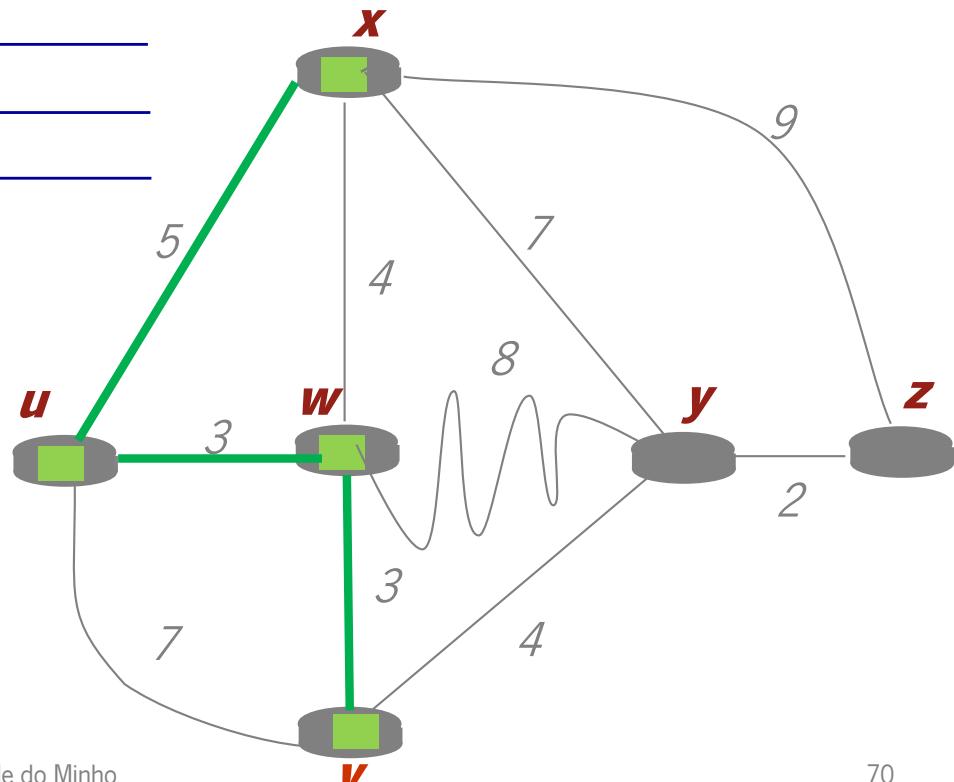
<i>Step</i>	$N'$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
		$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
0	$u$	$7,u$	$3,u$	$5,u$	$\infty$	$\infty$
1	$uw$	$6,w$		$5,u$	$11,w$	$\infty$
2	$uwx$	$6,w$			$11,w$	$14,x$

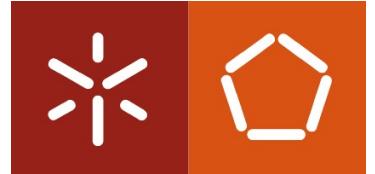




# LSA - Dijkstra: exemplo

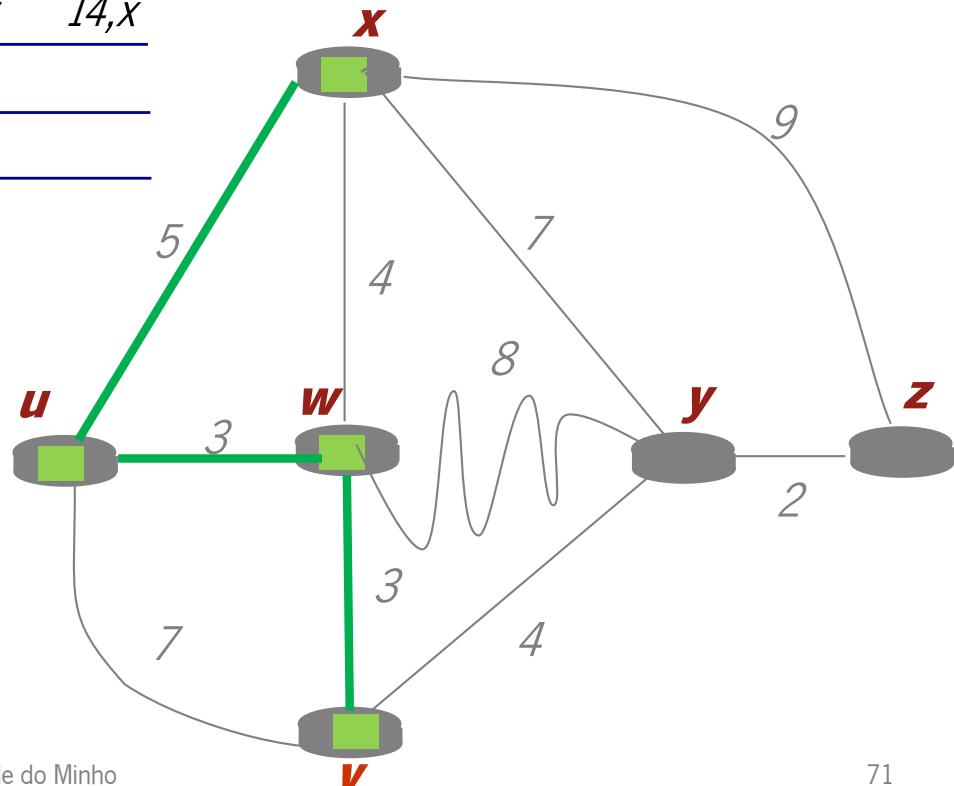
<i>Step</i>	$N'$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
		$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
0	$u$	$7, u$	$3, u$	$5, u$	$\infty$	$\infty$
1	$uw$	$6, w$		$5, u$	$11, w$	$\infty$
2	$uwx$	$6, w$			$11, w$	$14, x$
3	$uwxv$					

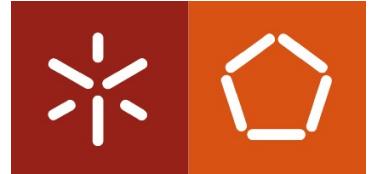




# LSA - Dijkstra: exemplo

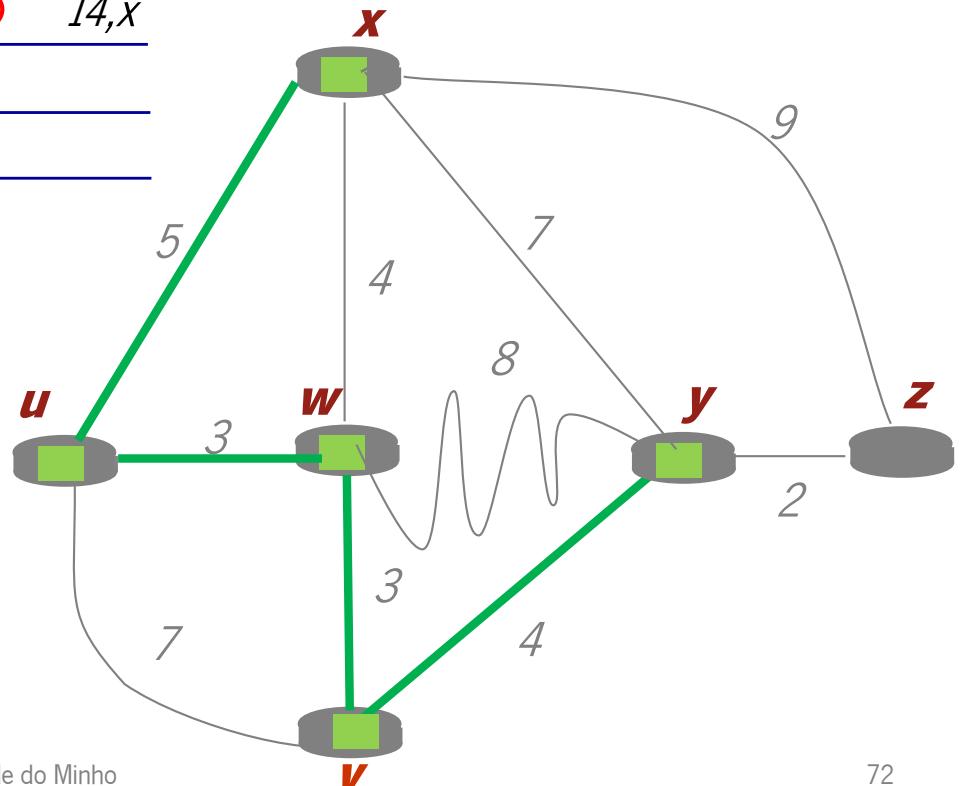
<i>Step</i>	$N'$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
		$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
0	$u$	$7, u$	$3, u$	$5, u$	$\infty$	$\infty$
1	$uw$	$6, w$		$5, u$	$11, w$	$\infty$
2	$uwx$	$6, w$			$11, w$	$14, x$
3	$uwxv$				$10, v$	$14, x$

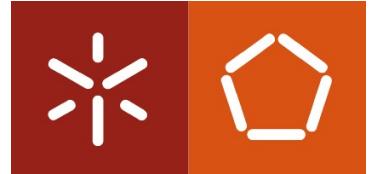




# LSA - Dijkstra: exemplo

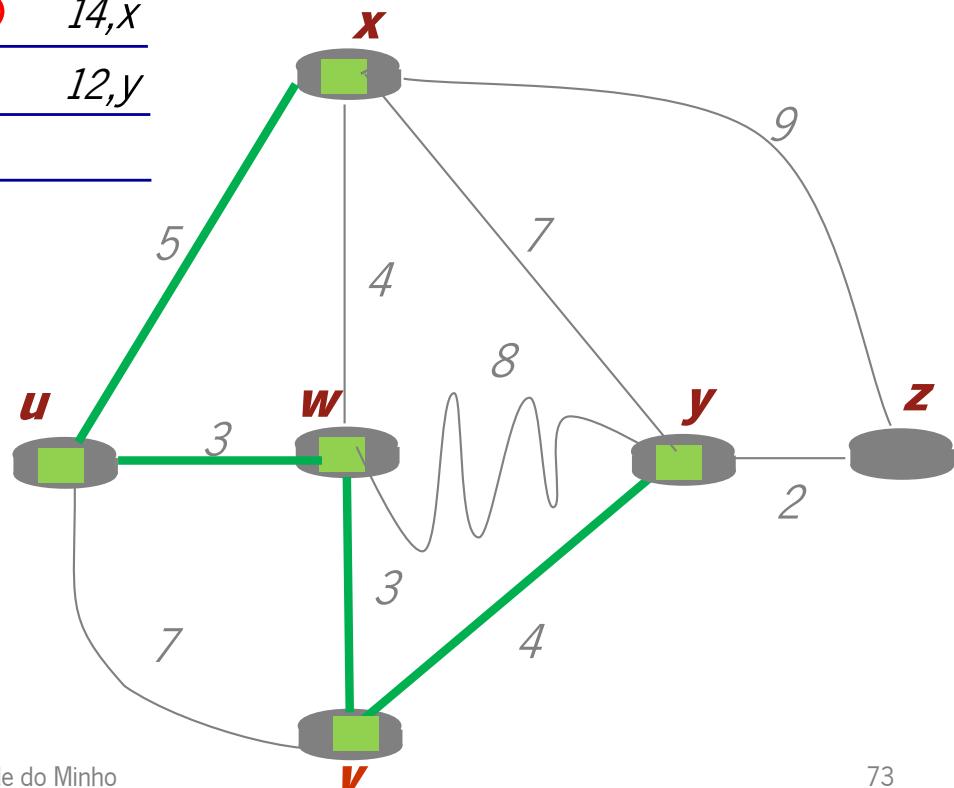
<i>Step</i>	$N'$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
		$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
0	$u$	$7, u$	$3, u$	$5, u$	$\infty$	$\infty$
1	$uw$	$6, w$		$5, u$	$11, w$	$\infty$
2	$uwx$	$6, w$			$11, w$	$14, x$
3	$uwxv$				$10, v$	$14, x$
4	$uwxvy$					

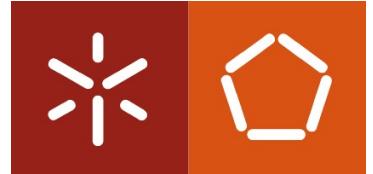




# LSA - Dijkstra: exemplo

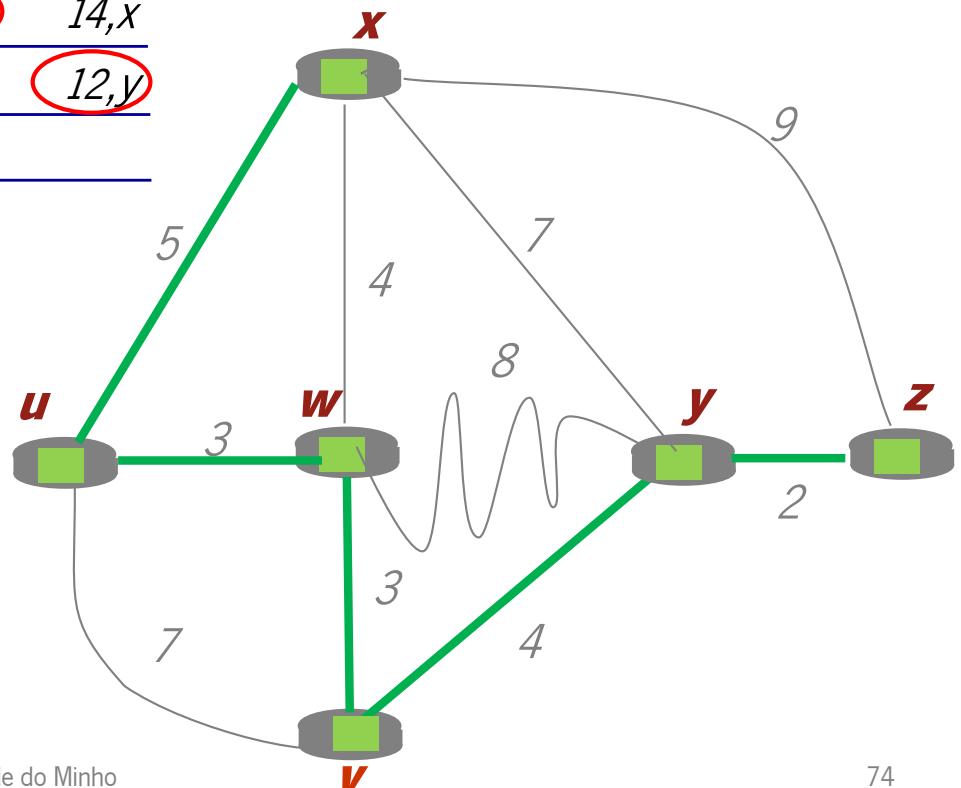
<i>Step</i>	$N'$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
		$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
0	$u$	$7, u$	$3, u$	$5, u$	$\infty$	$\infty$
1	$uw$	$6, w$		$5, u$	$11, w$	$\infty$
2	$uwx$	$6, w$			$11, w$	$14, x$
3	$uwxv$			$10, v$	$14, x$	
4	$uwxvy$				$12, y$	

