Curso de Redes de Computadores

Adriano Mauro Cansian
adriano.cansian@unesp.br

Capítulo 4
Camada de Rede
(Parte 3)

Referência para estudo:

REDES DE COMPUTADORES E A INTERNET - 6ª Edição

James F. Kurose e Keith W. Ross. Edit. Pearson 2014.

ISBN: 9788581436777.

Capítulo 4 – **Camada de Rede** (6a. Ed.)

Licença de uso deste material (1)

Estes slides foram produzidos e são disponibilizados sob a licença "Creative Commons Atribuição Não Comercial SemDerivações 4.0 Internacional (CC BY-NC-ND 4.0)".

Como dar o crédito apropriado:
Para qualquer reutilização ou distribuição, você deve deixar claro a terceiros os termos da licença a que se encontra submetida esta obra, e seu autor. A melhor maneira de fazer isso é citando a fonte e contato. Por exemplo:

Fonte: Adriano Mauro Cansian. "**Redes de Computadores I – Camada de Rede**", São Paulo, SP. (2025).
Contato: adriano.cansian@unesp.br. Licenciamento Creative Commons (CC BY-NC-ND 4.0)

Fair Use: Este material pode usar citações e figuras de outros autores. Direitos reservados aos devidos autores. Todas as citações feitas em caráter didático.

Avisos: Você não tem de cumprir com os termos da licença relativamente a elementos do material que estejam no domínio público ou cuja utilização seja permitida por uma exceção ou limitação que seja aplicável.
Não são dadas quaisquer garantias. A licença pode não lhe dar todas as autorizações necessárias para o uso pretendido. Por exemplo, outros direitos, tais como direitos de imagem, de privacidade ou direitos morais, podem limitar o uso do material.

Licença de uso deste material (2)

- **Você tem o direito de:**

Compartilhar - copiar e redistribuir o material em qualquer suporte ou formato.

- **De acordo com os termos seguintes:**
  - **(BY) Atribuição** - Você deve dar o crédito apropriado, prover um *link* para esta licença e indicar se mudanças foram feitas. Você deve fazê-lo em qualquer circunstância razoável, mas de nenhuma maneira que sugira que o licenciante apoia você ou o seu uso.
  - **(NC) Não Comercial** - Você não pode usar o material para fins comerciais.
  - **(ND) Sem Derivações** - Se você remixar, transformar ou criar a partir do material, você não pode distribuir o material modificado.
  - **Sem restrições adicionais** - Você não pode aplicar termos jurídicos ou medidas de caráter tecnológico que restrinjam legalmente outros de fazerem algo que a licença permita.

Interconexão de redes e roteamento

Já sabemos que:

- Internet = *INTERNETworking*
  - Interconexão de redes.
- Comunicação fim-a-fim sobre redes:
  - Em grande escala.
    - Deve ser possível escalar (crescer).
  - Heterogêneas.
    - Pode ser usados diversos protocolos e tecnologias de enlace.
  - Organizadas em federação hierárquica.
    - Cada instituição é “dona” de uma parte da rede.

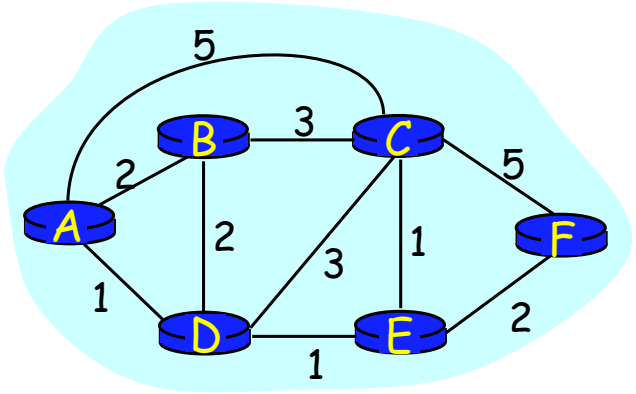
Equipamentos de Interconexão em várias camadas:

- **Roteadores** na **camada de rede** (*layer3*).
- *Switches* na camada de acesso ao meio (*layer2*).
  - Responsáveis pela interconexão na rede local (LAN).
  - Leva o pacote na rede local até o próximo salto da rota.
  - Vistos no próximo Capítulo – Camada de Enlace

Roteamento

Protocolo de roteamento

Meta: **determinar melhor caminho** (sequência de roteadores) pela rede, desde a origem até o destino.

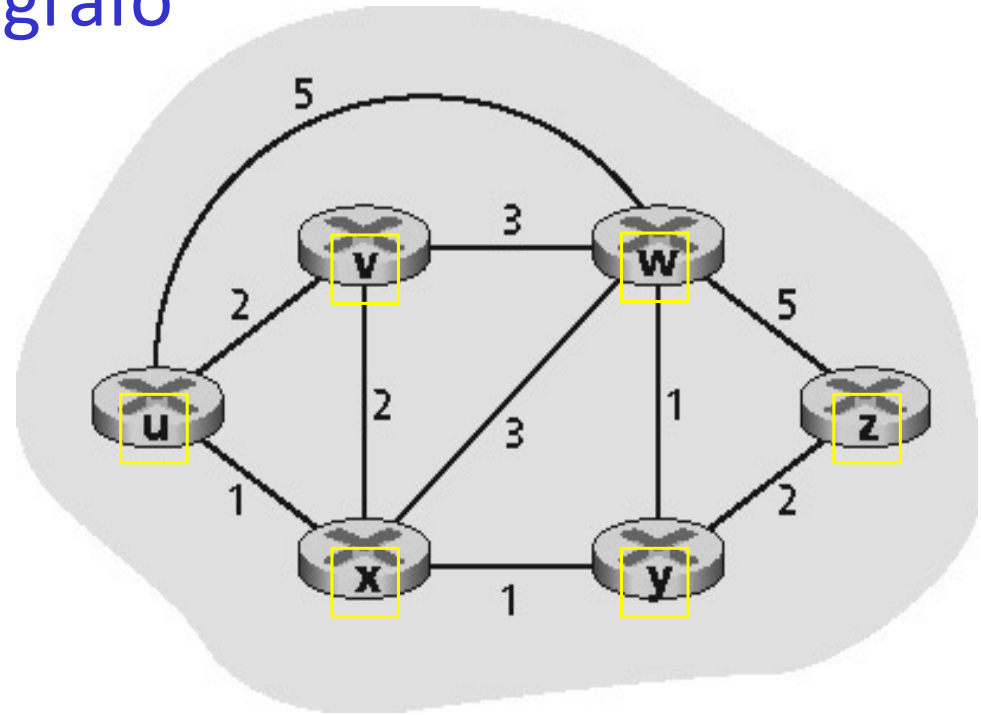


Abstração em **grafo**:

- **Nós** são **roteadores**.
- **Arestas** são os links físicos.
  - Fibras ópticas, cabos coaxiais, 4G/5G, satélite, microondas, etc...
  - Custos do enlace: atraso, financeiro, risco, congestionamento, etc...

- Achar o "**melhor**" caminho :
  - Geralmente significa **caminho de menor custo**.
  - **De forma geral: custo é medido em distância ou em número de saltos.**
  - Outras definições são possíveis.
  - "Custo" não é necessariamente $$.

Abstração em grafo

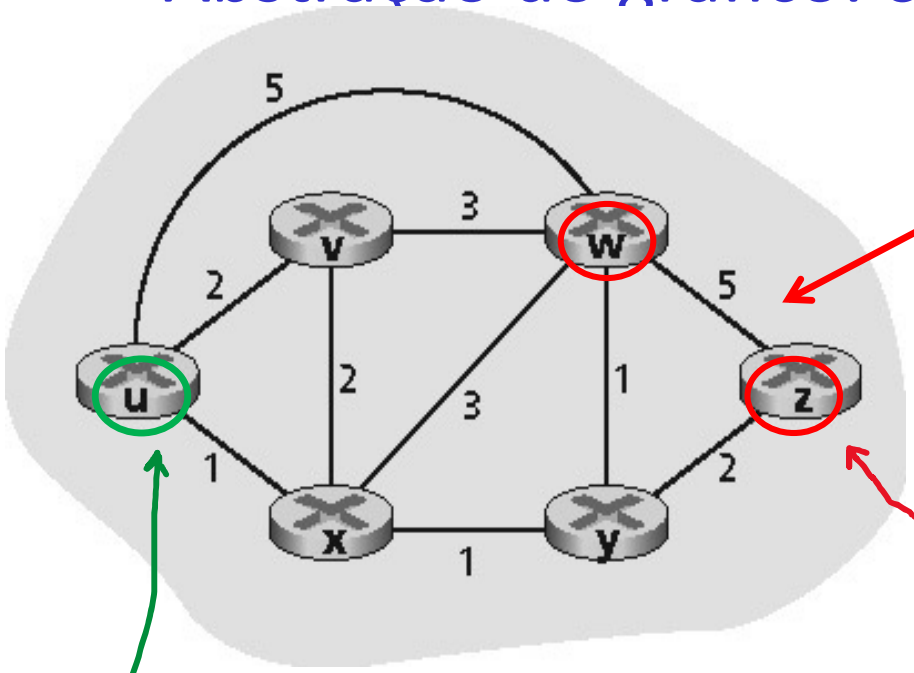


Grafo: $G = (N,E)$

Conjunto de roteadores: $N = \{ u, v, w, x, y, z \}$

Conjunto de links: $E = \{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z), (u,w) \}$

Abstração do gráfico: **custo**



- c(*a*,*b*) = custo do link (*a*-*b*)
  - Ex: c(w, z) = 5
- Pode ser fixo, relativo à distância, ou então inversamente proporcional à **largura de banda**.

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

Qual é o caminho de **menor custo** entre *u* e *z* ?

Algoritmo de roteamento: é algoritmo que **encontra o caminho de menor custo.**
Ou **o melhor caminho possível**.

Tipos de algoritmos de roteamento

- Classificações principais.
  1. Quanto ao tipo da informação.
  2. Quanto à mudança das rotas.
  3. Quanto à sensibilidade à carga.
- Veremos em seguida...

Classificação de Algoritmos de Roteamento (1)

Classificação 1 – Quanto ao Tipo da Informação:

Informação **Global:**

- **Todos os roteadores participantes** têm **informações completas** da topologia, distância e custos dos enlaces.
- Algoritmos de "estado de enlaces" (***link state - LS***).

Informação **Descentralizada:**

- **Roteador conhece os vizinhos diretos**, e **custos** até eles.
- Processo iterativo de cálculo, e troca de informações com vizinhos.
- Algoritmos de "vetor de distâncias" (***distance vector - DV***).

Classificação de Algoritmos de Roteamento (2)

Classificação 2 - Quanto à mudança das rotas:

Tipo Estático:

- Usado quando as **rotas mudam lentamente** ou raramente.
  - Tipicamente para **sistemas de borda** (*edge routers*), ou que possuem um ou poucos *links* de entrada/saída.

Tipo Dinâmico:

- Usado quando as rotas mudam mais rapidamente.
  - Tipicamente para **sistemas de núcleo** (*core routers*), com vários *links*.
  - **Utilizam atualização periódica automática,** em resposta a mudanças nos custos, ou estado dos enlaces (*link down* ou *up*).

Classificação de Algoritmos de Roteamento (3)

Classificação 3 - Quanto à Sensibilização à carga.

Insensível à carga:

- **Não** levam em consideração congestionamento.
- **Custo do enlace não reflete seu nível de congestionamento**.
- É o caso dos algoritmos usados na Internet atualmente: **RIP, OSPF e BGP**.

Sensível à carga:

- Custo de enlaces variam dinamicamente, para refletir o nível atual de congestionamento nos enlaces subjacentes.
  - Se houver custo alto associado com um link que está congestionado, o algoritmo deveria escolher rotas que evitam este enlace.
  - Algoritmos antigos eram sensíveis à carga, mas foram **descartados**.
    - Houve dificuldades técnicas em sua aplicação.

Algoritmos de roteamento

Introdução aos algoritmos mais usados

Algoritmos de roteamento

Algoritmos principais:

- *Link state:*
  - Informação **Global**.
  - Algoritmo de Dijskstra.
- *Distance vector* :
  - Informação **Descentralizada**
  - Equação de Bellman-Ford.