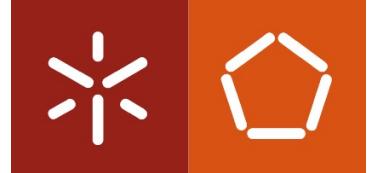




# LSA - Dijkstra: exemplo

<i>Step</i>	$N'$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
		$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
0	$u$	$7, u$	$3, u$	$5, u$	$\infty$	$\infty$
1	$uw$	$6, w$		$5, u$	$11, w$	$\infty$
2	$uwx$	$6, w$			$11, w$	$14, x$
3	$uwxv$				$10, v$	$14, x$
4	$uwxvy$				$12, y$	
5	$uwxvyz$					



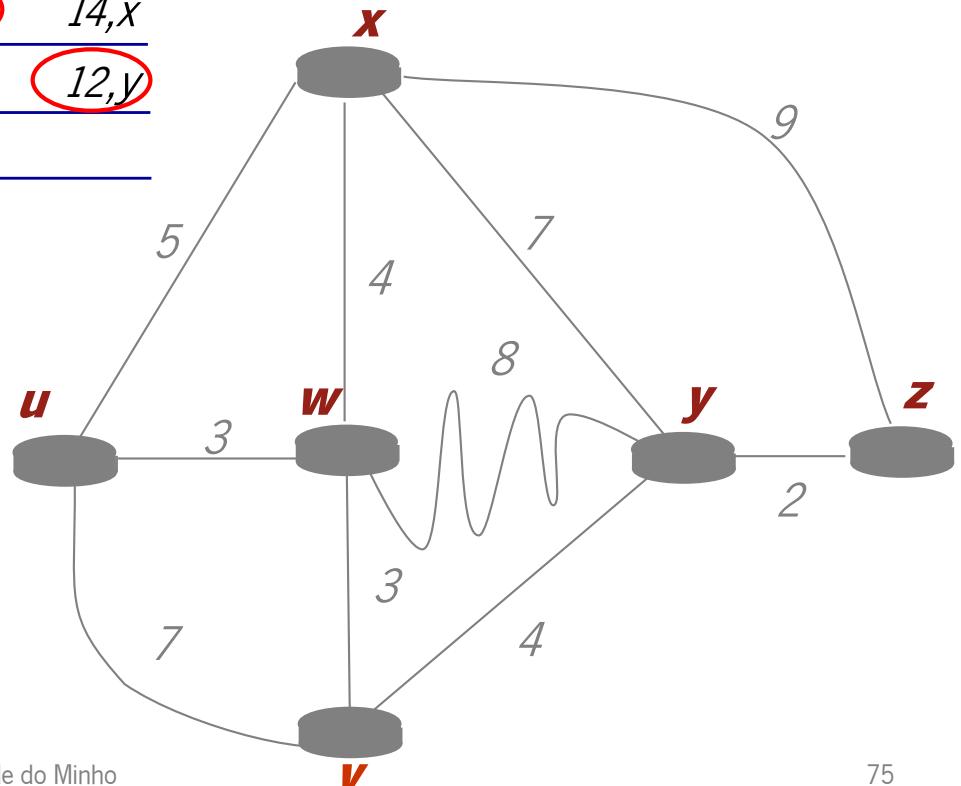


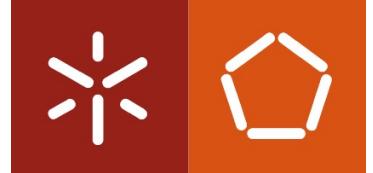
# LSA - Dijkstra: exemplo

<i>Step</i>	$N'$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
		$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
0	$u$	$7,u$	$3,u$	$5,u$	$\infty$	$\infty$
1	$uw$	$6,w$		$5,u$	$11,w$	$\infty$
2	$uwx$	$6,w$			$11,w$	$14,x$
3	$uwxv$				$10,v$	$14,x$
4	$uwxvy$					$12,y$
5	$uwxvyz$					

## Notas:

- ❖ Construir a árvore de aminhos mais curtos, registando os predecessores...
- ❖ Podem existir empates (que podem ser resolvidos arbitrariamente)



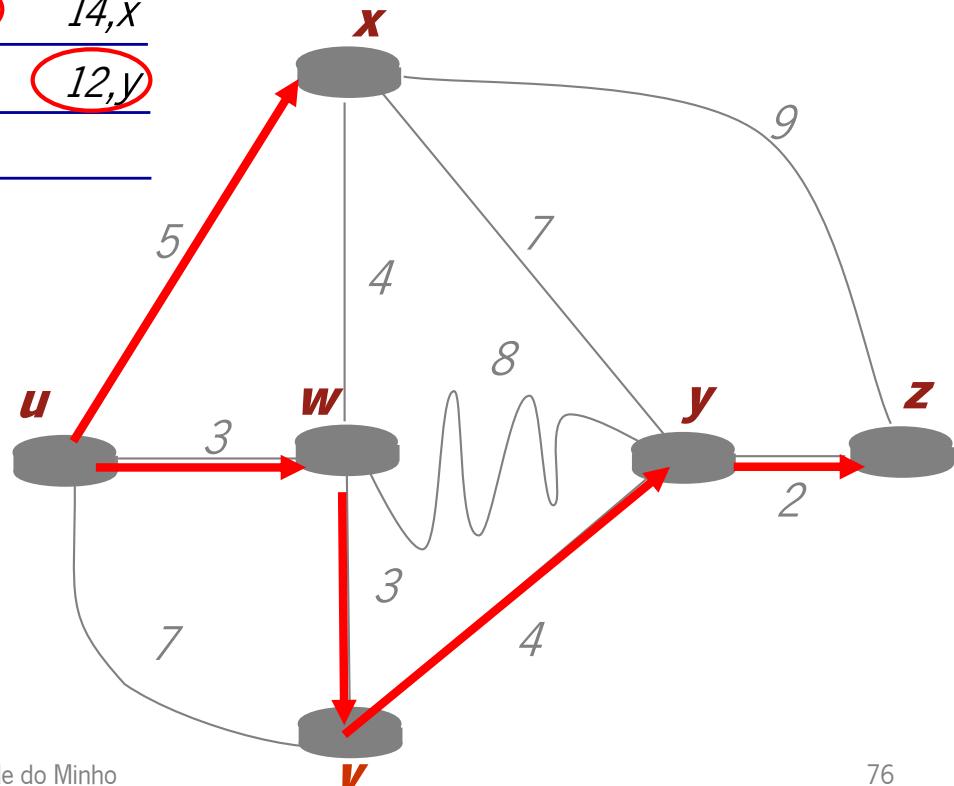


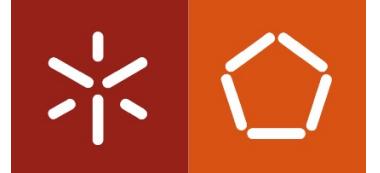
# LSA - Dijkstra: exemplo

Step	$N'$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
		$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
0	$u$	$7, u$	$3, u$	$5, u$	$\infty$	$\infty$
1	$uw$	$6, w$		$5, u$	$11, w$	$\infty$
2	$uwx$	$6, w$			$11, w$	$14, x$
3	$uwxv$				$10, v$	$14, x$
4	$uwxvy$					$12, y$
5	$uwxvyz$					

## Notas:

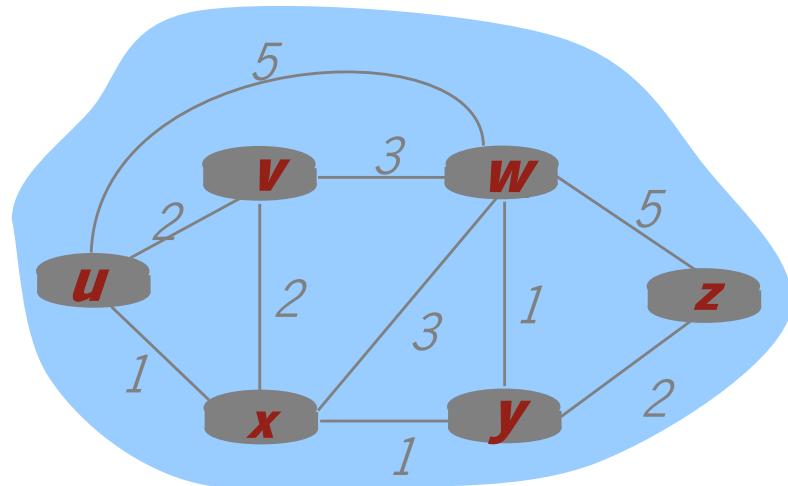
- ❖ Construir a árvore de aminhos mais curtos, registando os predecessores...
- ❖ Podem existir empates (que podem ser resolvidos arbitrariamente)



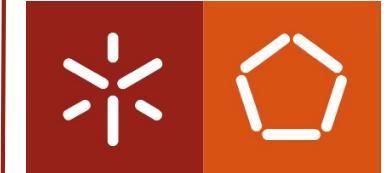


# LSA - Dijkstra: outro exemplo

Step	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	$u$	$2, u$	$5, u$	$1, u$	$\infty$	$\infty$
1	$ux$	$2, u$	$4, x$		$2, x$	$\infty$
2	$uxy$	$2, u$	$3, y$			$4, y$
3	$uxyv$		$3, y$			$4, y$
4	$uxyvw$					$4, y$
5	$uxyvwz$					

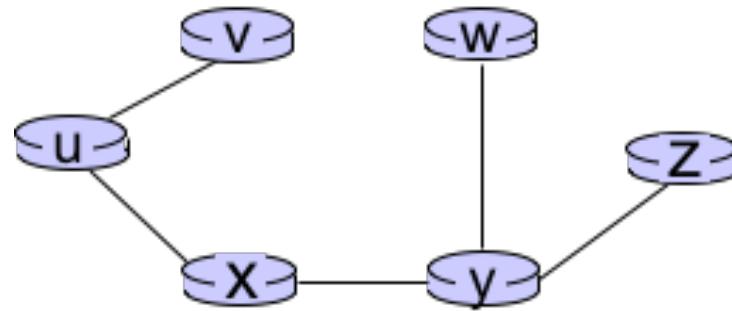


\* Check out the online interactive exercises for more examples: [http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)



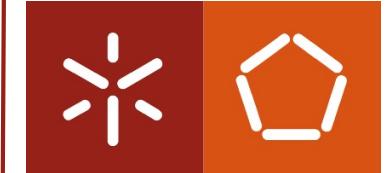
# LSA - Dijkstra: outro exemplo

*Árvore de caminhos mais curtos resultante:*

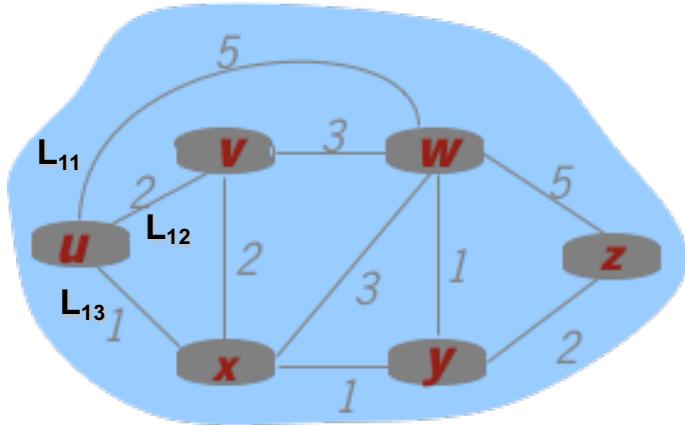


*Tabela de encaminhamento resultante:*

<i>destination</i>	<i>link</i>
<i>v</i>	$(u,v)$
<i>x</i>	$(u,x)$
<i>y</i>	$(u,x)$
<i>w</i>	$(u,x)$
<i>z</i>	$(u,x)$



# LSA - Tabela de encaminhamento



NÓ u

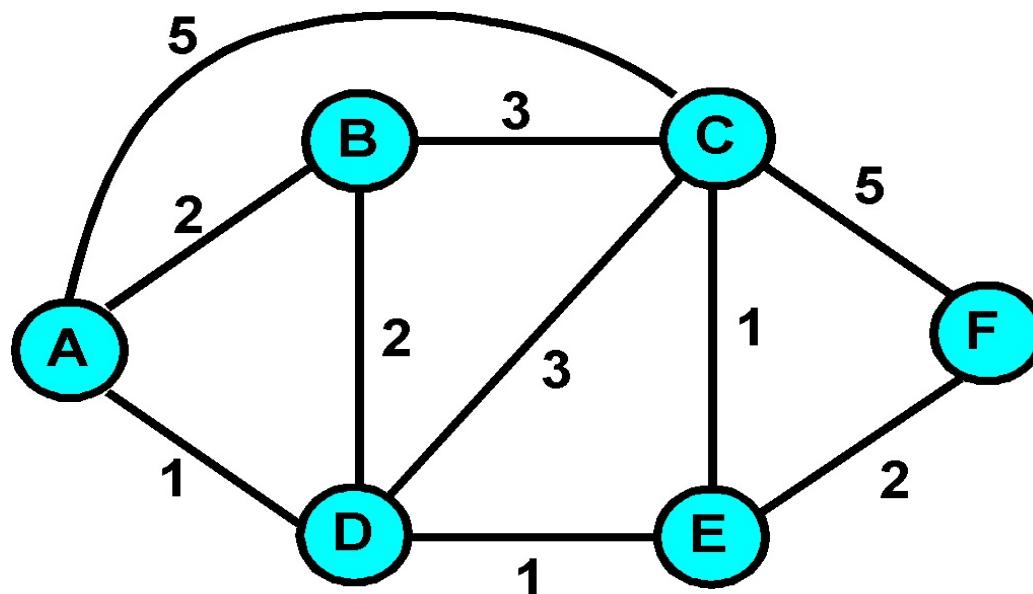
Destino	Próximo Nó	Link	Custo
u	u	—	0
v	v	L <sub>12</sub>	2
w	x	L <sub>13</sub>	3
x	x	L <sub>13</sub>	1
y	x	L <sub>13</sub>	2
z	x	L <sub>13</sub>	4