Link State

Algoritmo de Dijkstra

Roteamento *Link-state*

Algoritmo de Dijkstra usado para calcular melhor caminho.

- A topologia da rede e custo de **todos** os enlaces **são conhecidos por todos os nós**.
  - **Todos os nós têm a mesma informação.**
  - Implementado via mensagem "*link state broadcast*" (visto mais adiante).
  - Todos os nós têm uma visão **igual** e **completa** da rede.
- Calcula caminhos de **menor custo** de uma **origem** para **todos** os outros nós **destinos**.
  - Obtém uma **tabela de rotas** a partir daquela **origem**.
- Na prática, quem usa *Link-State* na Internet:
  - Algoritmo **OSPF** (visto mais adiante).

Notação do algoritmo de Dijkstra (1)

- `C(i,j):` custo do enlace do nó ***i*** até o nó ***j***.
  - Custo é **infinito** se não houver ligação entre ***i*** e ***j.***
- `D(v):` **menor custo** atual entre a origem e o destino ***V***, até a presente iteração do algoritmo.

Notação do algoritmo de Dijkstra (2)

- `C(i,j):` custo do enlace do nó ***i*** até o nó ***j***.
  - Custo é infinito se não houver ligação entre ***i*** e ***j***.
- `D(v):` **menor custo** atual entre a origem e o destino ***V***, até a presente iteração do algoritmo.
- `p(v):` **Predecessor** de ***v*** → **nó vizinho** a ***v*** ao longo do **caminho de menor custo** atual, da origem até o destino ***v.***
  - ***p(A)*** lê-se: "predecessor de ***A***" ou "passando por ***A***".
- `N':` subconjunto de nós → irá formando o conjunto de menor custo.
  - ***v*** pertence a ***N'*** se o caminho de menor custo entre a fonte e ***v*** for inequivocamente conhecido.

Algoritmo de Dijsktra básico: (1)

1. **Inicialização**: Define a distância inicial para o nó de partida como ***0*** e para todos os outros nós como **infinito**. Também marca todos os nós como não visitados.
2. **Visita de Nós**: Seleciona o nó não visitado com a menor distância conhecida, marca-o como visitado, e **atualiza as distâncias** dos seus vizinhos.
3. **Atualização de Distâncias**: Para cada vizinho ***v*** do nó atual, **calcula a distância total do nó de partida até *v* passando pelo nó atual**. Se essa nova distância for menor que a distância registrada anteriormente para ***v***, atualiza-a.
4. **Repetição**: Repete os passos 2 e 3 até que todos os nós tenham sido visitados ou que o menor valor de distância entre os nós não visitados seja infinito (o que significa que os nós restantes são inacessíveis).
5. **Resultado**: O algoritmo garante que a menor distância de um nó para outro será encontrada se existir um caminho.

Algoritmo de Dijsktra básico: (2)

1. **Inicialização**: Define a distância inicial para o nó de partida como ***0*** e para todos os outros nós como **infinito**. Também marca todos os nós como não visitados.
2. **Visita de Nós**: Seleciona o nó não visitado com a menor distância conhecida, marca-o como visitado, e **atualiza as distâncias** dos seus vizinhos.
3. **Atualização de Distâncias**: Para cada vizinho ***v*** do nó atual, **calcula a distância total do nó de partida até *v* passando pelo nó atual**. Se essa nova distância for menor que a distância registrada anteriormente para ***v***, atualiza-a.

u que o

ue os

ntrada

Se esta distância for **menor** que a distância registrada anteriormente, atualiza a distância.

$$D(v) = \min\{ D(v), D(w) + c(w,v) \}$$

Novo custo para ***v*** → é **ou** o velho custo para *v*, **ou** o menor custo de caminho conhecido para ***w*** (antecessor de ***v***), mais o custo de ***w*** até ***v***.

Algoritmo de Dijsktra básico: (3)

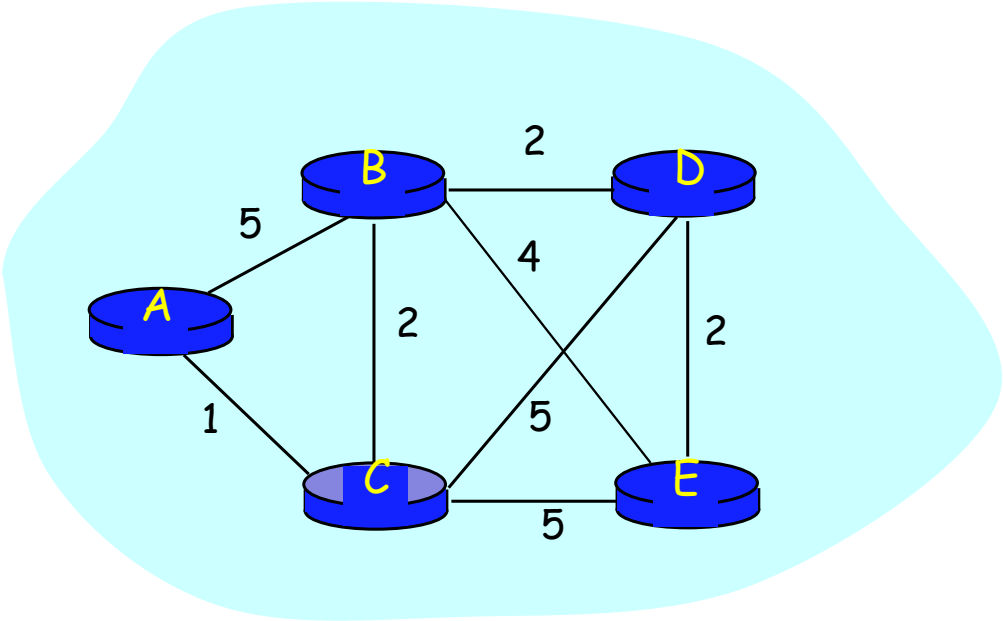
1. **Inicialização**: Define a distância inicial para o nó de partida como 0 e para todos os outros nós como infinito.
2. **Visita de Nós**: Seleciona o nó não visitado com a menor distância conhecida, marca-o como visitado e **atualiza as distâncias** dos seus vizinhos.
3. **Atualização de Distâncias**: Para cada vizinho v do nó atual, **calcula a distância total do nó de partida até ele**.

---

4. **Repetição**: Repete o processo até que todos os nós tenham sido visitados.

- O algoritmo garante que a menor distância de um nó para outro será encontrada se existir um caminho.

Exemplo do cálculo do Algoritmo de Dijsktra para uma rede com 5 roteadores

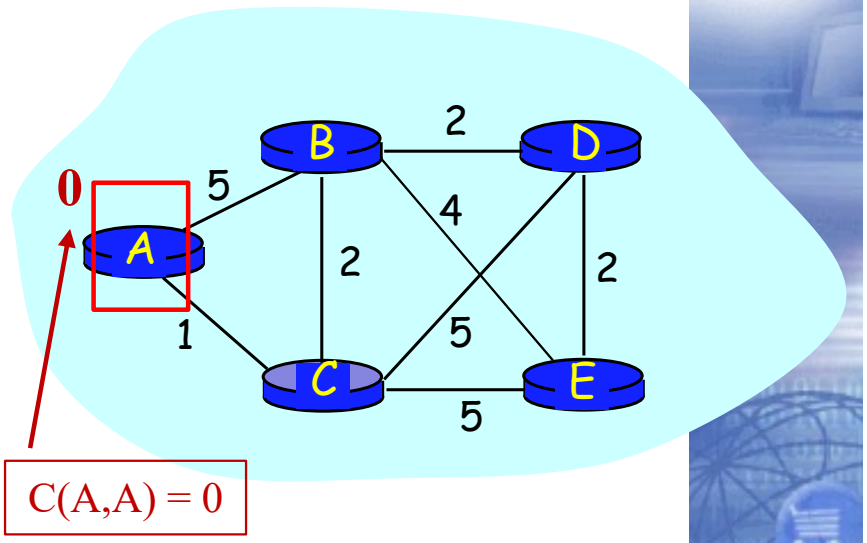


- Quantas Iterações fazer?

  Sendo ***n*** o número de vértices, o No. de iterações será igual a ***n.***

  Então, nesse exemplo, *n=5*.

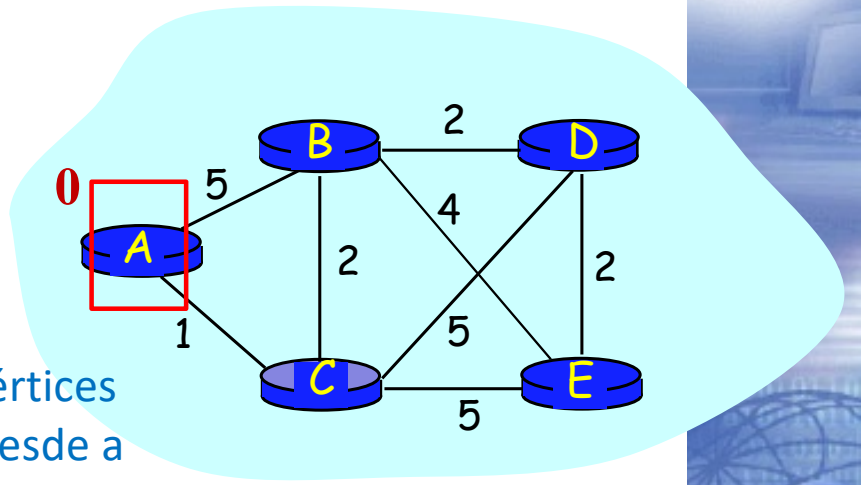
- Inicialize todas as distâncias dos vértices como **infinitas**, exceto o **vértice de origem**, que terá distância zero.
- Inicialize a lista de vértices visitados como vazia.



Origem: A

	A	B	C	D	E
1ª. Iter. ( )	**(0,A)**	∞	∞	∞	∞
2ª. Iter. ( )					
3ª. Iter. ( )					
4ª. Iter. ( )					
5ª. Iter. ( )					

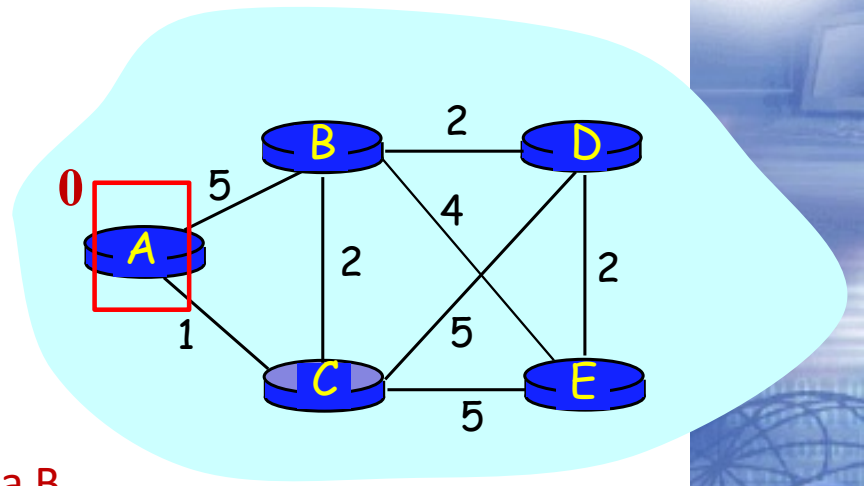
- O valor do vértice iterado fica imutável e **não poderá ser escolhido novamente para iterar**, até o fim das iterações dos demais.



Em seguida, vamos adicionar os vértices que consigo chegar diretamente desde a origem, nesse caso, ***B*** e ***C***.