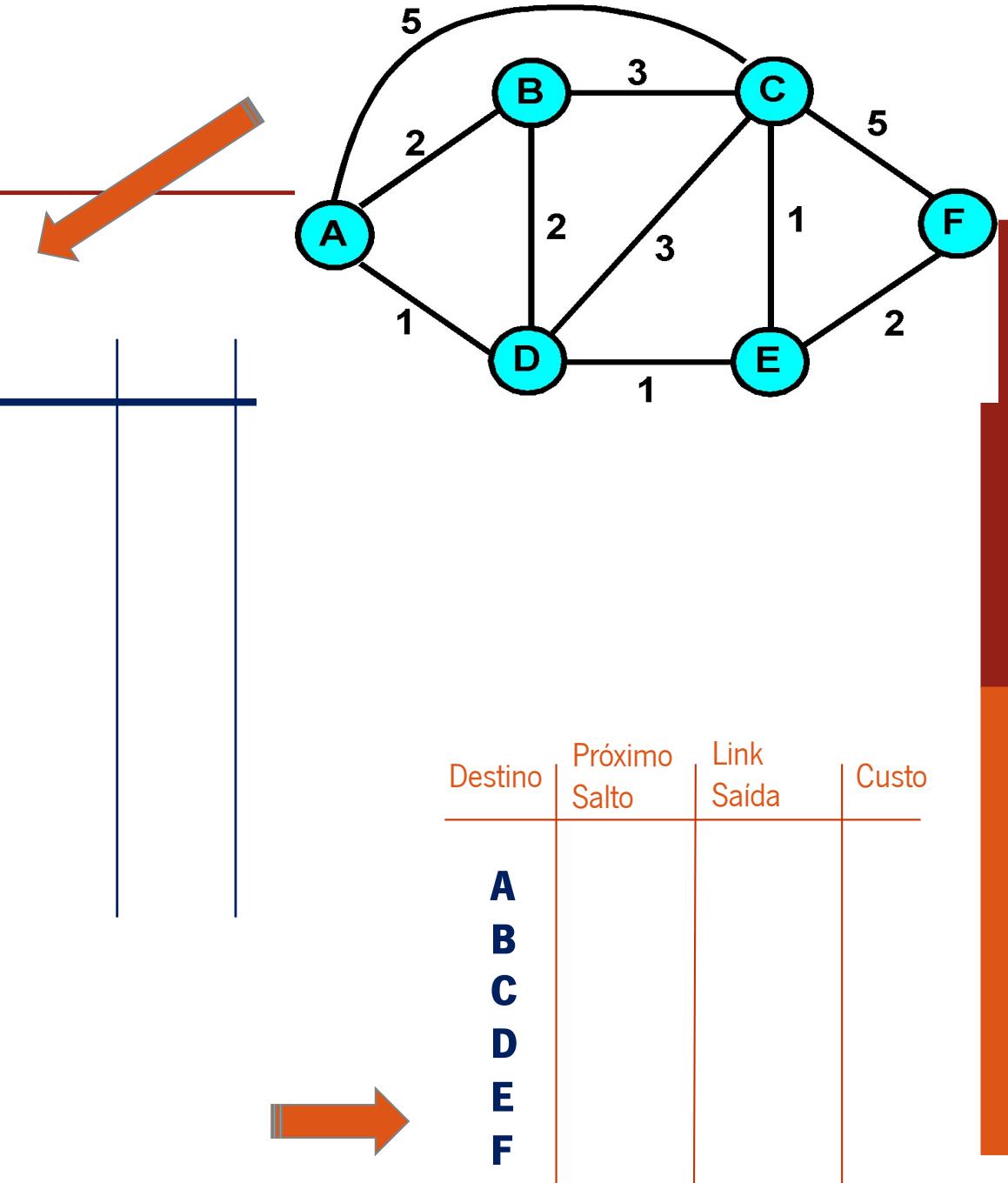
# Algoritmos do Estado das Ligações (LSA)

## ● Exercício:

Utilizando o algoritmo de **Dijkstra** determine a tabela de encaminhamento do nó **F** partindo do princípio que este nó tem um conhecimento completo da topologia da rede.

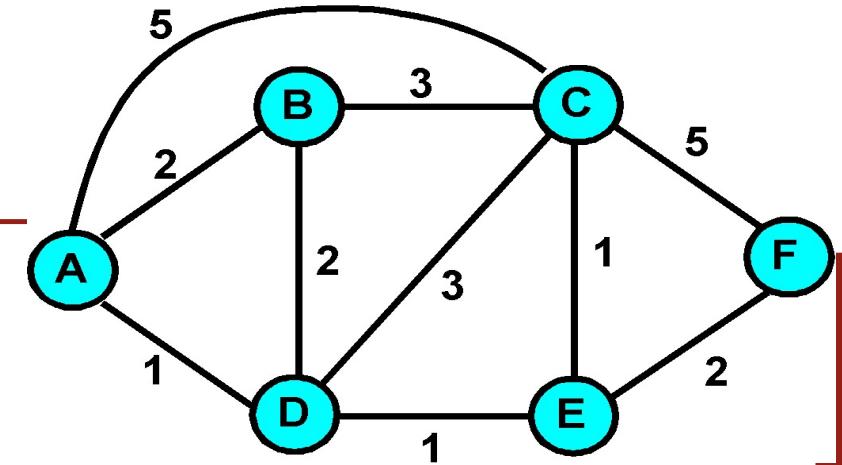
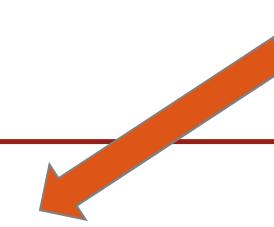


## Exercício (LSA)

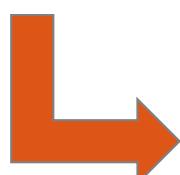


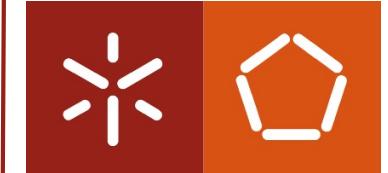
## Exercício (LSA)

N	Visitados	C(A) P(A)	C(B) P(B)	C(C) P(C)	C(D) P(D)	C(E) P(E)



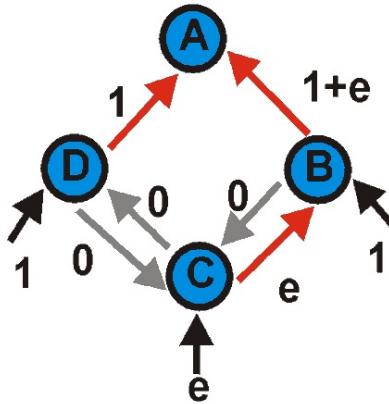
Destino	Próximo Salto	Link Saída	Custo
A			
B			
C			
D			
E			
F			



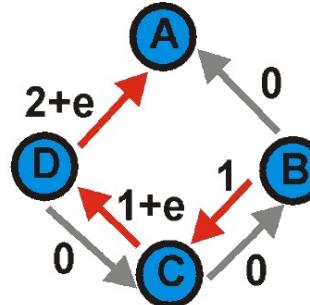


# LSA – Oscilações

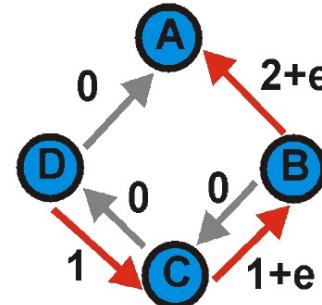
- Esforço do algoritmo:  $O(n^2)$
- Este algoritmo pode, na presença de métricas que dependem do estado da rede, apresentar alguns problemas
  - Por exemplo, se a métrica refletir a carga nas ligações, sendo por isso uma métrica assimétrica
  - No exemplo, B e D enviam uma unidade de tráfego para A e C envia  $e$  unidades de tráfego também para A



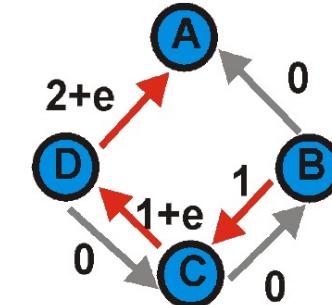
(a): initial  
routing



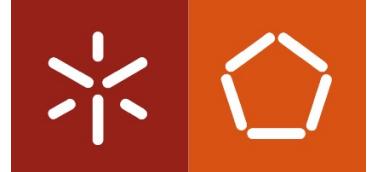
(b): B, C  
detect better  
path to A,  
clockwise



(c): B, C, D  
detect better  
path to A,  
counterclockwise



(d): B, C, D  
detect better  
path to A,  
clockwise



# Algoritmos de Vectores de Distância (DVA)

- Ao contrário dos algoritmos de estado de ligação, os algoritmos de vectores de distância não usam informação global;
- São distribuídos, iterativos e assíncronos
  - Cada nó recebe informação de encaminhamento de algum dos seus vizinhos directos, recalcula a tabela de encaminhamento e envia essa informação de volta para os vizinhos;
  - O processo continua até que não haja informação de encaminhamento a ser trocada entre nós vizinhos
  - Não exige que os nós estejam sincronizados uns com os outros



# Algoritmos de Vectores de Distância (DVA)

## Equação de Bellman-Ford

Seja

$d_x(y) := \text{custo do caminho de custo mínimo de } x \text{ para } y$

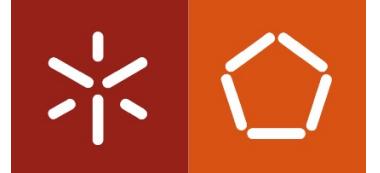
Então

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

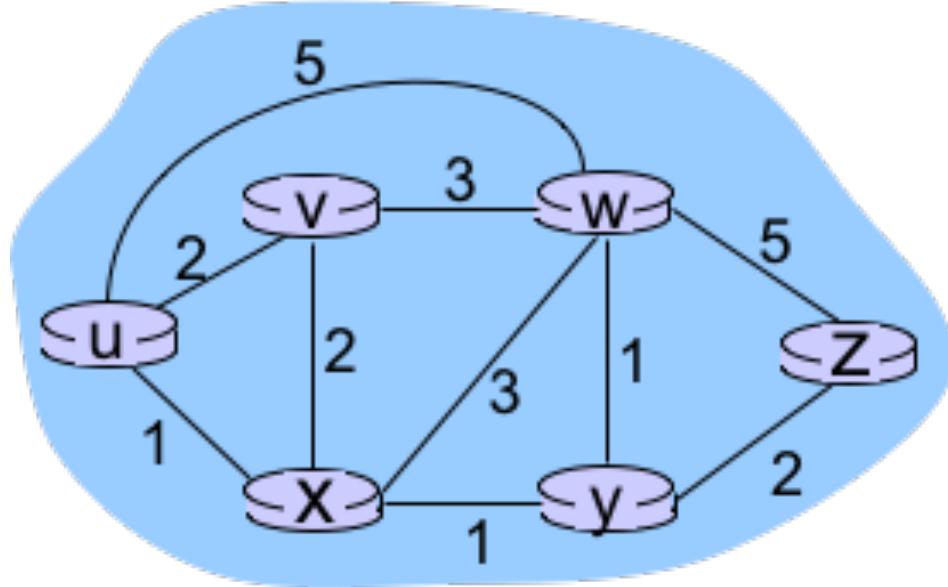
*Custo de X para vizinho V*

*Custo do vizinho V para o destino Y*

*Sendo o mínimo obtido de todos os vizinhos v de y*



# Exemplo

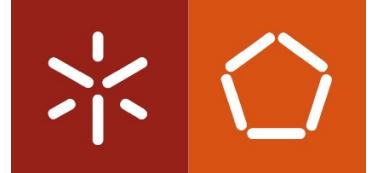


$$d_v(z) = 5, d_w(z) = 3, d_x(z) = 3$$

Bellman-Ford diz que:

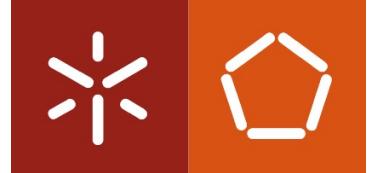
$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\quad c(u,w) + d_w(z), \\ &\quad c(u,x) + d_x(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 5 + 3, \\ &\quad 1 + 3 \} = 4 \end{aligned}$$

**O nó vizinho que conseguir o valor mínimo  
Será o próximo salto → tabela de reenvio**



# Algoritmos de Vectores de Distância (DVA)

- $D_x(y)$  = estimativa do custo mínimo de  $x$  para  $y$
- Nó  $x$  conhece o custo para todos os seus vizinhos  $v$ :  $c(x,v)$
- Nó  $x$  mantém um vector de distâncias  $D_x = [D_x(y): y \in N]$
- Nó  $x$  também mantém os vectores de distâncias dos seus vizinhos:
  - Para cada vizinho  $v$ ,  $x$  guarda  
 $D_v = [D_v(y): y \in N]$



# Algoritmos de Vectores de Distância (DVA)

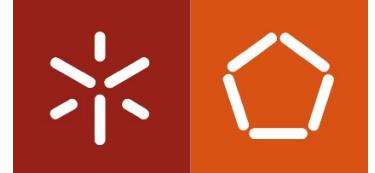
## A ideia básica:

- Cada nó envia periodicamente a sua estimativa do vector de distâncias a todos os seus vizinhos
- Quando um nó  $x$  recebe um novo vector VD de um dos seus vizinhos, actualiza o seu próprio vector de distâncias usando a equação de Bellman-Ford:

$$D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \quad \text{Para todos os nós } y \in N$$

- Em condições normais, a estimativa do vector distâncias  $D_x(y)$  converge para o vector de custo mínimo  $d_x(y)$

# Algoritmos de Vector de Distâncias (DVA)



Cada nó:

*wait* (msg do vizinho c/ info alteração menor-custo do link local)

*recalcular* tabela distâncias

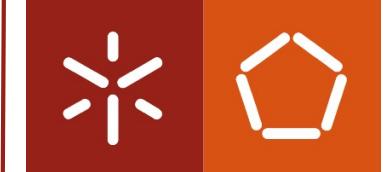
*if* mudou(menor-custo(qq-DEST) *then notify* vizinhos

Iterativo, assíncrono: cada iteração local é causada por:

- mudança custo link local
- mensagem do vizinho: vizinho anuncia novo custo

Distribuído:

- cada nó notifica vizinhos só quando muda o menor custo p/ qq destino
- vizinhos notificam vizinhos (se necessário!)