	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	∞	∞	∞	∞
2ª. Iter. ( )	(0,A)				
3ª. Iter. ( )	(0,A)				
4ª. Iter. ( )	(0,A)				
5ª. Iter. ( )	(0,A)				

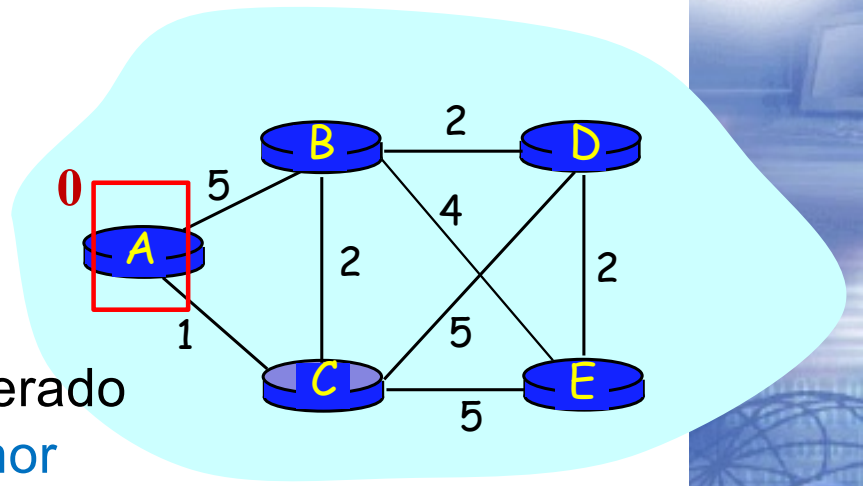
- Consigo chegar diretamente até ***B*** e ***C***.
- Custo de A até B = 5
- Custo de A até C = 1
- Os que não são acessíveis diretamente continuam com custo infinito.



Vértice A é imediatamente vizinho antes de chegar a B

	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	**(5,A)**	**(1,A)**	∞	∞
2ª. Iter. ( )	(0,A)				
3ª. Iter. ( )	(0,A)				
4ª. Iter. ( )	(0,A)				
5ª. Iter. ( )	(0,A)				

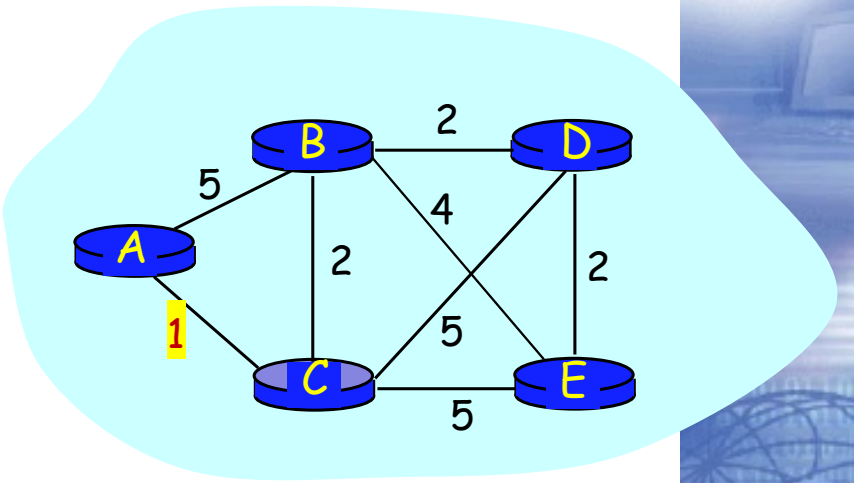
- Consigo chegar diretamente até B e C.
- Custo de A até B = 5
- Custo de A até C = 1
- Os que não são acessíveis diretamente continuam com custo infinito.



- O **próximo vértice** a ser iterado será aquele que tem o menor custo entre eles: B, C, D e E.

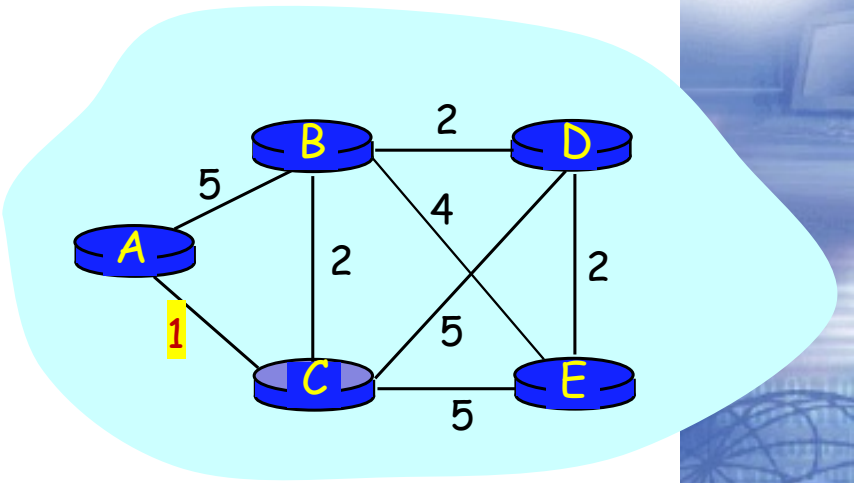
	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	**(5,A)**	**(1,A)**	∞	∞
2ª. Iter. ( )	(0,A)				
3ª. Iter. ( )	(0,A)				
4ª. Iter. ( )	(0,A)				
5ª. Iter. ( )	(0,A)				

- O **próximo vértice** a ser iterado será aquele que tem o menor custo.
- Min(5,1)= C



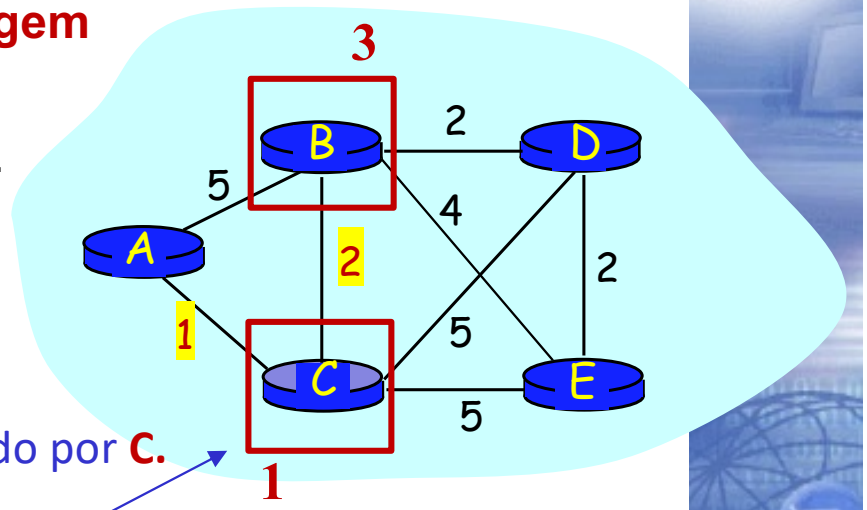
	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. ( )	(0,A)				
3ª. Iter. ( )	(0,A)				
4ª. Iter. ( )	(0,A)				
5ª. Iter. ( )	(0,A)				

- Min(5,1)= C
- O valor do vértice iterado fica imutável até o final das iterações.



	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)		(1,A)		
3ª. Iter. ( )	(0,A)		(1,A)		
4ª. Iter. ( )	(0,A)		(1,A)		
5ª. Iter. ( )	(0,A)		(1,A)		

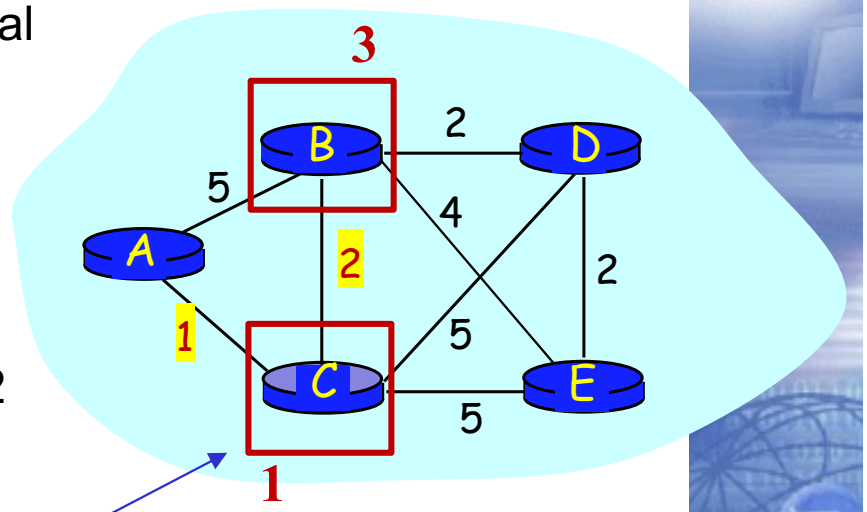
- Agora há duas opções de caminhos **para a origem A chegar até B**:
- Custo 5, diretamente conectado, ou seja, **(5,A)**.
- Custo 3, passando por C, ou seja, **(3,C)**.
- Vamos escolher o menor custo (Min{3,5})



Menor caminho de A até **B** é passando por **C.**

	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	**(3,C)**	(1,A)		
3ª. Iter. ( )	(0,A)		(1,A)		
4ª. Iter. ( )	(0,A)		(1,A)		
5ª. Iter. ( )	(0,A)		(1,A)		

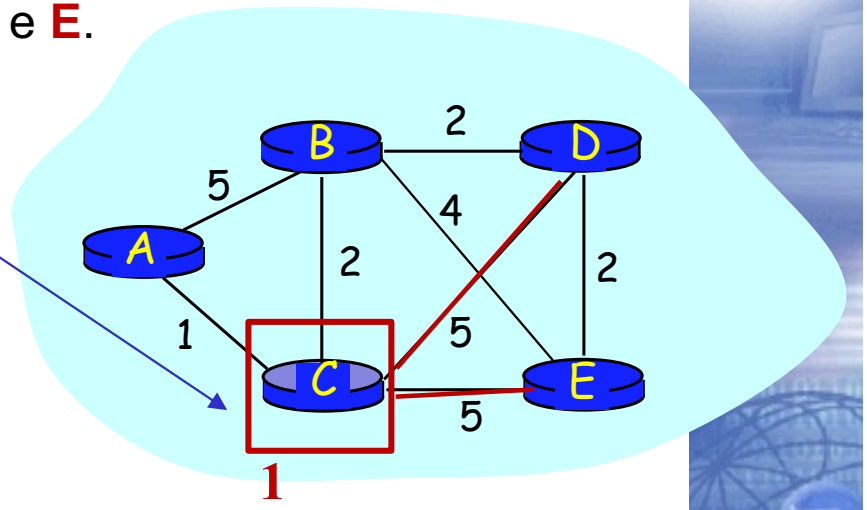
- O valor do vértice iterado fica imutável até o final das iterações.
- Qual menor custo para chegar até B?
- O menor custo é 3, pois:
  - Para chegar a **C**, o menor custo é 1.
  - Para chegar de ***C*** até ***B*** o menor custo é 2



	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	(3,C)	(1,A)		
3ª. Iter. ( )	(0,A)		(1,A)		
4ª. Iter. ( )	(0,A)		(1,A)		
5ª. Iter. ( )	(0,A)		(1,A)		

- Estamos iterando ***C***: agora consigo chegar até ***D*** e **E**.
- Qual custo para chegar até ***D*** através de ***C*** ?
- O menor custo é 6, pois:
  - Para chegar a C, o menor custo é 1.
  - Para chegar de C até D, o menor custo é 5.

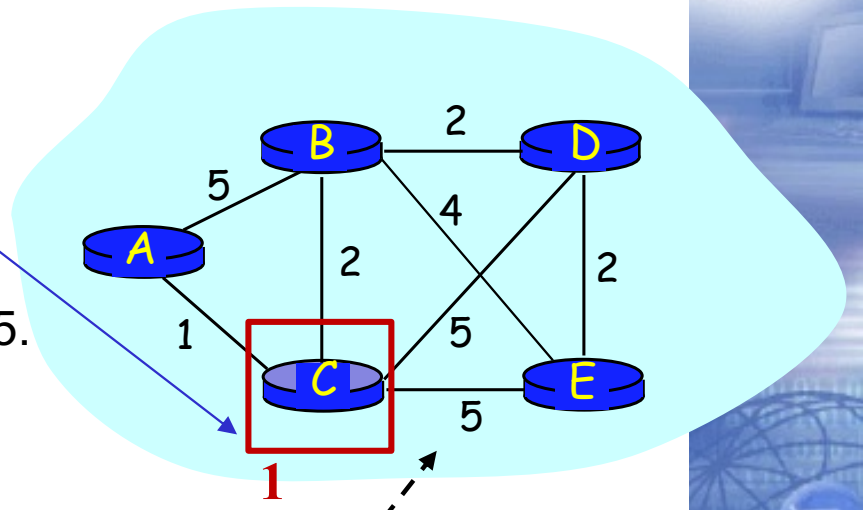
(Note que ***B*** ainda não entrou nas iterações)



	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	(3,C)	(1,A)	(6,C)	
3ª. Iter. ( )	(0,A)		(1,A)		
4ª. Iter. ( )	(0,A)		(1,A)		
5ª. Iter. ( )	(0,A)		(1,A)		

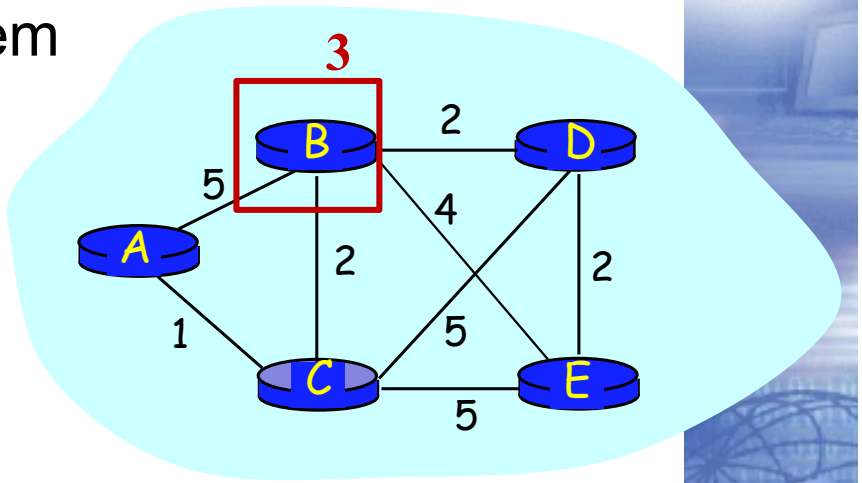
- Qual custo para chegar até ***E*** através de ***C*** ?

  (Note que ***B*** ainda não entrou nas iterações)
- O menor custo é 6, pois:
  - Para chegar a C, o menor custo é 1.
  - Para chegar de C até E, o menor custo é 5.



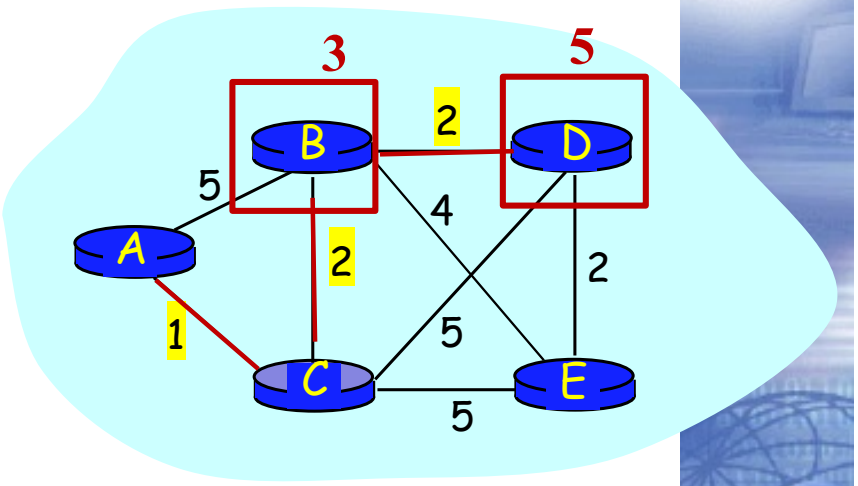
	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	(3,C)	(1,A)	(6,C)	(6,C)
3ª. Iter. ( )	(0,A)		(1,A)		
4ª. Iter. ( )	(0,A)		(1,A)		
5ª. Iter. ( )	(0,A)		(1,A)		

- Próximo vértice a ser iterado é o que tem o menor custo entre ***B, D*** e ***E***.
- Será o vértice ***B*** e seu valor ficará imutável até o final das iterações.



	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	(3,C)	(1,A)	(6,C)	(6,C)
3ª. Iter. (**B**)	(0,A)	(3,C)	(1,A)		
4ª. Iter. ( )	(0,A)	(3,C)	(1,A)		
5ª. Iter. ( )	(0,A)	(3,C)	(1,A)		

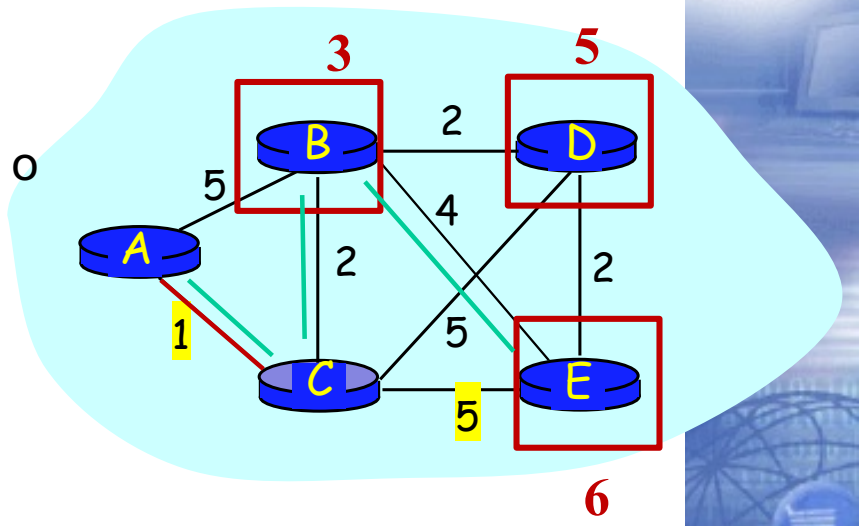
- Iterando ***B***, o menor custo até **D** passa a 5.



	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	(3,C)	(1,A)	(6,C)	(6,C)
3ª. Iter. (**B**)	(0,A)	(3,C)	(1,A)	(5,B)	
4ª. Iter. ( )	(0,A)	(3,C)	(1,A)		
5ª. Iter. ( )	(0,A)	(3,C)	(1,A)		

- Iterando ***B***, o menor custo até **D** passa a 5.
- O menor custo até ***E*** permanece 6, visto que ao iterar ***B***, resulta em custo até ***E*** de valor 7, então o **custo não é alterado**, permanecendo (6,C)

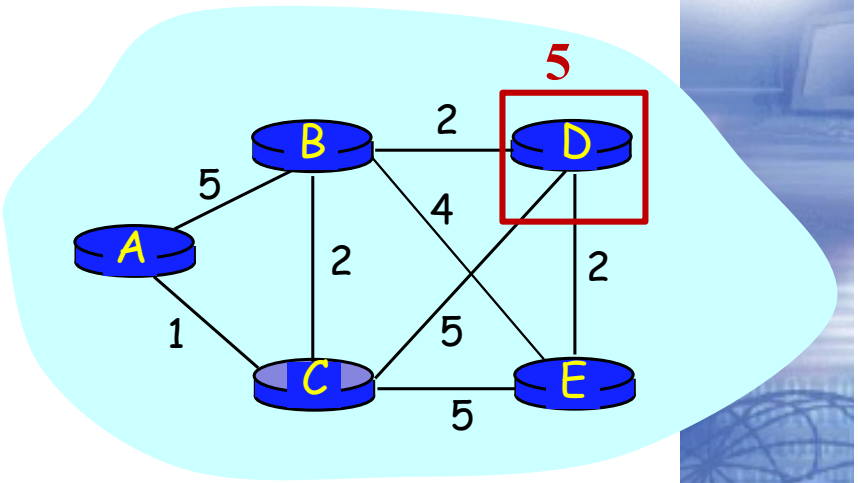
(Note que ***D*** ainda não entrou nas iterações)



	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	(3,C)	(1,A)	(6,C)	(6,C)
3ª. Iter. (**B**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)
4ª. Iter. ( )	(0,A)	(3,C)	(1,A)		
5ª. Iter. ( )	(0,A)	(3,C)	(1,A)		

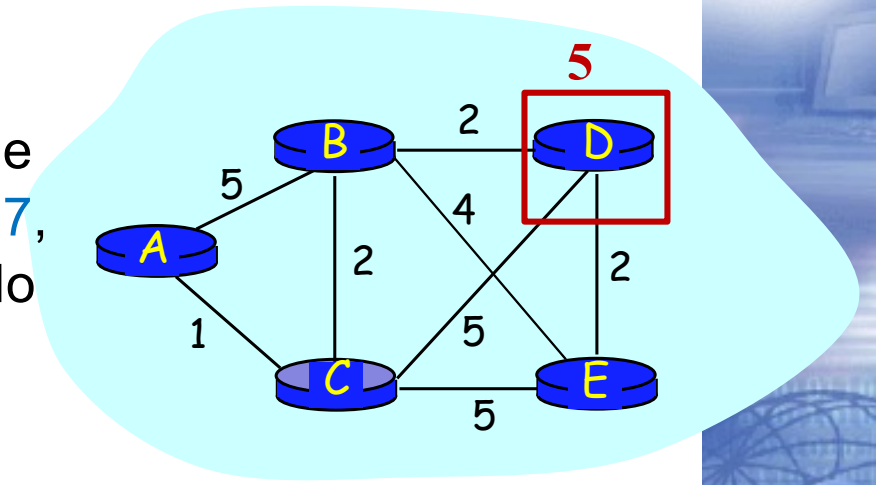
(7,B) → A,C,B

- O próximo vértice a ser iterado será aquele que tem o menor custo → ***D***.
- E seu valor ficará imutável até o final das iterações.



	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	(3,C)	(1,A)	(6,C)	(6,C)
3ª. Iter. (**B**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)
4ª. Iter. (**D**)	(0,A)	(3,C)	(1,A)	(5,B)	
5ª. Iter. ( )	(0,A)	(3,C)	(1,A)	(5,B)	

- Iterando ***D***:
- O menor custo até ***E*** permanece 6, visto que ao iterar ***D***, resulta em custo até ***E*** de valor 7, então o custo não é alterado, permanecendo (6,C)

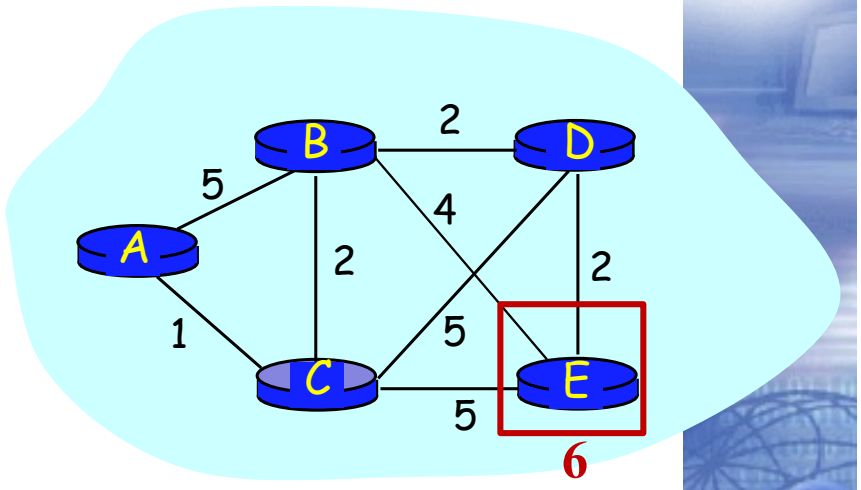


	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	(3,C)	(1,A)	(6,C)	(6,C)
3ª. Iter. (**B**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)
4ª. Iter. (**D**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)
5ª. Iter. ( )	(0,A)	(3,C)	(1,A)	(5,B)	

~~(7,D)~~

Menor custo se mantém via C.

- Finalmente, falta iterar o vértice E,
- E seu valor ficará imutável até o final das iterações.
- E chegamos ao fim de nossa tabela!