# Algoritmo Belman-Ford

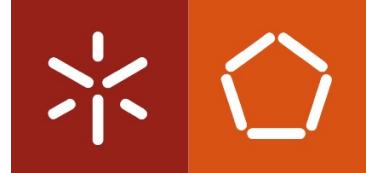
$$\begin{aligned} D^X(Y,Z) &= \text{distance from } X \text{ to } Y, \text{ via } Z \text{ as next hop} \\ &= c(X,Z) + \min_w \{D^Z(Y,w)\} \end{aligned}$$

At each node, X:

- 1 *Initialization:*
- 2 *for* all adjacent nodes v:
  - 3  $D^X(*,v) = \text{infty}$  /\* the \* operator means "for all rows" \*/
  - 4  $D^X(v,v) = c(X,v)$
  - 5 *for* all destinations, y
  - 6 send  $\min_w D(y,w)$  to each neighbor /\* w over all X's neighbors \*/

7

Fonte: Computer Networking: A Top-Down Approach Featuring the Internet,  
J. Kurose, Addison-Wesley



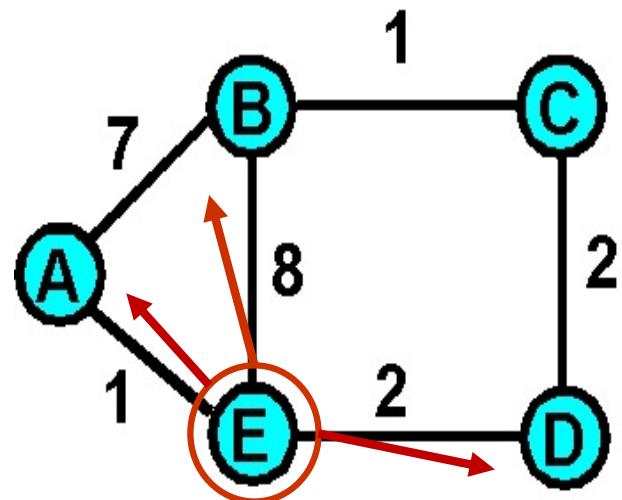
# Algoritmo Belman-Ford

```
8 loop forever
9   wait (until I see a link cost change to neighbor V
10    or until I receive update from neighbor V)
12   if (c(X,V) changes by d)
13     /* change cost to all dest's via neighbor v by d */
14     /* note: d could be positive or negative */
15     for all destinations y: DX(y,V) = DX(y,V) + d
16
17   else if (update received from V wrt destination Y)
18     /* shortest path from V to some Y has changed */
19     /* V has sent a new value for its minw DV(Y,w) */
20     /* call this received new value is "newval" */
21     for the single destination y: DX(Y,V) = c(X,V) + newval
22
23   if we have a new minw DX(Y,w) for any destination Y
24     send new value of minw DX(Y,w) to all neighbors
```



# Tabela de distâncias

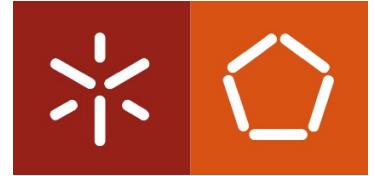
DVA



		cost to destination via		
		A	B	D
d e s t i n a t.	A	1	14	5
	B	7	8	5
	C	6	9	4
	D	4	11	2
	E	( )		

Fonte: Computer Networking: A Top-Down Approach Featuring the Internet, J. Kurose, Addison-Wesley, 2001

# Da Tabela de Distâncias à Tabela de Encaminhamento



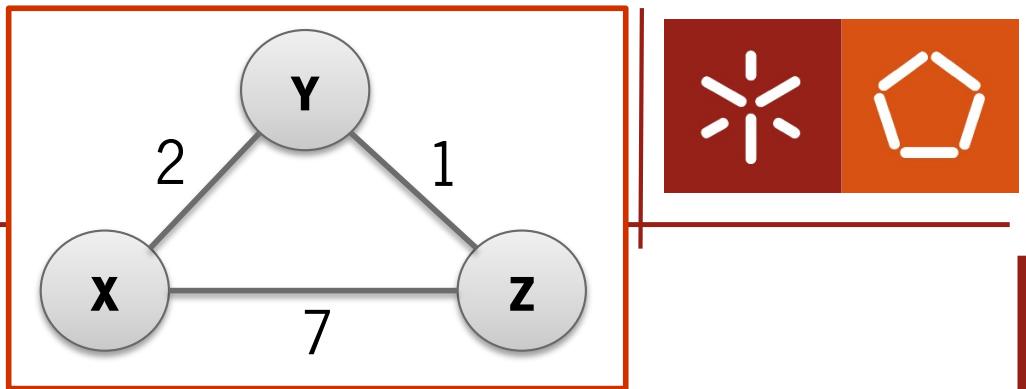
D <sup>E</sup> ( )	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination		Link Saída, custo
A	A,	1
B	D,	5
C	D,	4
D	D,	2

Tabela Distâncias → Tabela de Routing

# DVA – Operação

Tabelas de Distâncias



Vizinhos	
Destinos	X Y Z
D <sup>X</sup>	Y 2 Z $\infty$
Y	Z $\infty$

Vizinhos	
Destinos	X Z
D <sup>Y</sup>	X 2 Z $\infty$
X	Z $\infty$

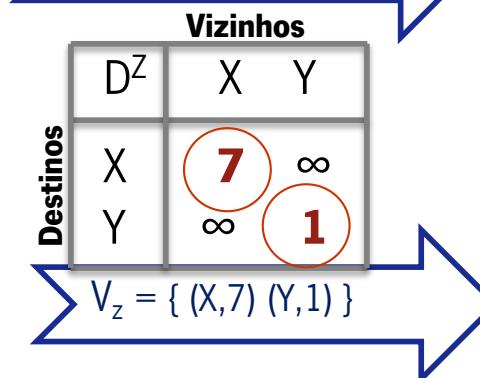
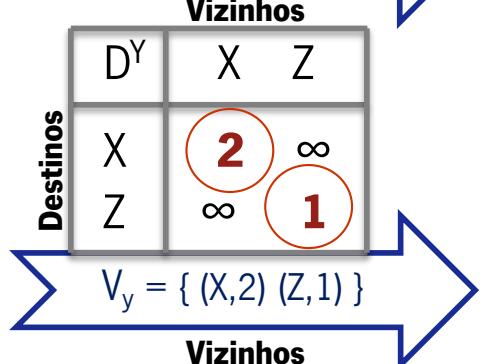
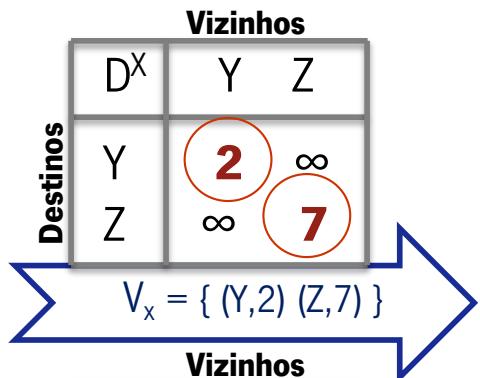
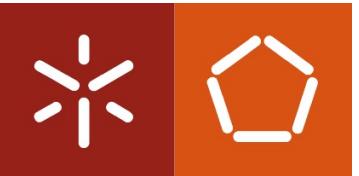
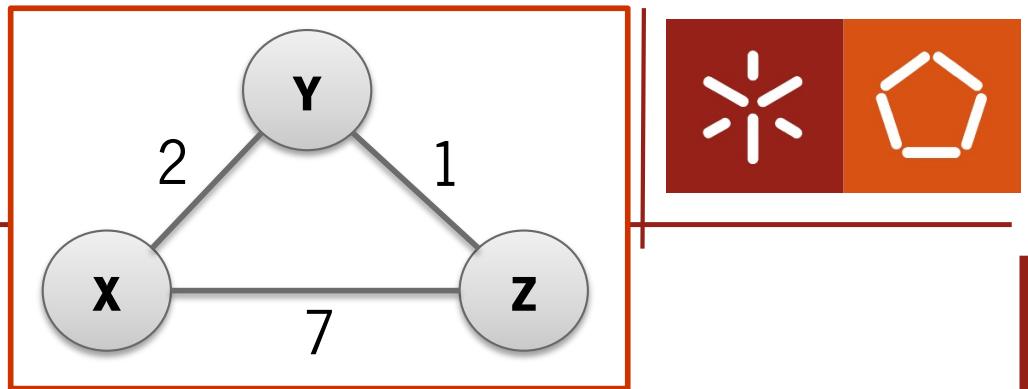
Vizinhos	
Destinos	X Y
D <sup>Z</sup>	X 7 Y $\infty$
X	Y $\infty$

Inicialização:

- . Custo para os vizinhos é o custo do link direto
- . Todos os outros a infinito

# DVA – Operação

Tabelas de Distâncias



Preparar vetor com as melhores distâncias  
Enviar vetor distâncias a todos os vizinhos

# DVA – Operação

Tabelas de Distâncias

		Vizinhos	
		$D^X$	$Y \quad Z$
Destinos	$D^Y$	$Y$	$2 \quad \infty$
	$Z$	$\infty$	$7$

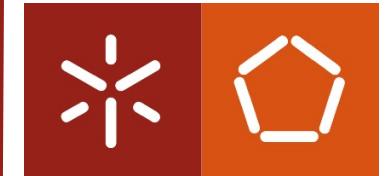
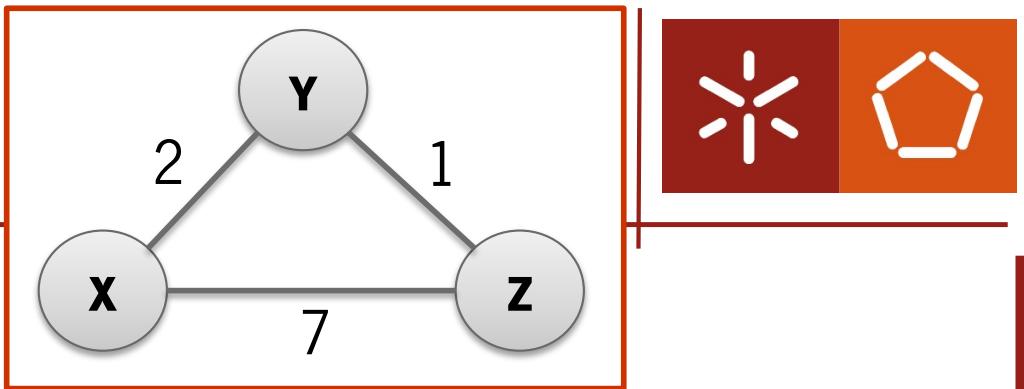
$$V_x = \{ (Y, 2) (Z, 7) \}$$

		Vizinhos	
		$D^Y$	$X \quad Z$
Destinos	$D^Z$	$X$	$2 \quad \infty$
	$Z$	$\infty$	$1$

$$V_y = \{ (X, 2) (Z, 1) \}$$

		Vizinhos	
		$D^Z$	$X \quad Y$
Destinos	$D^X$	$X$	$7 \quad \infty$
	$Y$	$\infty$	$1$

$$V_z = \{ (X, 7) (Y, 1) \}$$



O Nó X Recebe Vetores de Y e Z e atualiza tabela:

→ Y diz que chega a Z com custo 1

$$D^X(Z, Y) = c(X, Y) + D^Y(Z, Z) = 2 + 1 = 3$$

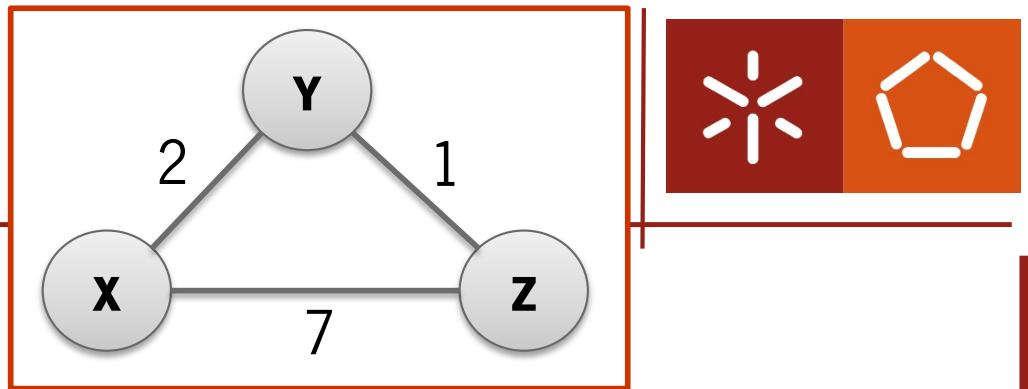
→ Z diz que chega a Y com custo de 1

$$D^X(Y, Z) = c(X, Z) + D^Z(Y, Y) = 7 + 1 = 8$$

aplicando a equação Bellman-Ford !

# DVA – Operação

Tabelas de Distâncias



Vizinhos	
Destinos	D <sup>X</sup>
Y	2
Z	8

Vizinhos	
Destinos	D <sup>X</sup>
Y	2
Z	3

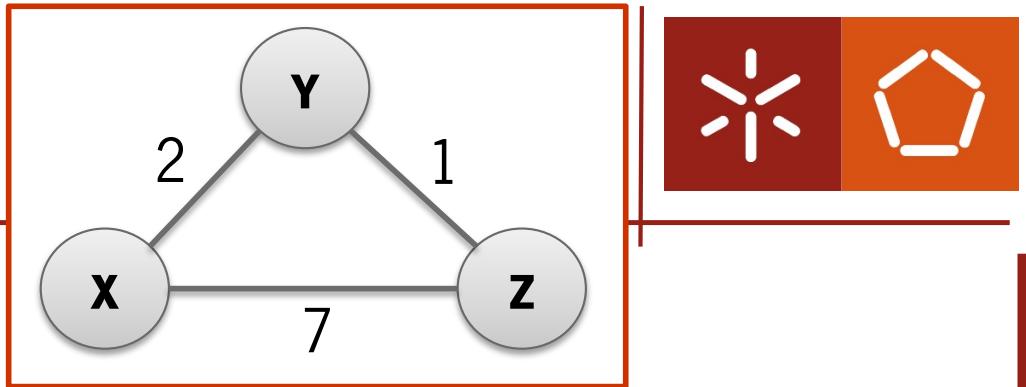
Vizinhos	
Destinos	D <sup>Y</sup>
X	2
Z	1

Vizinhos	
Destinos	D <sup>Z</sup>
X	7
Y	1

Escolhe as melhores distâncias...  
... se houver alterações envia aos vizinhos!

# DVA – Operação

Tabelas de Distâncias



Vizinhos	
Destinos	D <sup>X</sup>
Y	2
Z	8

$$V_x = \{ (Y, 2) (Z, 8) \}$$

Vizinhos	
Destinos	D <sup>X</sup>
Y	2
Z	3

$$V_x = \{ (Y, 2) (Z, 3) \}$$

Vizinhos	
Destinos	D <sup>Y</sup>
X	2
Z	1

$$V_y = \{ (X, 2) (Z, 1) \}$$

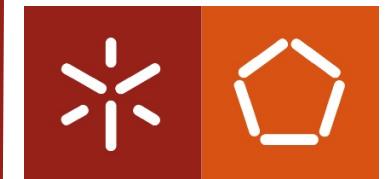
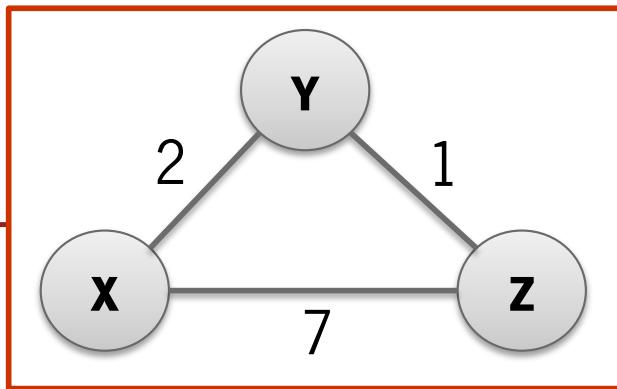
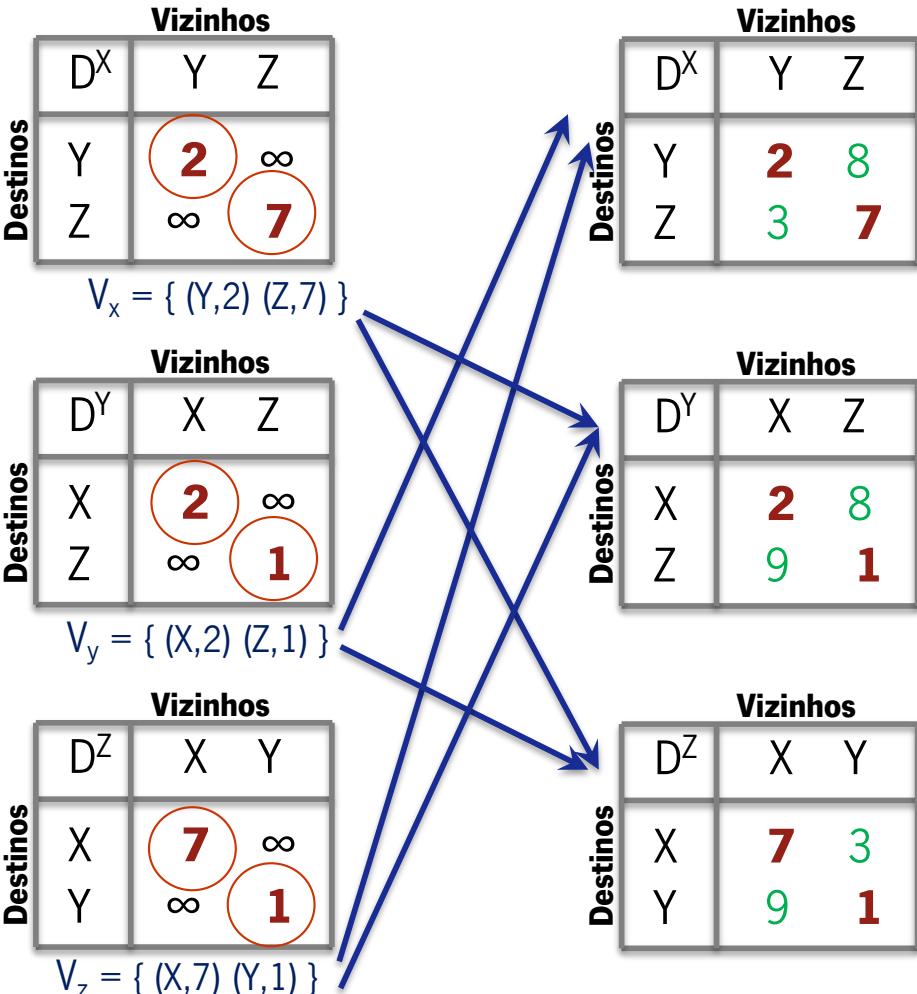
Vizinhos	
Destinos	D <sup>Z</sup>
X	7
Y	1