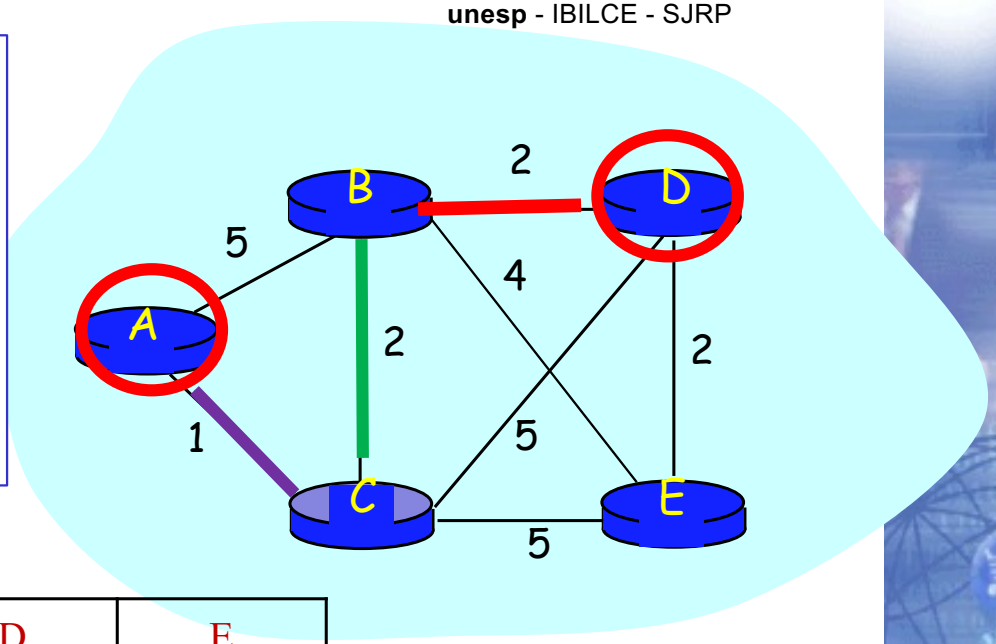


	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	(3,C)	(1,A)	(6,C)	(6,C)
3ª. Iter. (**B**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)
4ª. Iter. (**D**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)
5ª. Iter. (**E**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)

Para achar o menor caminho entre A e D:

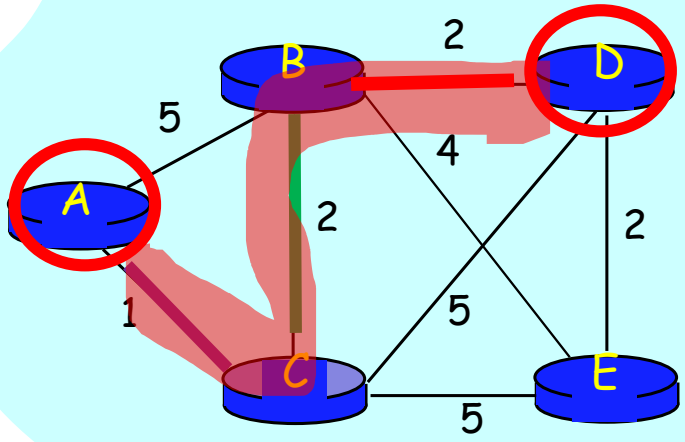
- Menor custo até D: (5,B)
- Menor custo até B: (3,C)
- Menor custo até C: (1,A)
- Portanto, menor custo A até D:

  Caminho: (A-C-B-D) com custo total 5



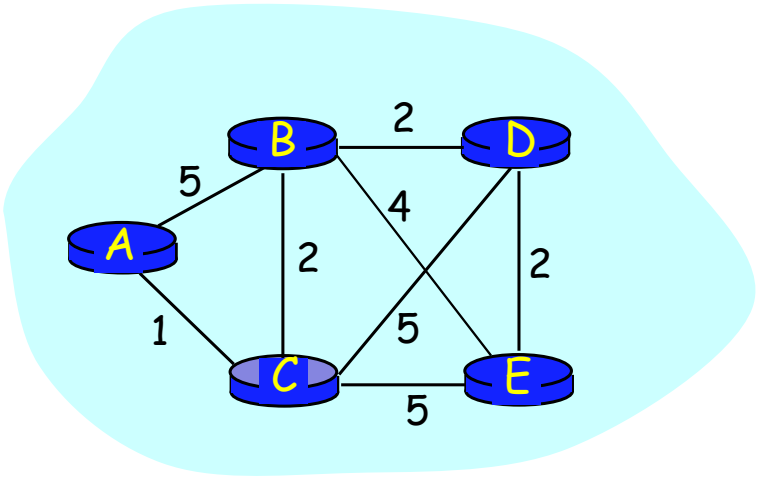
	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	(3,C)	(1,A)	(6,C)	(6,C)
3ª. Iter. (**B**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)
4ª. Iter. (**D**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)
5ª. Iter. (**E**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)

- Portanto, menor custo A até D: Caminho (A-C-B-D) com custo total 5



	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	(3,C)	(1,A)	(6,C)	(6,C)
3ª. Iter. (**B**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)
4ª. Iter. (**D**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)
5ª. Iter. (**E**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)

Resultado da tabela de repasse em ***A***:



Origem: A

Destino:	*Link (via):*	Custo total
B	(A,C)	5
C	(A,C)	1
D	(A,C)	5
E	(A,C)	6

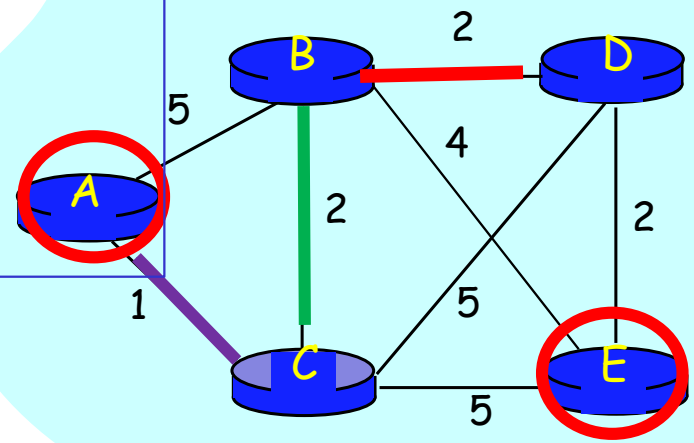
Caminhos de menor custo resultantes, e a tabela de repasse para o nó de origem ***A***.

Deve-se obter a tabela para todas demais origens B, C, D e E

EXERCÍCIO

Exercício: Ache o menor caminho entre A e E:

- Menor custo até E: (6,C)
- Mnor custo até C: (1,A)
- Portanto, menor custo A até E:
  Caminho (A-C-E) com custo total 6



	A	B	C	D	E
1ª. Iter. (**A**)	(0,A)	(5,A)	(1,A)	∞	∞
2ª. Iter. (**C**)	(0,A)	(3,C)	(1,A)	(6,C)	(6,C)
3ª. Iter. (**B**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)
4ª. Iter. (**D**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)
5ª. Iter. (**E**)	(0,A)	(3,C)	(1,A)	(5,B)	(6,C)

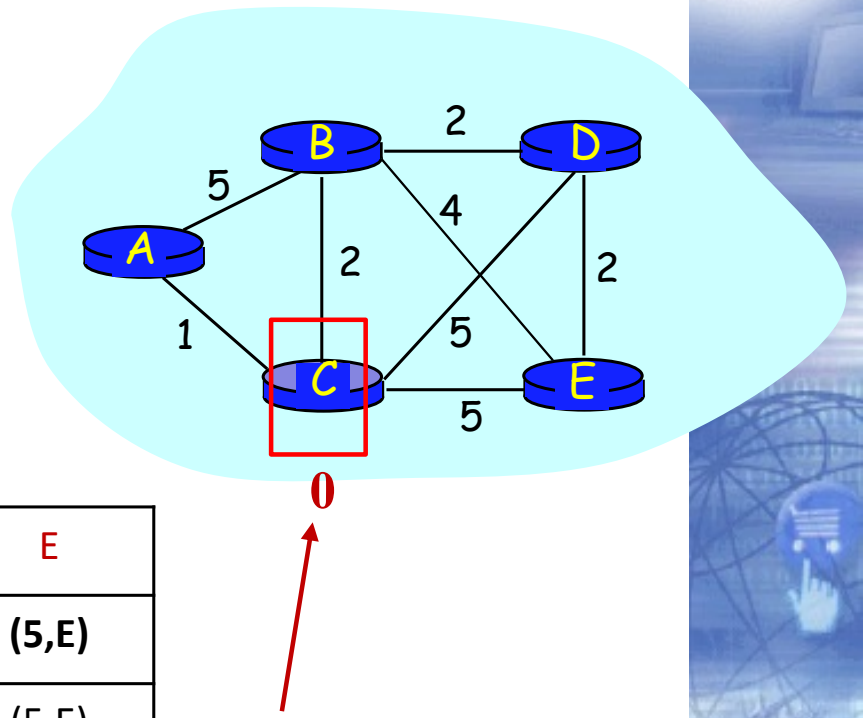
Veja esse vídeo: https://youtu.be/IIZOWRwKa_Q

EXERCÍCIO

Faça para a origem em *C*

Origem: C

Destino:	*Link (via):*	Custo total
A	(C,A)	1
B	(C,B)	2
D	(C,B)	4
E	(C,E)	5



	A	B	C	D	E
1ª. Iter. (C)	**(1,C)**	**(2,C)**	**(0,C)**	**(5,C)**	**(5,E)**
2ª. Iter. (A)	**(1,C)**	**(2,C)**	**(0,C)**	~~(5,C)~~	(5,E)
3ª. Iter. (B)	**(1,C)**	**(2,C)**	**(0,C)**	**(4,B)**	(5,E)
4ª. Iter. (D)	**(1,C)**	**(2,C)**	**(0,C)**	**(4,B)**	(5,E)
5ª. Iter. (E)	**(1,C)**	**(2,C)**	**(0,C)**	**(4,B)**	(5,E)

(Verifique se não errei em nenhuma iteração ☺)

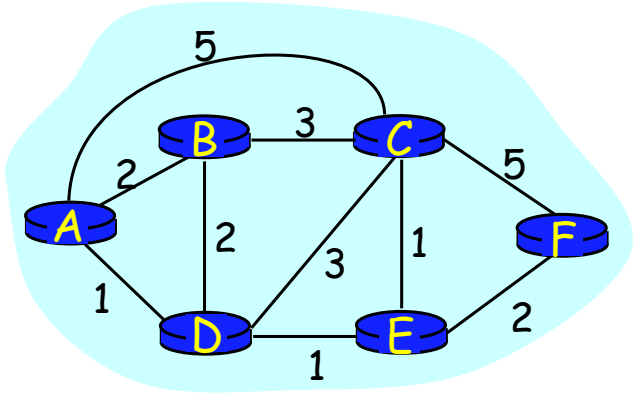
Exercício: obter as tabelas para todos as origens

EXERCÍCIO

Exercício:
Fazer o Exemplo do livro: Rodando o Algoritmo de Dijkstra's iniciando no nó ***A:***

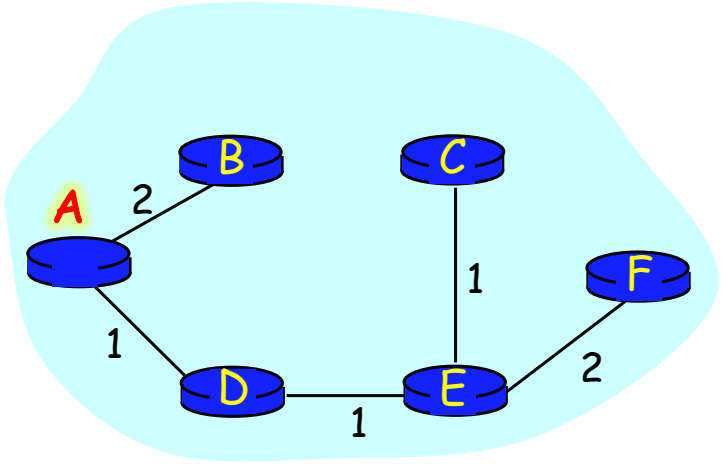
Passo	início N'	D(**B**),p(B)	D(**C**),p(C)	D(**D**),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinito	infinito
1	AD	2,A	4,D		2,D	infinito
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					

- Calculando os **caminhos de menor custo** de ***A*** até **todos os destinos possíveis**.
- Lembrando: ***p(x)*** → predecessor de *x* ao longo do caminho de menor custo atual.
- Cada linha da tabela fornece os valores das variáveis do algoritmo ao final da iteração.



EXERCÍCIO

Resultado da tabela de repasse em ***A***:



A

Destino:	***Link:***
B	(A,B)
C	(A,D)
D	(A,D)
E	(A,D)
F	(A,D)

Caminhos de menor custo resultantes, e a tabela de repasse para o nó ***A***.

Exercício: obter as tabelas para todos os nós

Distance Vector

Equação Bellman - Ford

Algoritmo *Distance Vector*

É **iterativo**, **assíncrono** e **distribuído**.

- Iterativo:
  - Continua até que os nós não troquem mais informações.
  - *Self-terminating* → **não** há sinal de parada.
- Assíncrono:
  - Nós **não** precisam trocar informações **simultaneamente**!
- Distribuído:
  - **Cada nó se comunica** apenas com os seus vizinhos, diretamente conectados.
- Na prática, quem usa *Distance Vector* na Internet:
  - Algoritmo RIP (visto mais adiante).

Equação Bellman – Ford (1)

- Equação de **Bellman-Ford (B-F)** define:

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

Então:

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

Onde min é calculado sobre todos os vizinhos v de x.

Interpretação: o menor custo de uma origem ***X*** até um destino final ***Y***, é o custo de menor valor de um determinado vizinho ***V*** até o destino ***Y***, mais o custo para ***X*** chegar até esse vizinho ***V*** de menor custo (vejamos a seguir).

Equação Bellman – Ford (2)

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

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

Onde min é calculado sobre todos os vizinhos ***v*** de x.

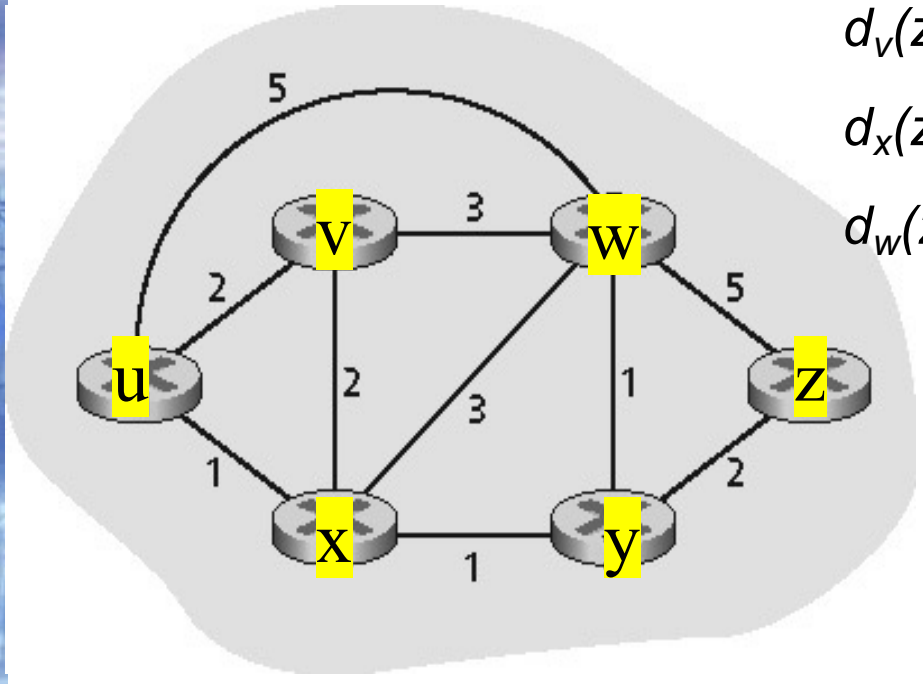
Interpretação:

O menor caminho de ***X*** até um destino ***Y*** é:

O menor custo de ***X*** até um vizinho ***V***, mais o custo desse deste tal vizinho ***V*** até ***Y***,

considerando todos os "***V***" vizinhos de ***X***.

Calculando com Bellman Ford (1)



Vemos que:

$d_v(z) = 5$

$d_x(z) = 3$

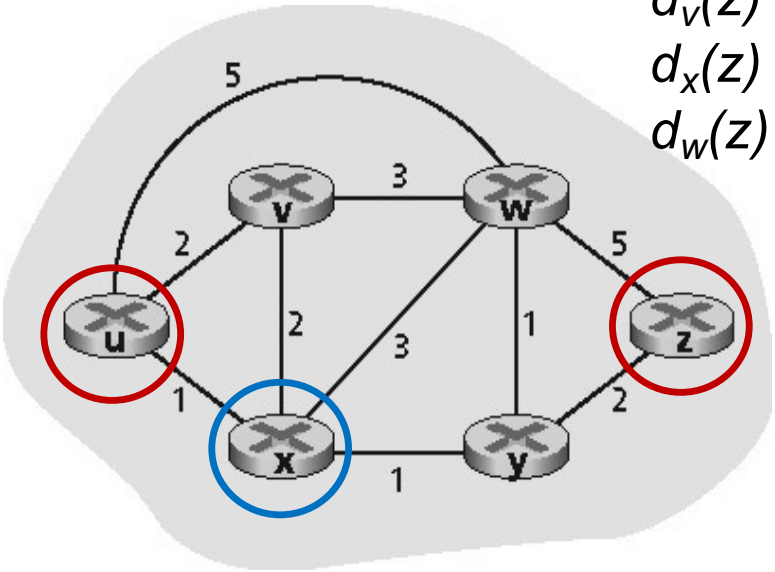
$d_w(z) = 3$

Calculando o menor caminho de ***u*** até ***z*** com a equação B-F diz que:

$$\mathbf{d_u(z)} = \min \{ c(u,v) + d_v(z),\ c(u,x) + d_x(z),\ c(u,w) + d_w(z) \}$$

$$= \min \{ 2 + 5,\ 1 + 3,\ 5 + 3 \} = 4$$

Calculando com Bellman Ford (2)



Vemos que:

$d_v(z) = 5$

$d_x(z) = 3$

$d_w(z) = 3$

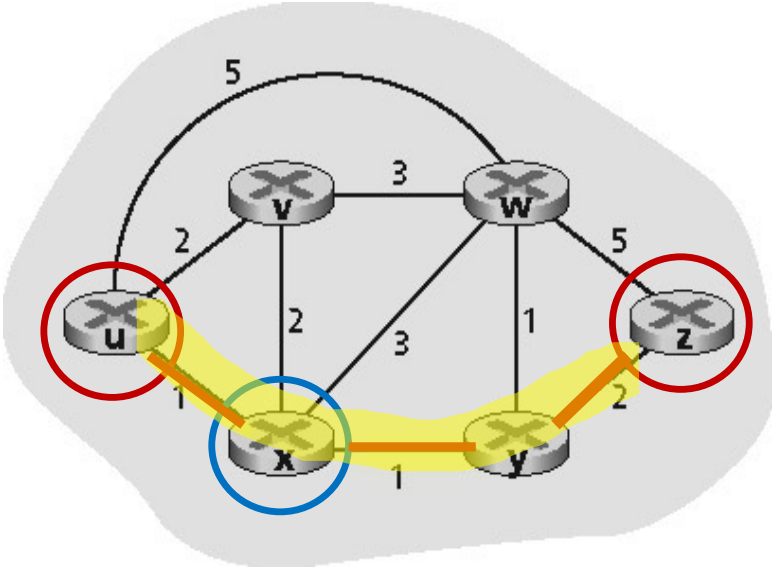
Calculando o menor caminho de ***u*** até ***z*** com a equação B-F diz que:

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

O Menor custo de ***u*** até ***z*** é **4,** e a rota é através de ***x***

O nó que atinge o mínimo é o **próximo salto** no caminho mais curto da origem até o destino ➜ resulta na tabela de roteamento.

Calculando com Bellman Ford (3)



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

$$= \min \{ 2 + 5,\ 1 + 3,\ 5 + 3 \} = 4$$

O Menor custo de ***u*** até ***z*** é **4,** e a rota é através de ***x***

O nó que atinge o mínimo é o **próximo salto** no caminho mais curto da origem até o destino ➜ resulta na tabela de roteamento de u.

O que cada nó mantém:

- Cada nó *x* mantém os seguintes dados de roteamento:

1. O custo $c(x,v)$ **até cada vizinho *v***, diretamente conectado.
2. O **vetor de distâncias dele (nó x)**, contendo a estimativa dos custos de x até todos os destinos y em N.
3. Os **vetores de distâncias de seus *N* vizinhos,** para cada vizinho v de x:

$$D_v = [\ D_v(y):\ y \text{ em } N\ ].$$

Veremos a seguir:

O que é o Vetor de Distâncias

- Vetor de distâncias do nó x é um **conjunto**.
- É um conjunto contendo todas as estimativas de custos de x até todos os outros nós y pertencentes ao conjunto *N* de vizinhos diretamente conectados (vizinhos adjacentes).

$$D_x = \{ D_x(y) \} : y \text{ em } N]$$

Vai ficar mais claro em seguida, ao entender como a informação se propaga:

Conjunto N dos vizinhos:

- No algoritmo de vetor de distância, o conjunto ***N*** representa um conjunto de nós adjacentes ao roteador em questão.
  - Em outras palavras, são os roteadores com os quais o roteador atual tem uma **conexão direta**.
- O conjunto *N* é crucial para o funcionamento do algoritmo, pois ele define os destinos para os quais o roteador envia atualizações de sua tabela de roteamento.
- Essas atualizações contêm informações sobre a distância (ou custo) para cada destino, permitindo que os roteadores vizinhos atualizem suas próprias tabelas de roteamento e construam uma visão global da rede, mesmo com uma informação descentralizada.

Distance Vector – propagação da informação

Cada nó envia periodicamente o seu de **vetor de distância** aos seus vizinhos.

Ou seja, **avisa os vizinhos** quais os custos ele tem até os DESTINOS para os quais ele conhece.

- Quando o nó ***x*** recebe **nova estimativa** do vizinho, ele atualiza sua própria tabela usando a equação B-F:

$$D_x(y) = \min_v\{c(x,v) + D_v(y)\}$$

para cada nó $y \in N$.

- Em condições normais, a estimativa $D_x(y)$ **converge** para o menor custo atual.

Exemplo de cálculo de *Distance Vector*

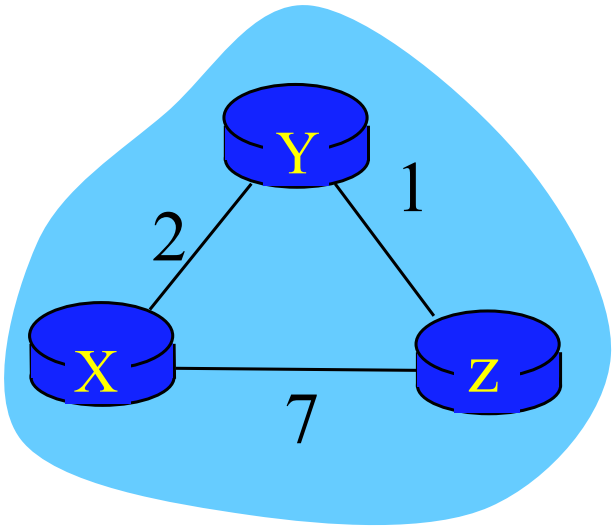


Tabela nó x

A partir de:	Custo para: x	y	z
x	**0**	**2**	**7**
y	∞	∞	∞
z	∞	∞	∞

Tabela nó y

A partir de:	Custo para: x	y	z
x	∞	∞	∞
y	**2**	**0**	**1**
z	∞	∞	∞

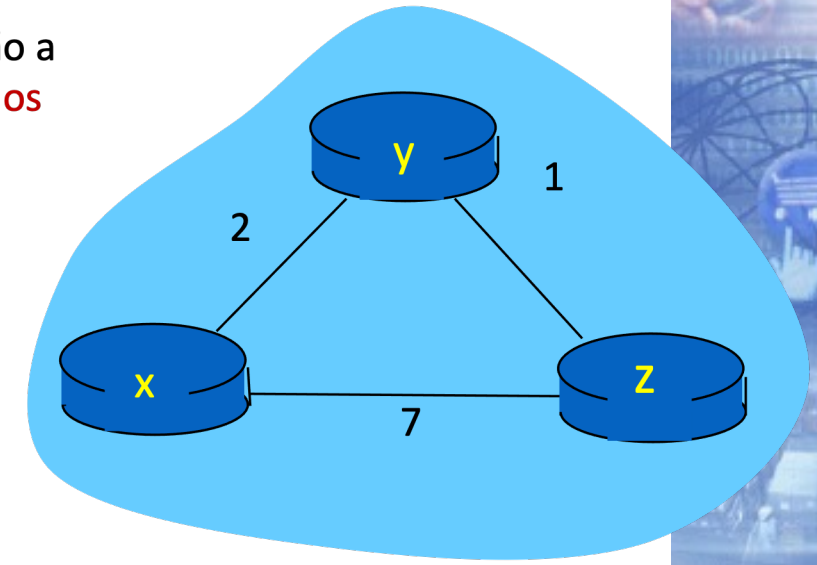
Tabela nó z

A partir de:	Custo para: x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	**7**	**1**	**0**

Quando o algoritmo é iniciado, cada nó **só possui seu custo até seu vizinho diretamente conectado**.