$$V_z = \{ (X, 7) (Y, 1) \}$$

... na segunda iteração

# DVA – Operação

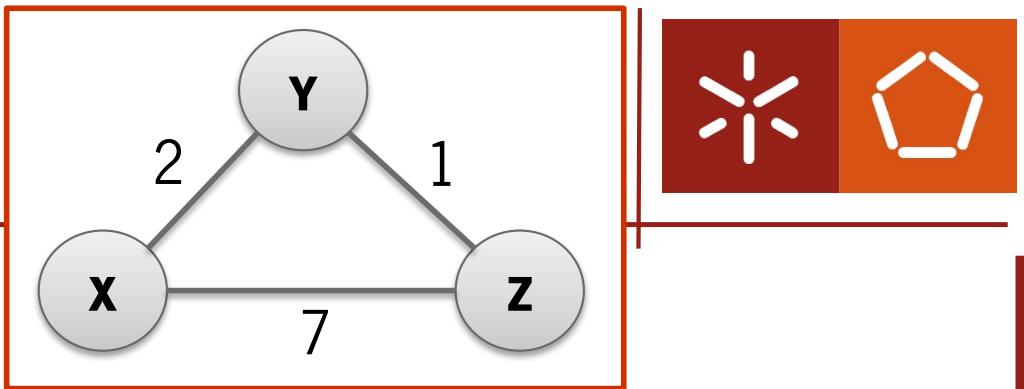
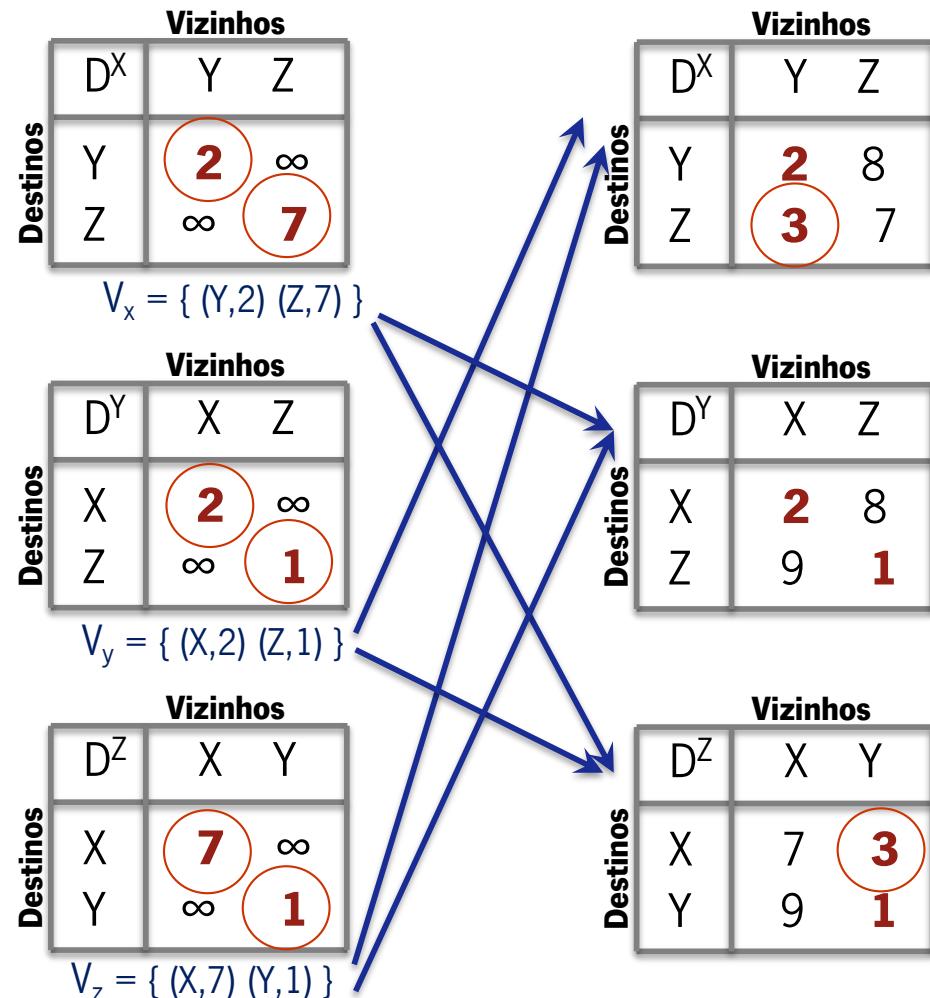
Tabelas de Distâncias



... o mesmo se passa nos nós Y e Z

# DVA – Operação

Tabelas de Distâncias



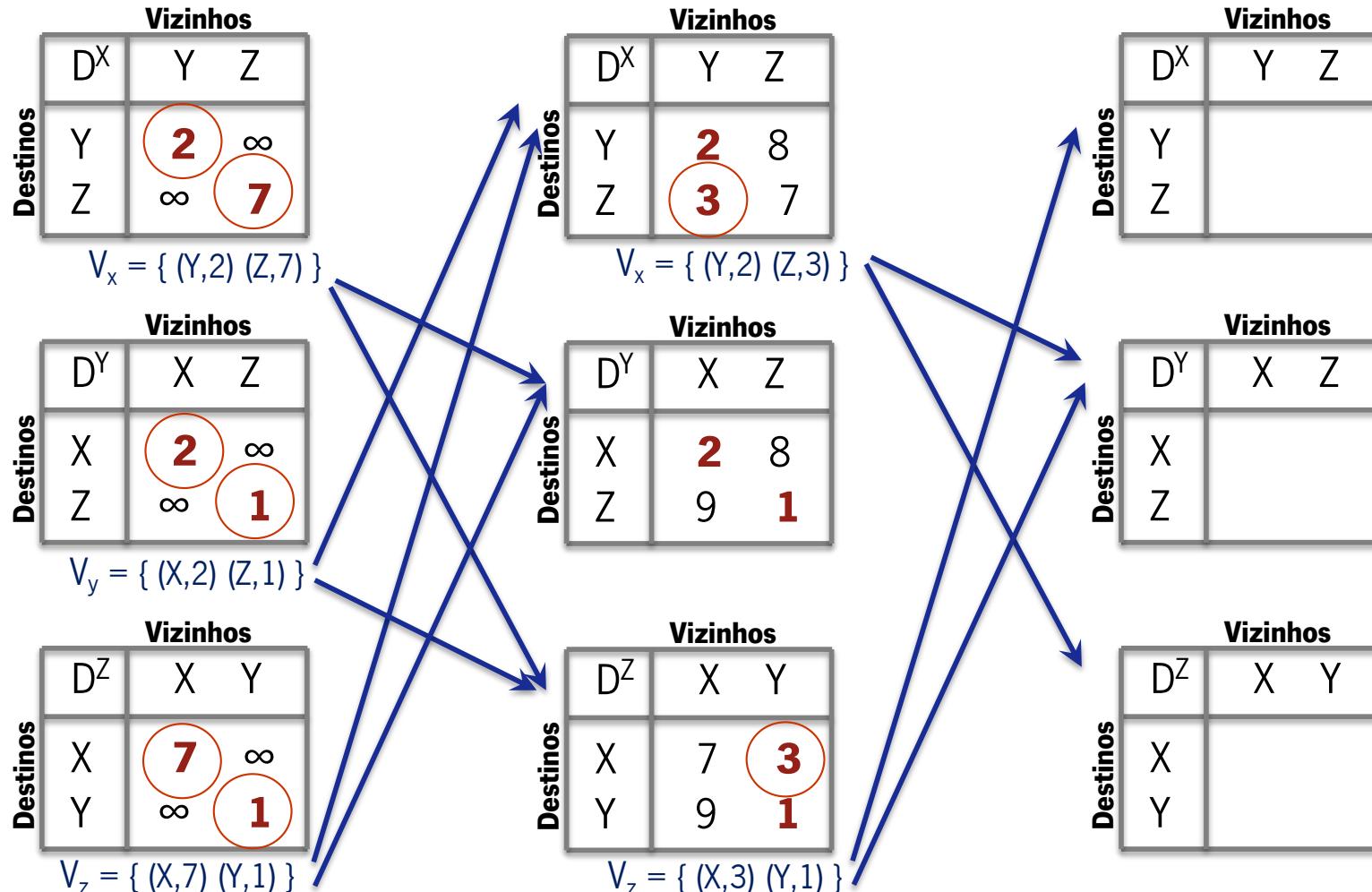
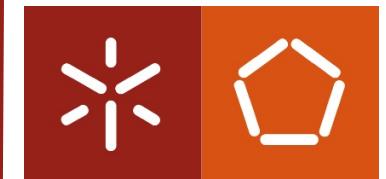
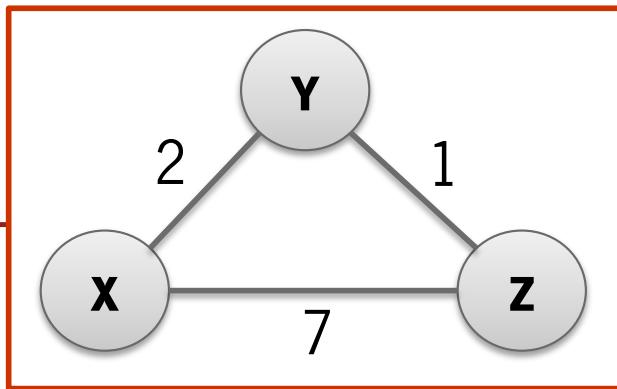
Nó X: Aprendeu uma nova rota para Z por Y

Nó Y... não melhorou nenhuma rota.  
Convergiu. Não envia nada.

Nó Z: Aprendeu uma nova rota para X por Y

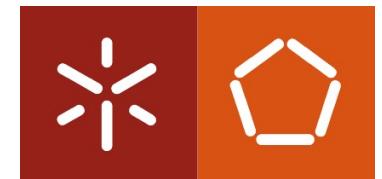
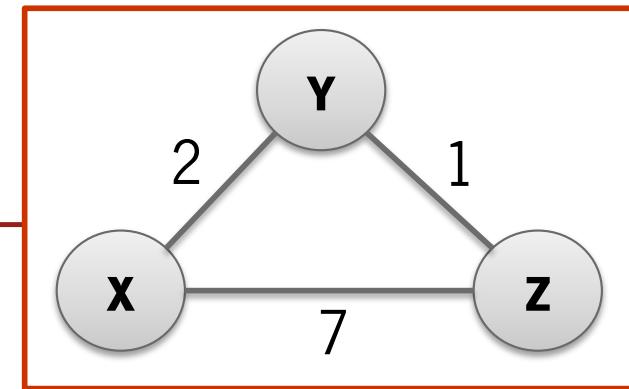
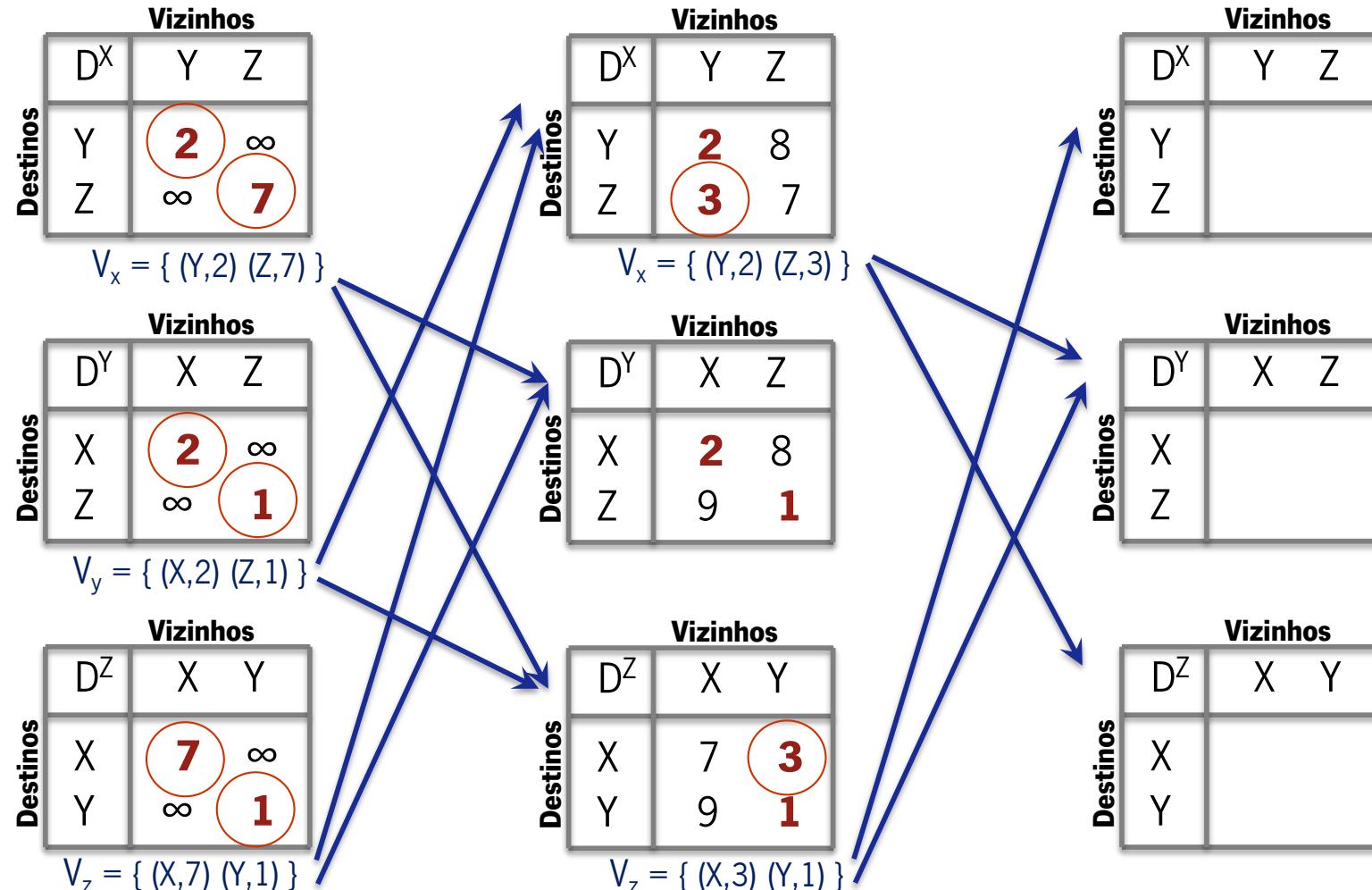
# DVA – Operação

Tabelas de Distâncias



# DVA – Operação

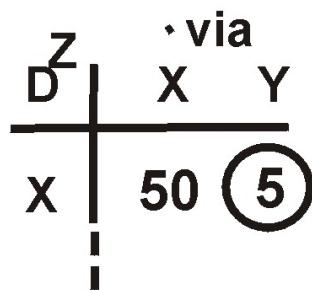
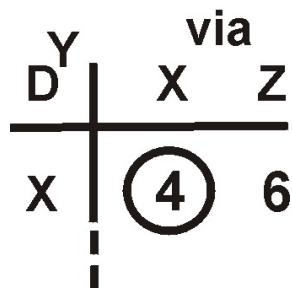
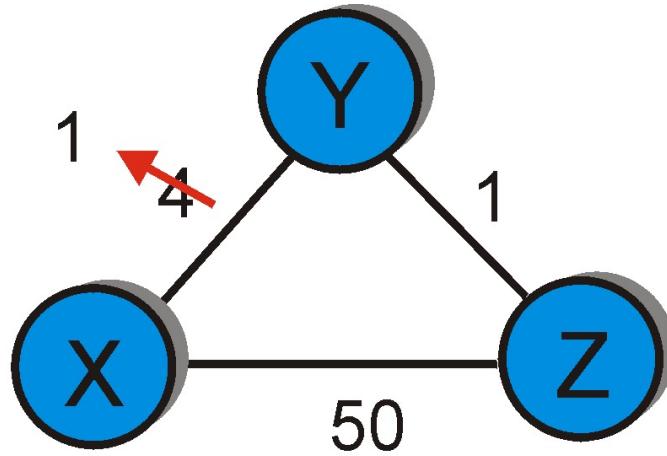
Tabelas de Distâncias



Que  
acontece  
na  
segunda  
iteração?

Já  
convergiu?

## DVA: Good News...

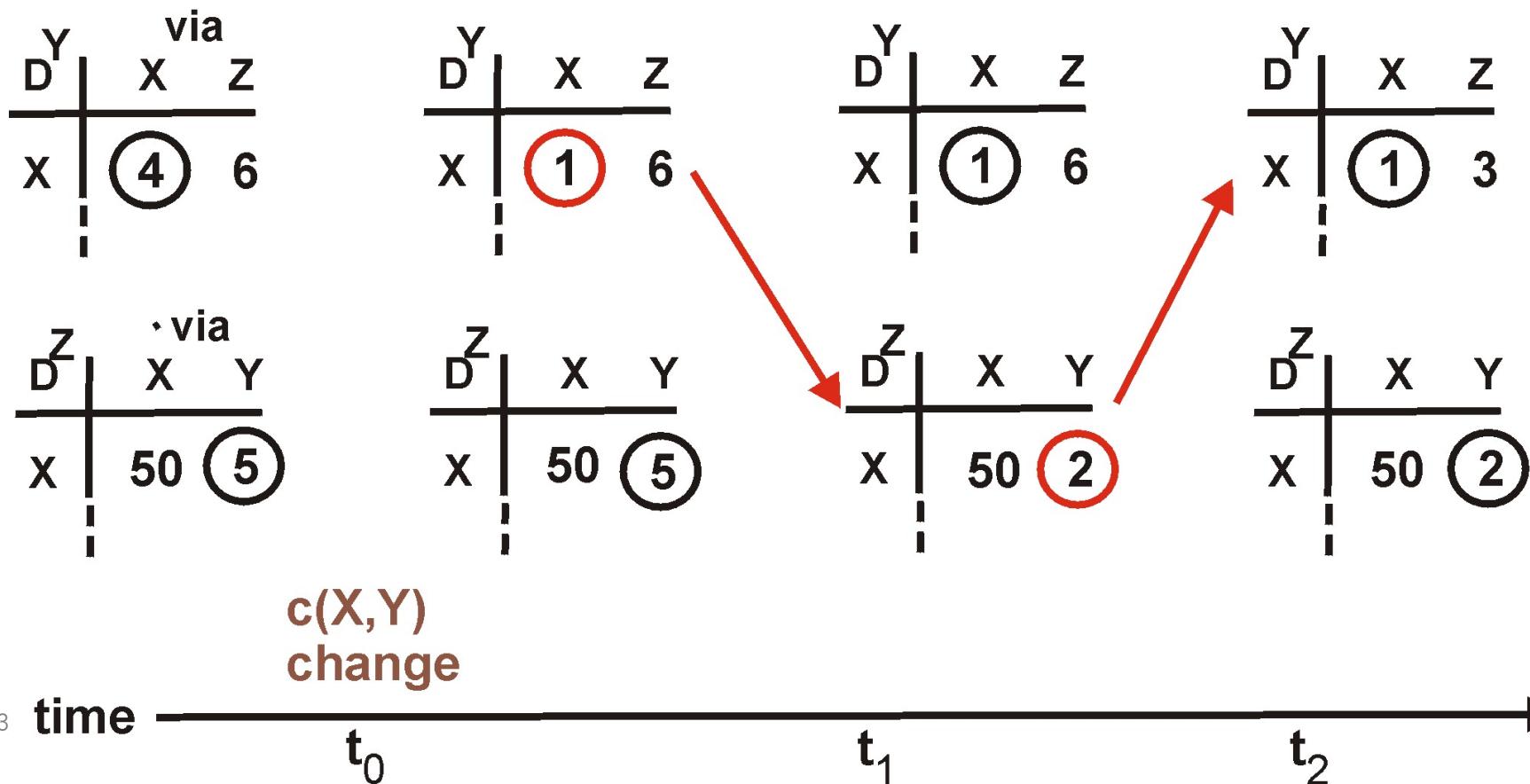
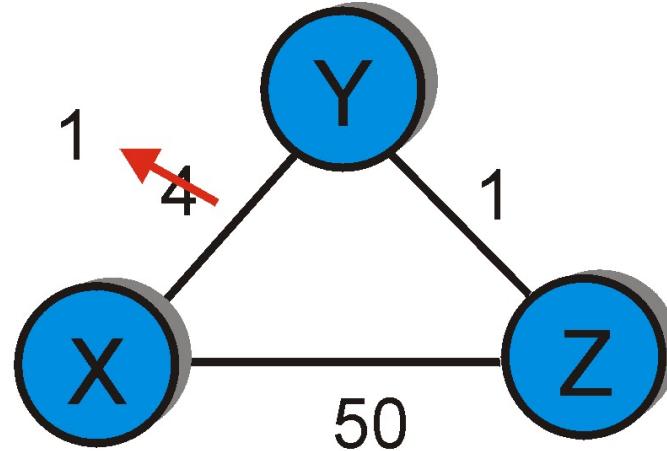


$c(X,Y)$   
change

03/ time —————  $t_0$   $t_1$   $t_2$  → 103

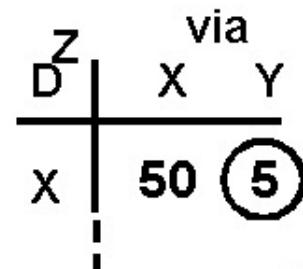
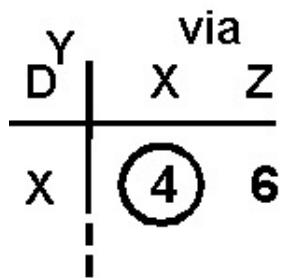
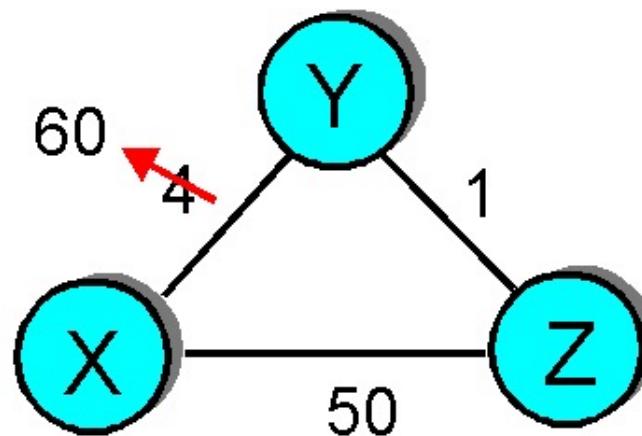
Fonte: Computer Networking: A Top-Down Approach Featuring the Internet, J. Kurose, Addison-Wesley, 2001

# DVA: Good News... Travel Fast





## DVA: Bad News...



$c(X,Y)$   
**change**

time

$t_0$

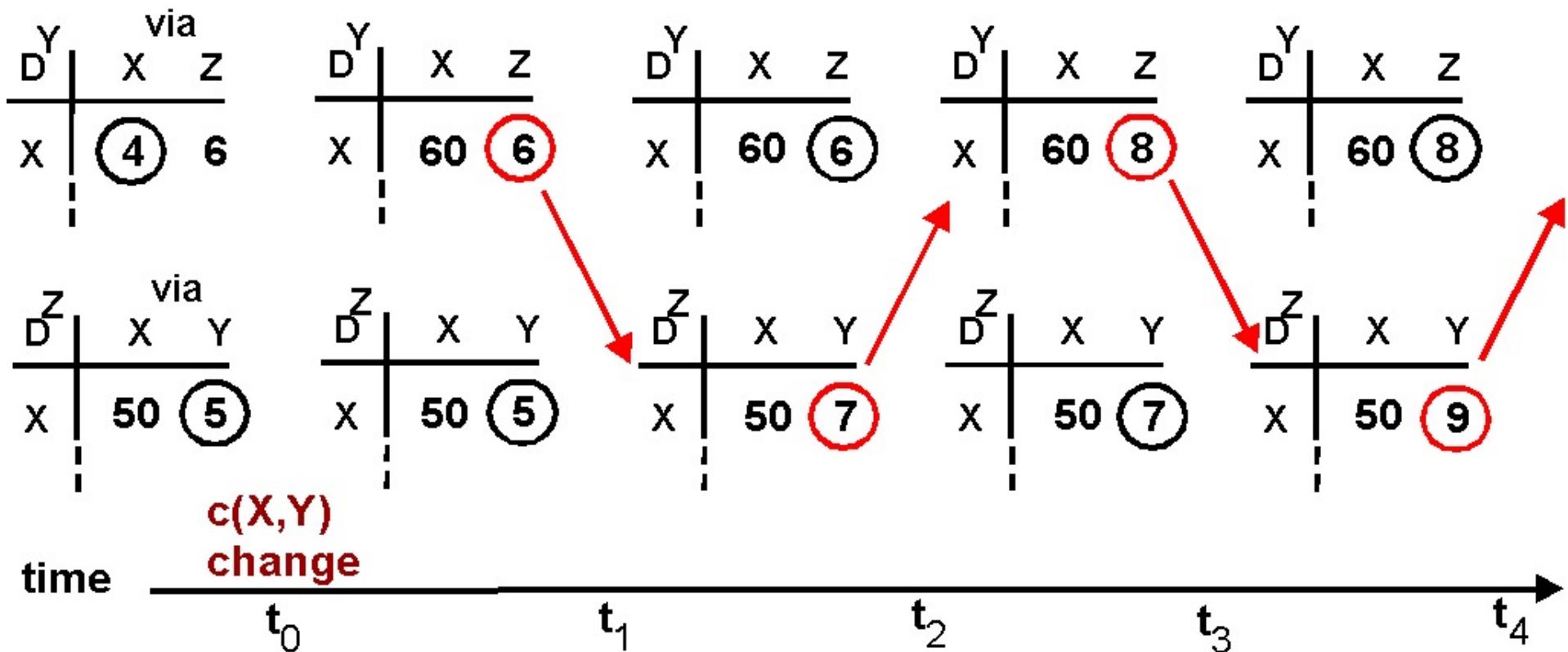
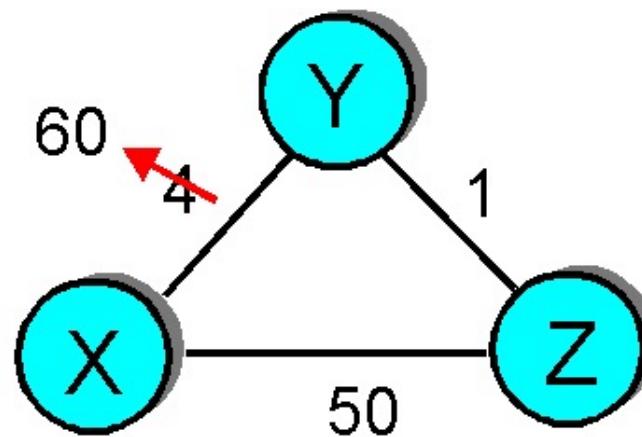
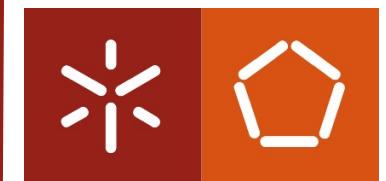
$t_1$