Em seguida, os nós começarão a enviar seus menores custos aos seus vizinhos diretamente conectados.

Ou seja, cada nó informa ao vizinho **quem ele consegue alcançar com menor custo**.



time

Tabela nó x

Custo para:

A partir de:	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

X envia para Y e Z

Tabela nó y

Custo para:

A partir de:	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

Y envia para X e Z

Tabela nó z

Custo para:

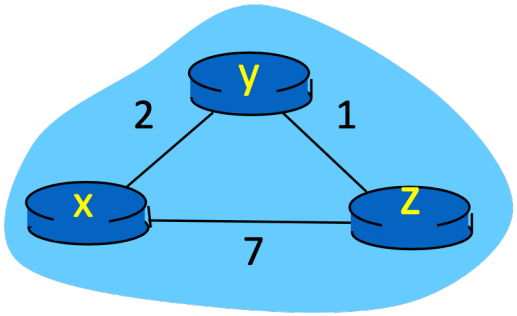
A partir de:	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

Z envia para Y e X

tempo

Os nós começarão a notificar seus menores custos aos seus vizinhos diretamente conectados.

Ou seja, cada nó notifica ao vizinho **quem ele consegue alcançar com menor custo**.



Agora cada nó **verifica se seu próprio caminho de menor custo a cada destino foi alterado**, em função da notificação do vizinho

Tabela nó x

Custo para:

A partir de:	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

Tabela nó y

Custo para:

A partir de:	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

Tabela nó z

Custo para:

A partir de:	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

X envia para Y e Z

Y envia para X e Z

Z envia para Y e X

Y avisa seus custos para seu vizinho ***X***:

C(x,y)= 2
C(y,y)= 0
C(z,y)= 1

Z avisa seus custos para seu vizinho ***X***:

C(x,z)= 7
C(y,z)= 1
C(z,z)= 0

X avisa seus custos para seu vizinho ***Y***:

C(x,x)= 0
C(y,x)= 2
C(z,x)= 7

Z avisa seus custos para seu vizinho ***Y***:

C(x,z)= 7
C(y,z)= 1
C(z,z)= 0

Y avisa seus custos para seu vizinho ***Z***:

C(x,y)= 2
C(y,y)= 0
C(z,y)= 1

X avisa seus custos para seu vizinho ***Z***:

C(x,x)= 0
C(y,x)= 2
C(z,x)= 7

Atualizando apenas para X:

Tabela nó x

Custo para:

A partir de:	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

X envia para Y e Z

Tabela nó y

Custo para:

A partir de:	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

Y envia para X e Z

Tabela nó z

Custo para:

A partir de:	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

Z envia para Y e X

time

Y avisa seus custos para seu vizinho ***X***:

$C(x,y) = 2$ (custo para x a partir de Y)

$C(y,y) = 0$

$C(z,y) = 1$ (custo para z a partir de Y)

Z avisa seus custos para seu vizinho ***X***:

$C(x,z) = 7$ (custo para x a partir de z)

$C(y,z) = 1$ (custo para a partir de z)

$C(z,z) = 0$

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$$

$$= \min\{2+0\ ,\ 7+1\} = 2$$

Não houve alteração

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$$

$$= \min\{2+1\ ,\ 7+0\} = 3$$

Alterou de 7 para 3:

Altera a tabela e avisa os vizinhos.

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$$
$$= \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$$
$$= \min\{2+1\ ,\ 7+0\} = 3$$

Tabela nó x

Custo para:

A partir de:	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

Custo para:

A partir de:	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

Tabela nó y

Custo para:

A partir de:	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

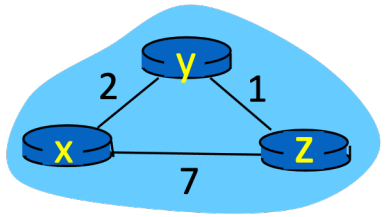
Tabela nó z

Custo para:

A partir de:	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

- Y avisa X que consegue chegar a Z com custo 1.
- Então **X** altera seu menor custo até **Z** de **7** para **3**.
- Agora **X** sabe que:
  - O custo para **X** a partir de **Y** é 2.
  - O custo para **Z** a partir de **Y** é 1.
  - O custo para X a partir de Z é 7.
  - O custo para Y a partir de Z é 1.

- Vizinhos de X terão de ser informados da alteração de custos.
- O mesmo acontece com os demais nós que sofrerem alteração.



$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$$
$$= \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$$
$$= \min\{2+1\ ,\ 7+0\} = 3$$

Tabela nó x

Custo para:

A partir de:	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

Custo para:

A partir de:	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

Custo para *Y* não mudou.

Custo para *Z* mudou:
Avisa os vizinhos!

Tabela nó y

Custo para:

A partir de:	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

Custo para:

A partir de:	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

Tabela nó z

Custo para:

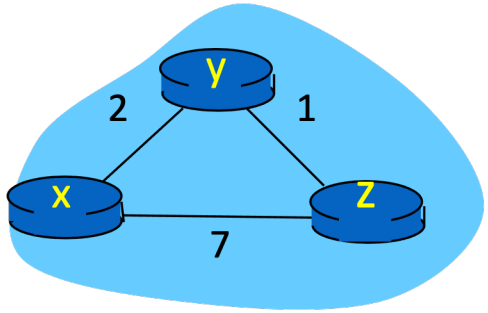
A partir de:	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

Custo para:

A partir de:	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

Custo para *Y* não mudou.

Custo para *X* mudou:
Avisa os vizinhos!



time

$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$

Custo para:

A partir de:	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

Custo para:

A partir de:	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

Custo para:

A partir de:	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

Custo para:

A partir de:	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

Custo para:

A partir de:	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

Custo para:

A partir de:	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

Custo para:

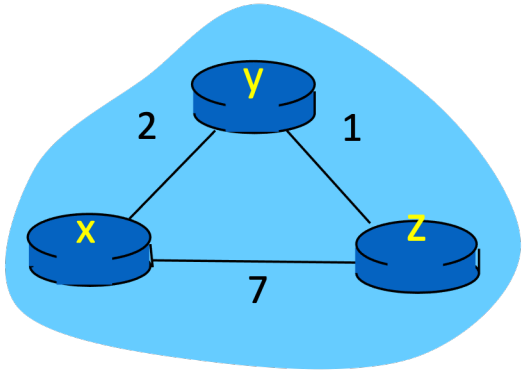
A partir de:	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Custo para:

A partir de:	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Custo para:

A partir de:	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

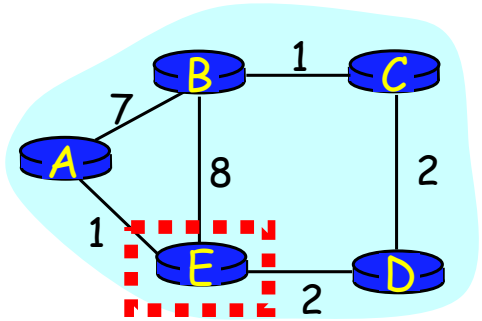


Aplicar a equação B-F para todas as alterações e recalcular os menores custos.

EXERCÍCIO

Calcular a tabela de distâncias do nó *E usando Distance Vector (1)*

$$D^{X}(Y,Z) = c(X,Z) + \min_{w}\{ D^{Z}(Y,w) \}$$



$$D^{E}(C,D) = c(E,D) + \min_{w}\{D^{D}(C,w)\}$$
$$= 2+2 = 4$$

$$D^{E}(A,D) = c(E,D) + \min_{w}\{D^{D}(A,w)\}$$
$$= 2+3 = 5$$

$$D^{E}(A,B) = c(E,B) + \min_{w}\{D^{B}(A,w)\}$$
$$= 8+6 = \boxed{14}$$

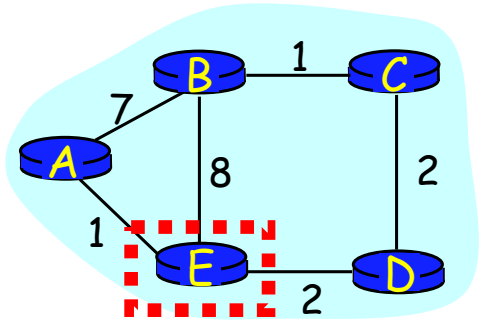
Custo de *E* até *A* dado que o primeiro passo ao longo do caminho é *D*: é o custo de ir de *E* até *D*, mais qualquer que seja o custo mínimo de ir de *D* até *A*.

Note que o caminho menor de *D* até *A* é 3 e esta rota passa novamente por E !

Por que não é <u>15</u> ??

EXERCÍCIO

Calcular a tabela de distâncias do nó *E usando Distance Vector (2)*



$$D^E(C,D) = c(E,D) + \min_w\{D^D(C,w)\}$$
$$= 2+2 = 4$$

$$D^E(A,D) = c(E,D) + \min_w\{D^D(A,w)\}$$
$$= 2+3 = 5$$

$$D^E(A,B) = c(E,B) + \min_w\{D^B(A,w)\}$$
$$= 8+6 = 14$$

custo via nó vizinho

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destino

EXERCÍCIO

A tabela de distâncias gera a tabela de roteamento

custo através de

$D^E()$ (destino)	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destino	Enlace de saída, custo
A	A,1
B	D,5
C	D,4
D	D,2

Tabela de distância → Tabela de Roteamento

EXERCÍCIO

A tabela de distâncias gera a tabela de roteamento

Tabela de roteamento de E

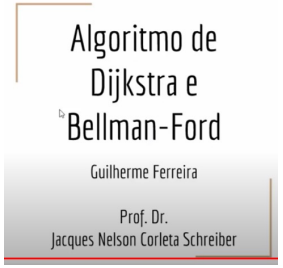
destino	Enlace de saída,custo	
A	A,1	Melhor rota de E para A é através de A, com custo 1
B	D,5	Melhor rota de E para B é através de D, com custo 5
C	D,4	Melhor rota de E para C é através de D, com custo 4
D	D,2	Melhor rota de E para D é através de D, com custo 4

Tabela de distância → Tabela de Roteamento

EXERCÍCIO

Atividade / Referência:

- Algoritmo de Dijkstra e de Bellman-Ford (UNISC)
  Guilherme Ferreira / Prof. Dr. Jacques N.C. Schreiber
  https://youtu.be/aJ_2c9NVCIc - 17 min (19 May 2025)



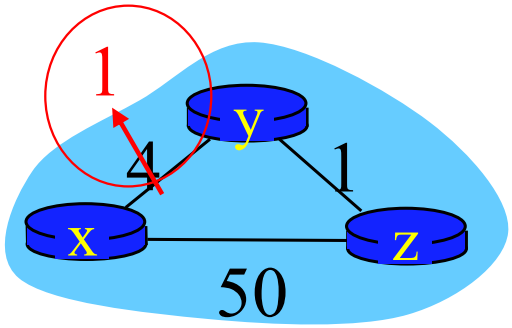
Outras referências:

- *Link State:* *https://www.javatpoint.com/link-state-routing-algorithm*
- *Distance Vector:* https://www.javatpoint.com/distance-vector-routing-algorithm

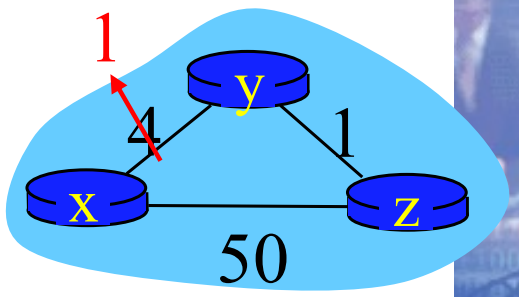
O problema de “Boas notícias viajam depressa” (1)

- Mudanças no custo do enlace:
  - c(x,y) muda de **4** para **1**.
  - Nó ***y*** detecta mudança no custo do enlace local.
  - Atualiza informações de roteamento, e recalcula o vetor de distância.
  - **Menor custo mudou** → avisa vizinhos.