$t_2$

$t_3$

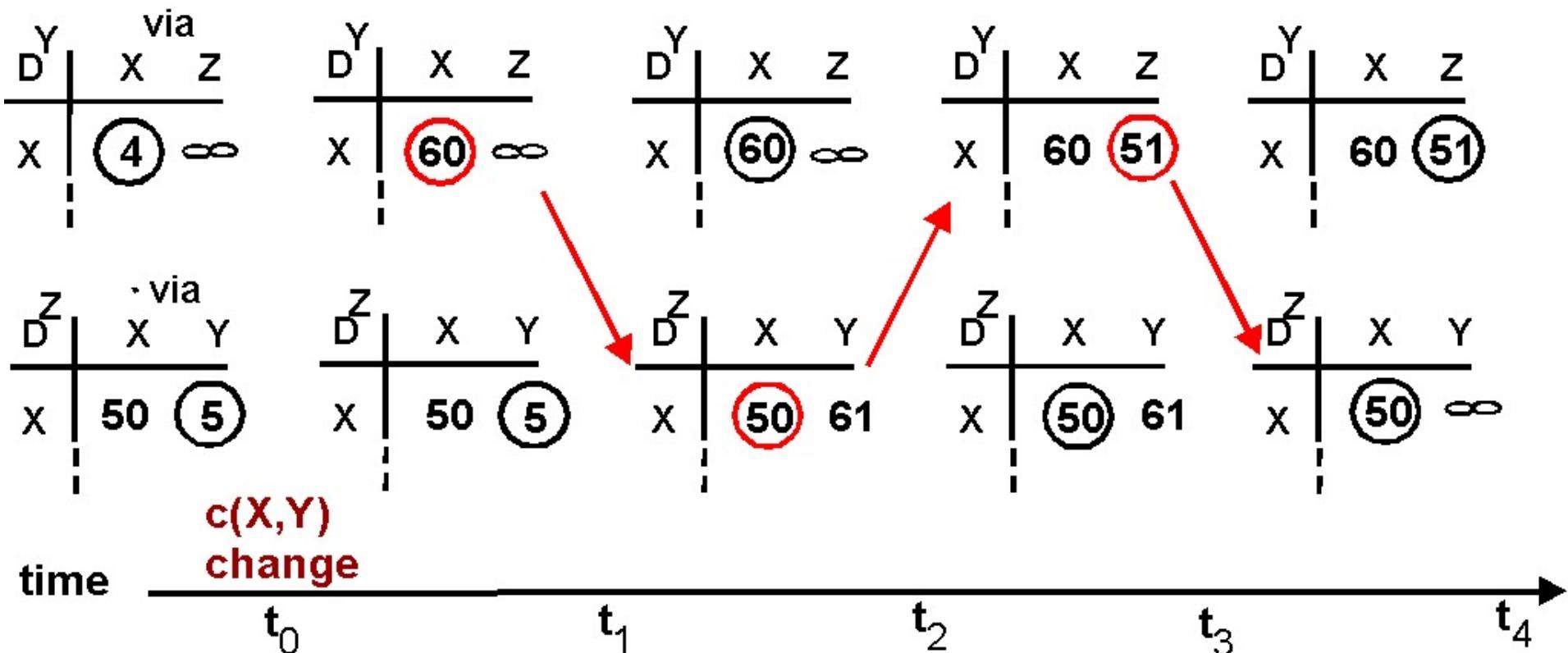
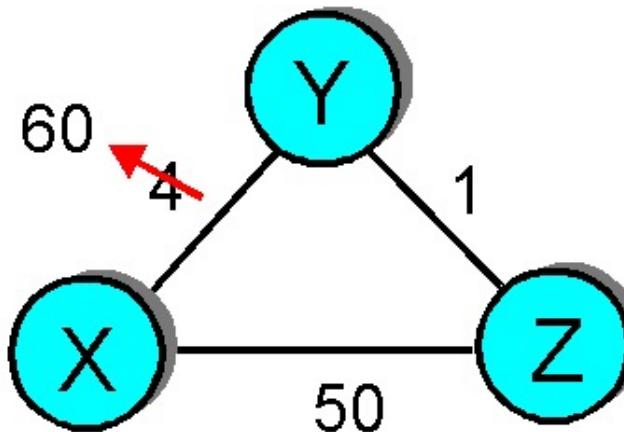
$t_4$

# DVA: Bad News... Count to Infinity!





## DVA: Bad News... Count to Infinity!





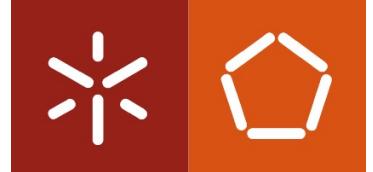
## ● Soluções

- Divisão do horizonte

**Se x aprendeu rota para z com y, nunca ensina essa rota a y!**

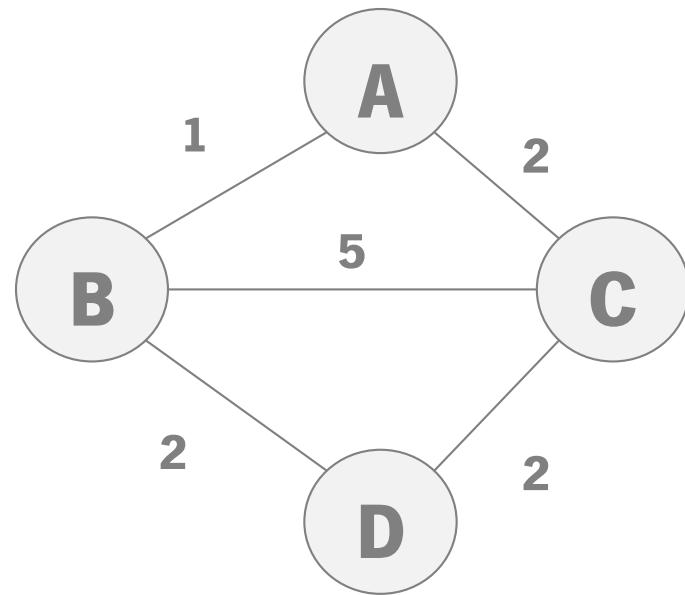
- Envenenamento do percurso inverso (*poison reverse*)

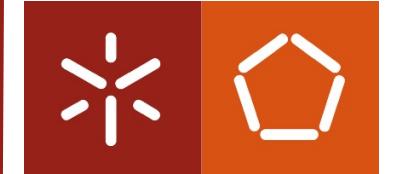
**Se x aprendeu rota para z com y, então “mente” a y dizendo-lhe que o custo da sua rota para z é infinito!  $Dx(z) = \text{Infinito}$**



# Exercício

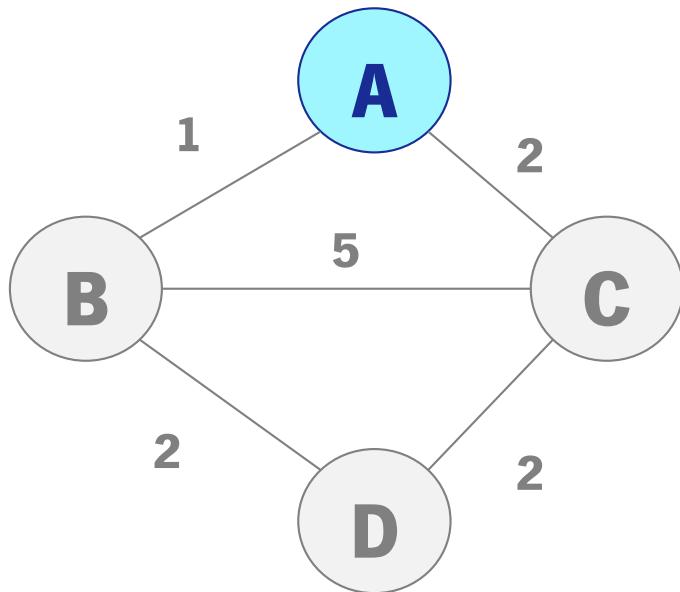
- Qual a tabela de distâncias final do nó A?
- Nessas circunstâncias, que anúncios faz/faria A para os seus vizinhos, usando divisão do horizonte e envenenamento do percurso inverso?



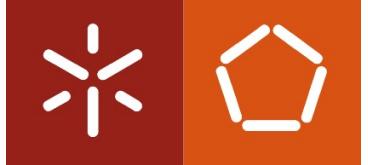


# Exercício

- Qual a tabela de distâncias final do nó A?
- Nessas circunstâncias, que anúncios faz/faria A para os seus vizinhos, usando divisão do horizonte e envenenamento do percurso inverso?



$D^A$	B	C
B		
C		
D		

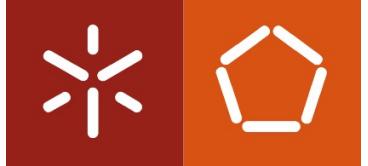


- **Sobrecarga introduzida pela mensagens de controle**

- nos algoritmos de estado de ligação todos os nós necessitam de conhecer o custo de todas as ligações, por isso sempre que o custo de uma ligação muda, uma mensagem com o novo custo tem que ser enviada para todos os nós
- nos algoritmos de vector de distâncias a mudança do custo de uma ligação só provoca o envio de mensagens se resultar na mudança da tabela de encaminhamento

- **Convergência**

- os algoritmos de estado de ligação convergem mais depressa mas, com algumas métricas estão sujeitos a oscilações;
- em contrapartida os algoritmos de vector de distâncias convergem lentamente, podem apresentar ciclos enquanto não convergem e sofrem do problema da contagem até ao infinito.

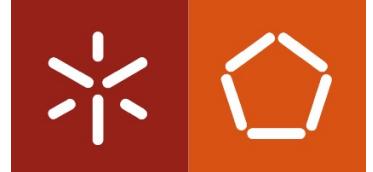


## ● Robustez

- nos algoritmos de estado de ligação cada encaminhador calcula a sua tabela de encaminhamento usando a base de dados topológica, de forma independente dos outros encaminhadores. Isso confere a este tipo de algoritmos robustez
- nos algoritmos de vector de distâncias se algum encaminhador estiver a calcular mal a sua tabela de encaminhamento, os erros cometidos vão-se propagar aos outros encaminhadores da topologia

## ● Recursos computacionais

- os algoritmos de estado de ligação são mais exigentes do que os algoritmos de vector de distâncias, quer em termos de memória (base de dados topológica versus tabela de distâncias) , quer em termos de capacidade de processamento



# Protocolos

- **Baseados Estado da Ligação:**

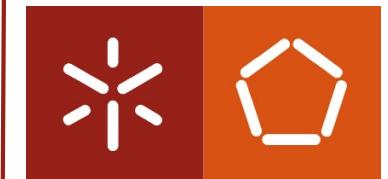
- OSPF – Open Shortest Path First
- OSI ISIS – OSI Intermediate System to Intermediate System Routing Protocol

- **Baseados no Vector Distância**

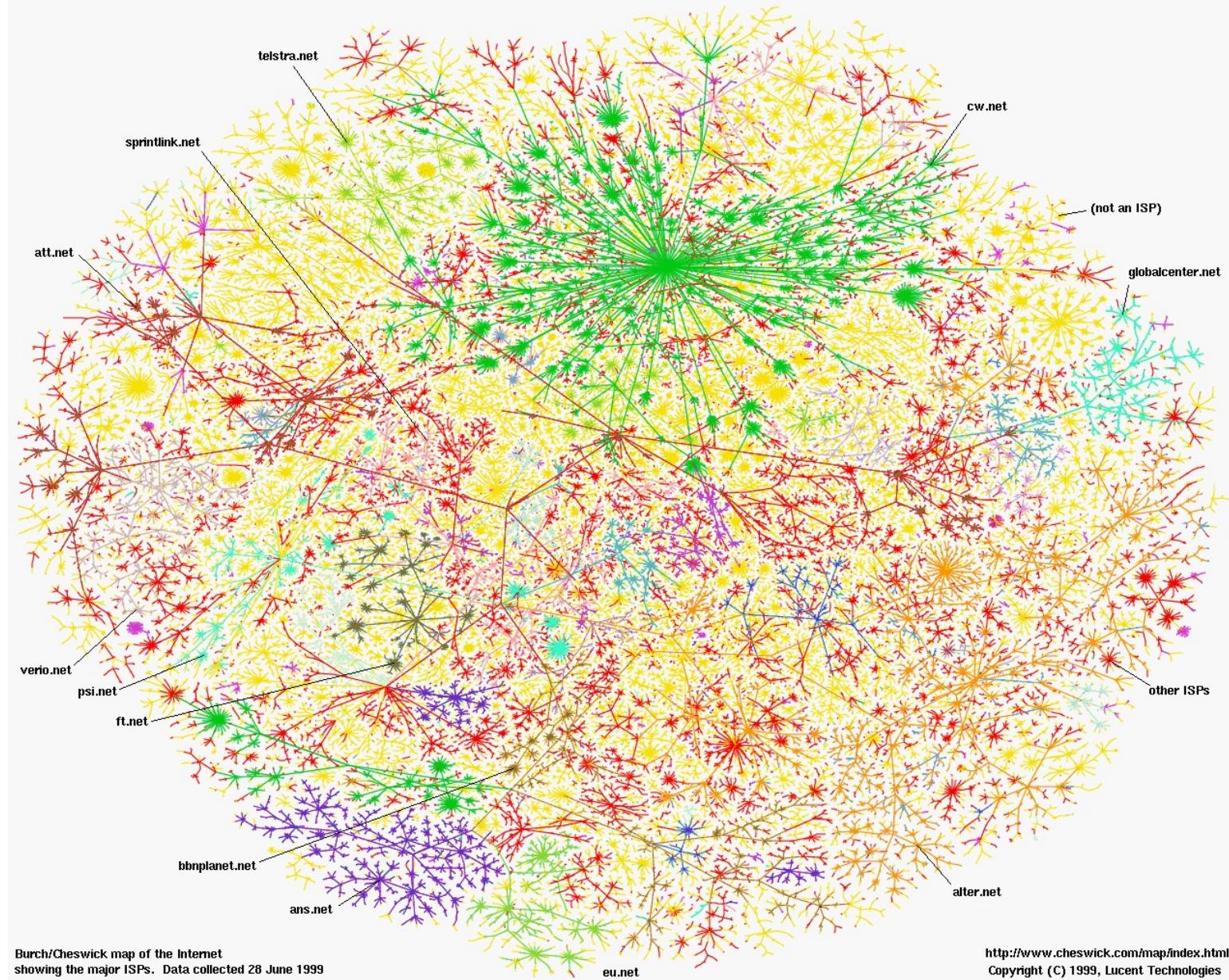
- RIP – Routing Information Protocol
  - Existe em todos os sistemas operativos (routed)
- IGRP – Interior Gateway Routing Protocol (CISCO)
- EIGRP – Extended IGRP (CISCO)

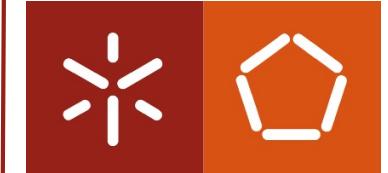
- **Implementações Open Source**

- Zebra → Quagga (para Unix e Unix like systems)



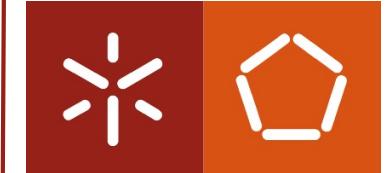
# Encaminhamento na Internet





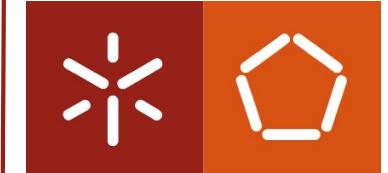
# Encaminhamento na Internet

- Por razões de escala e de autonomia administrativa, a internet não pode ser encarada como uma topologia de rede onde todos os encaminhadores executam o mesmo algoritmo de encaminhamento para encontrar os melhores caminhos para todos os destinos possíveis
  - O número de encaminhadores é demasiado grande o que torna a sobrecarga necessária ao cálculo, armazenamento e comunicação da informação de encaminhamento demasiado grande;
  - Idealmente uma organização deveria poder escolher o algoritmo de encaminhamento que deseja utilizar nas suas redes;

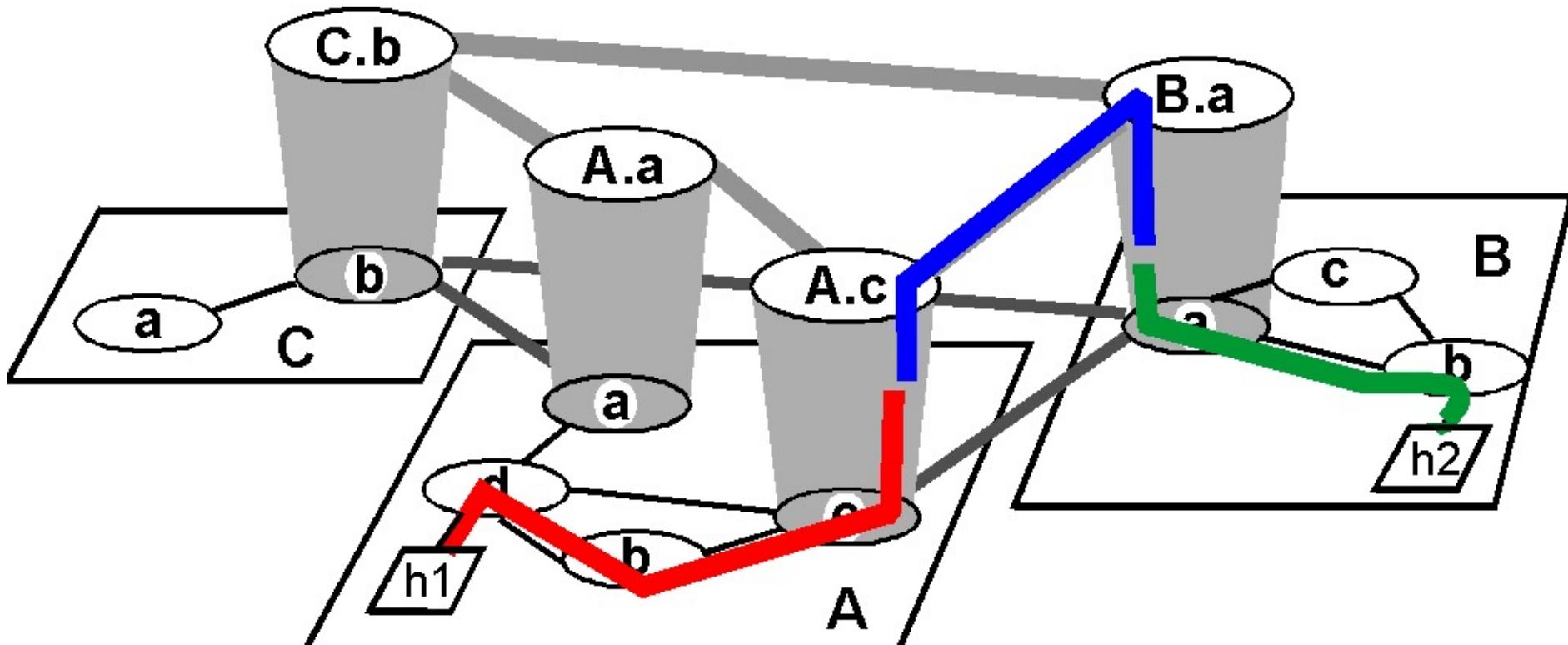


# Encaminhamento na Internet

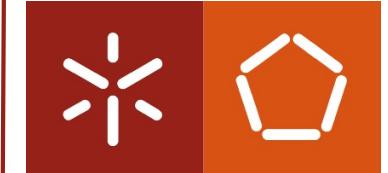
- Estes problemas são resolvidos agregando os encaminhadores em Sistemas Autónomos (AS – Autonomous Systems).
  - Os encaminhadores dentro de um mesmo sistema autónomo utilizam todos o mesmo algoritmo de encaminhamento (LS ou DV) e possuem informação acerca de todos os encaminhadores que fazem parte do sistema autónomo.
  - Os protocolos de encaminhamento que se utilizam no interior de um sistema autónomo designam-se por protocolos intra-domínio (***intradomain routing protocols***) ou internos (***IGP - Interior Gateway Protocol***)
- Para interligar os diferentes Sistemas Autónomos entre si é necessário utilizar pelo menos um encaminhador por Sistema Autónomo e com eles constituir uma rede de “nível superior”.
  - Esses encaminhadores além de executarem o protocolo intra-domínio, utilizam um protocolo de encaminhamento inter-domínio (***interdomain routing protocol***) ou externos (***EGP – Exterior Gateway Protocol***)



# Encaminhamento na Internet

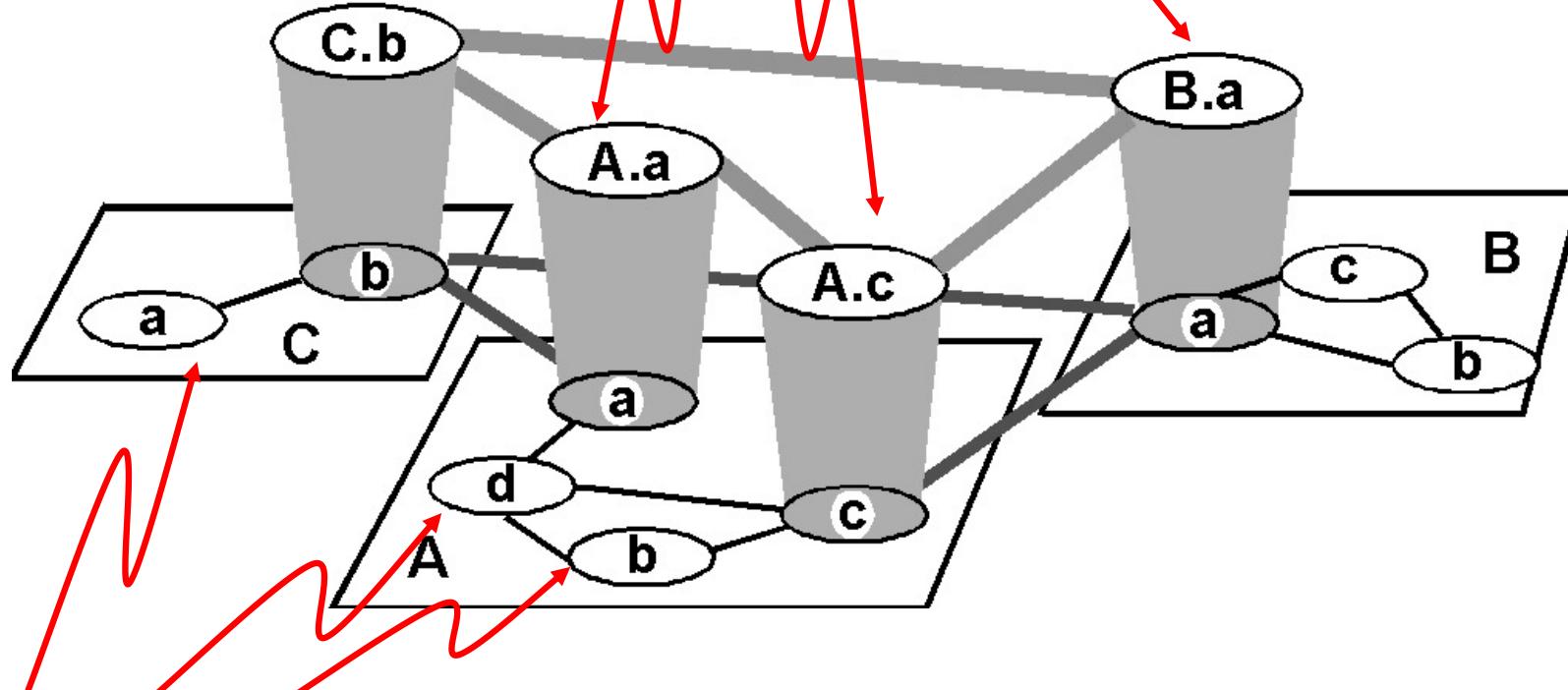


Fonte: Computer Networking: A Top-Down Approach Featuring the Internet, J. Kurose, Addison-Wesley, 2001



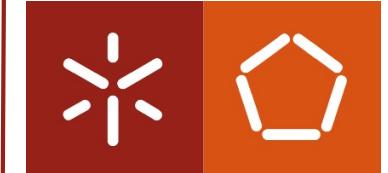
# Encaminhamento na Internet

Inter-AS border (exterior gateway) routers



Intra-AS interior (gateway) routers

Fonte: Computer Networking: A Top-Down Approach Featuring the Internet, J. Kurose, Addison-Wesley, 2001



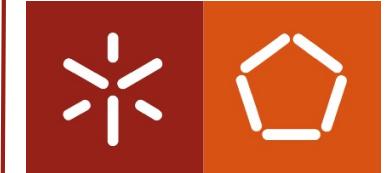
# Encaminhamento na Internet

- **Número de Sistema Autónomo**

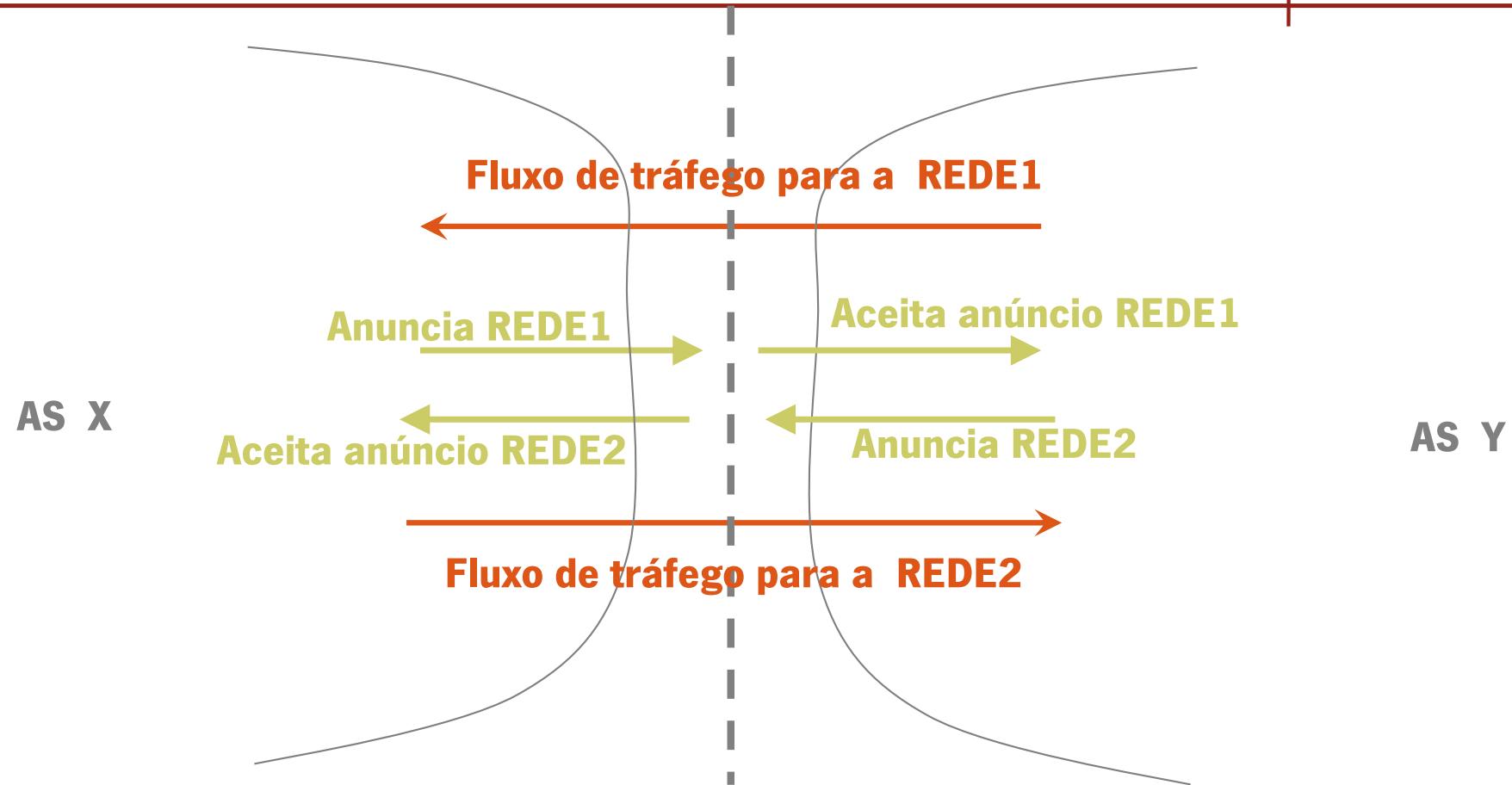
- Os números de sistema autónomo podem ser privados (AS 64512 até ao AS 65535) ou públicos (atribuídos pelo IANA, ou autoridades regionais, como o RIPE na Europa)
- Usado nas trocas de informação de encaminhamento com os sistemas autónomos vizinhos

- **Sistemas autónomos que fazem negócio com a conectividade também se designam por ISP (*Internet Service Providers*)**

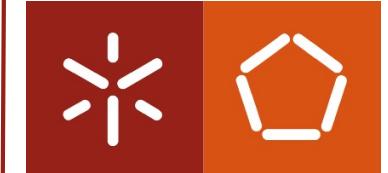
- Estabelecem acordos de parceria entre si (*peering agreements*)
  - São apenas de troca de rotas se estão ao mesmo nível
- Se não estão ao mesmo nível o de nível mais baixo (*downstream*) é *cliente* e do nível acima (*upstream*) é *fornecedor*



# Encaminhamento na Internet



- A comunicação entre a REDE1 e a REDE2 é possível se e só se:
  - (1) REDE1 anunciada por ASX, (2) anúncio aceite por ASY,
  - (3) REDE2 anunciada por ASY, (4) anúncio aceite por ASX



# Encaminhamento na Internet

## ● Protocolos IGP

- Usam processos automáticos de descoberta e troca de informação
- Todos os encaminhadores são de confiança, sujeitos à mesma administração e às mesmas regras
- As rotas e outra informação de encaminhamento pode ser difundida livremente entre todos os encaminhadores (todos têm a mesma visão da rede)

*OSPF e ISIS → Estado da Ligação  
RIP e EIGRP → Vector Distância*

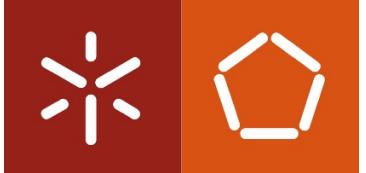
## ● Protocolos EGP

- As relações com os pares são previamente definidas e configuradas manualmente
- A conectividade com redes externas é definida por políticas (divulgação e aceitação de rotas, preferências, etc)
- Define limites administrativos

*BGP – Border Gateway Protocol*

*→ Vector Distância que associa a cada prefixo o caminho AS completo*

# Encaminhamento Intra-Domínio *versus* Encaminhamento Inter-domínio



## Políticas:

- No encaminhamento inter-domínio é fundamental ter o controle sobre a forma como o encaminhamento é efectuado. Por exemplo, a decisão de não encaminhar determinado tipo de tráfego através de um AS, tem de ser possível.
- No encaminhamento intra-domínio as decisões “políticas” de encaminhamento assumem pouca importância, uma vez que todos os nós estão sob a mesma autoridade administrativa.

## Escala:

- O encaminhamento hierárquico reduz o tamanho das tabelas de encaminhamento e a quantidade e tamanho das mensagens de actualização da informação de encaminhamento.

## Desempenho:

- No encaminhamento intra-domínio o desempenho é a preocupação principal, ao passo que no encaminhamento inter-domínio tem um papel secundário, sendo ultrapassado pelas políticas.