Vejamos o que acontece...



O problema de "Boas notícias viajam depressa" (2)



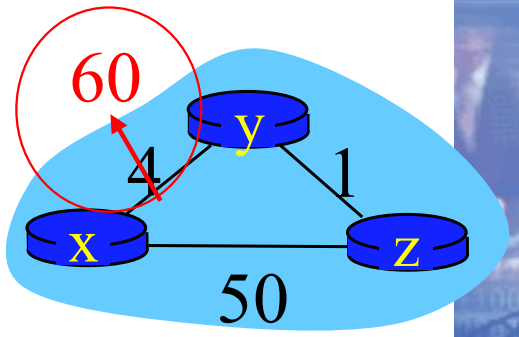
- No tempo t_0 → ***y*** detecta a mudança no custo do enlace: atualiza seu DV, e informa seus vizinhos.
- No tempo t_1 → ***z*** recebe a atualização de ***y*** e atualiza sua própria tabela.
  - Custo c(z,x) mudou de **5** para ***2***.
  - ***Z*** Avisa seu novo vetor aos vizinhos, com seu menor custo.

- No tempo t_2 → ***y*** recebe a atualização de ***z*** dizendo que consegue chegar a X com custo 2 por intermédio dele mesmo (***y***), e atualiza sua tabela de distância.
  - **O menor custo de *y* para chegar a x não muda** (continua sendo 1, diretamente).
  - Então ***y*** **não** envia nenhuma mensagem para ***z***.
- Só 2 iterações são necessárias para o DV alcançar o estado de inatividade.

Boas notícias viajam depressa.

EXERCÍCIO

Más notícias viajam devagar (1)



- Mudanças no custo do enlace:
  - c(x,y) muda de 4 para **60**.
  - Nó ***y*** detecta mudança no custo do enlace local.
  - Atualiza informações de roteamento, e recalcula o vetor de distância:

$$D_y(x) = \min \{ c(y,x) + D_x(x) , c(y,z) + \boldsymbol{D_z(x)} \}$$

$$= \min \{ 60 + 0 , 1 + \mathbf{5} \} = 6$$

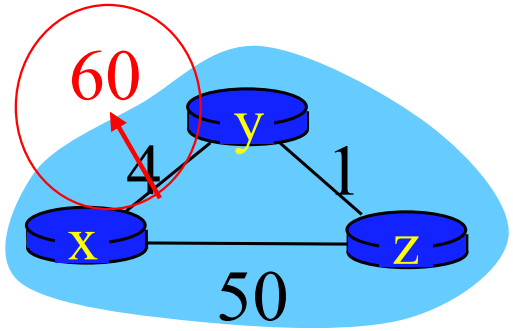
Custo até ***z*** até ***x*** em ***y*** ainda está errado!

Por que o erro?

EXERCÍCIO

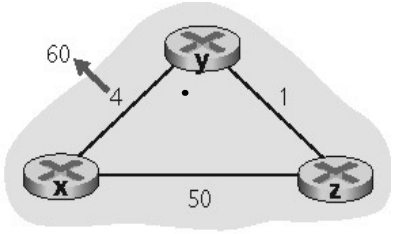
Más notícias viajam devagar (2)

- Por que ocorre o erro?
  - Porque o Algoritmo não é global.
    - Cada nó só tem informação de menor custo dos vizinhos.
- As únicas informações que ***y*** possui:
  - Que seu custo direto até ***x*** agora é 60.
  - E ***z*** disse a ***y*** que pode chegar a *x* com custo 5.
- Em seguida, no instante t_1 temos um *looping* de roteamento.



EXERCÍCIO

Más notícias viajam devagar (3)



b.

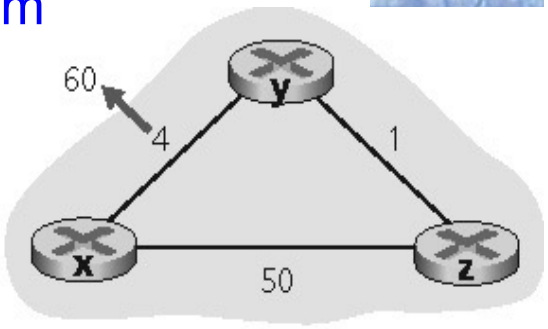
- Assim que ***y*** tenha calculado um novo custo mínimo até ***x***:
  - ***y*** informa ***z*** este novo vetor de distâncias.
- Algum tempo depois de t_1 ocorre que ***z*** recebe o novo vetor de ***y***.
  - Indica que o custo mínimo de ***y*** até ***x*** é 6.
- ***z*** sabe que pode chegar até ***y*** com custo 1.
  - $D_z(x) = \min \{ c(z,x) + D_x(x) , c(z,y) + D_y(x) \}$
  - $= \min \{ 50 + 0 , 1 + 6\} = 7$
- Este processo continua... até estabilizar.

EXERCÍCIO

Problema de Contagem ao Infinito

Mudanças no custo do enlace para pior:

- Neste exemplo: **44 iterações** antes de o algoritmo estabilizar.
- O que aconteceria se *c(y,x)* tivesse mudado para 10.000 e o custo anterior *c(z,x)* fosse 9.999 ?
  - **Más notícias viajam devagar**: problema da "contagem ao infinito".



b.

EXERCÍCIO

Poisoned reverse

- Solução para o problema de contagem ao infinito: reverso envenenado.
- Se a rota de ***Z*** a **X** passa por ***Y*** → *Z* anuncia a *Y* que seu **custo de rota para *X* é infinito**.
  - No exemplo: *Z* anuncia para *Y* que $D_z(X) = \infty$, mesmo que **Z** saiba que, no momento, $D_z(x)=5$.
- *Z* "mente" para *Y*, enquanto sua rota para *X* estiver passando por *Y*.
  - Enquanto *Y* acreditar que *Z* não tem rota até *X*, o nó *Y* nunca vai tentar usar uma rota para *X* através de *Z*.
- **Exercício**: Isso resolve todos os problemas de contagem ao infinito ?

Roteamento hierárquico

Autonomous Systems

Roteamento Hierárquico

Problemas do mundo real:

- Roteadores não são todos idênticos.
- Na prática as redes não são “planas”.

Escala:

- 50 milhões de redes de destinos.
- 500K+ rotas possíveis.
- Não é possível armazenar todos os destinos numa única tabela de rotas.
- Os anúncios com as mudanças na tabela de rotas poderiam congestionar a própria Internet.

Autonomia Administrativa

- Internet = rede de redes.
- Cada administração de rede pode desejar controlar o roteamento na sua própria rede.

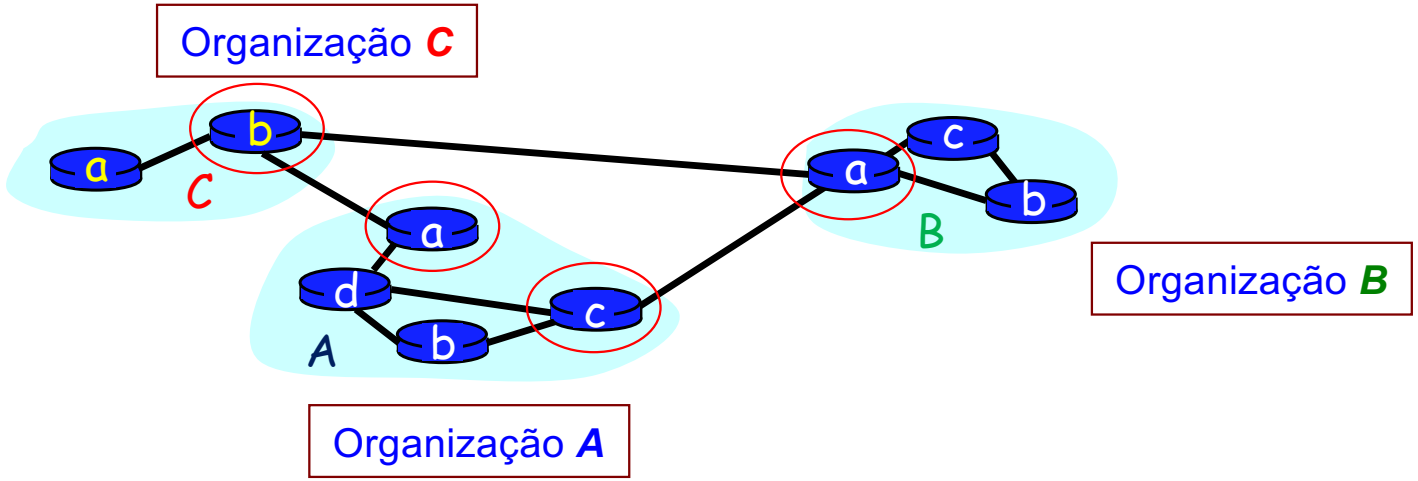
Roteamento Hierárquico

- **Agrega roteadores em regiões**:
  - "Sistemas autônomos" ou "*Autonomous System*" (AS).
- Roteadores **dentro** do **mesmo AS** rodam o **mesmo protocolo de roteamento**:
  - Protocolo de roteamento ***Intra-AS***.
    - Exemplo: UNESP roda OSPF.
  - **Roteadores em AS diferentes podem rodar protocolos de roteamento diferentes.**

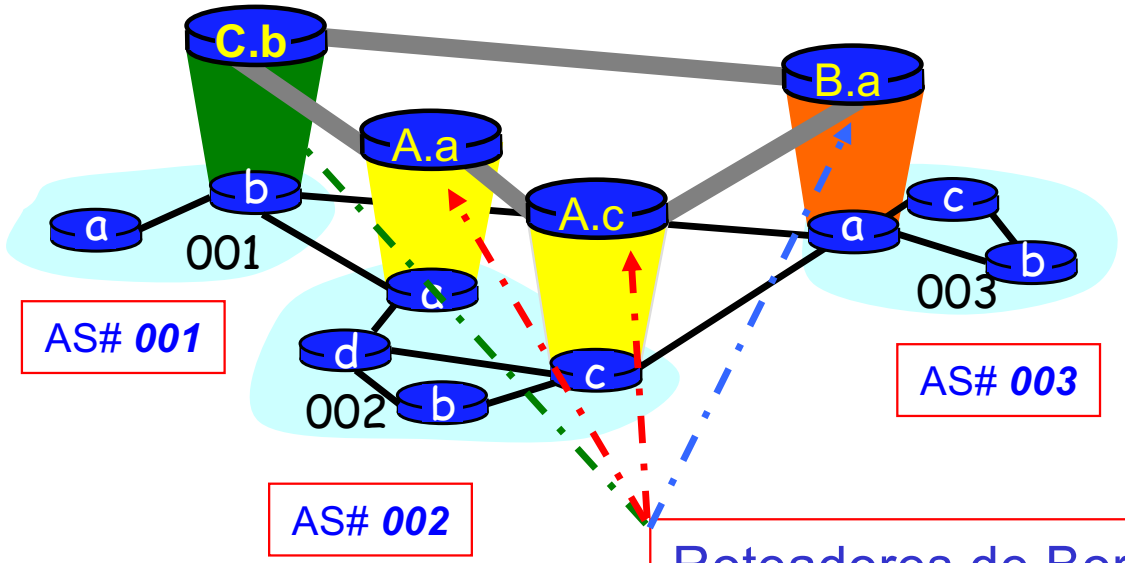
Roteadores de borda

- Estão na borda de um **AS**.
- Rodam protocolos de roteamento **Intra-AS** com os outros roteadores do AS.
- Também responsáveis por enviar mensagens **para fora do AS .**
  - Rodam *protocolo de roteamento* ***inter-AS*** com outros roteadores.

Roteamento Intra-AS e Inter-AS



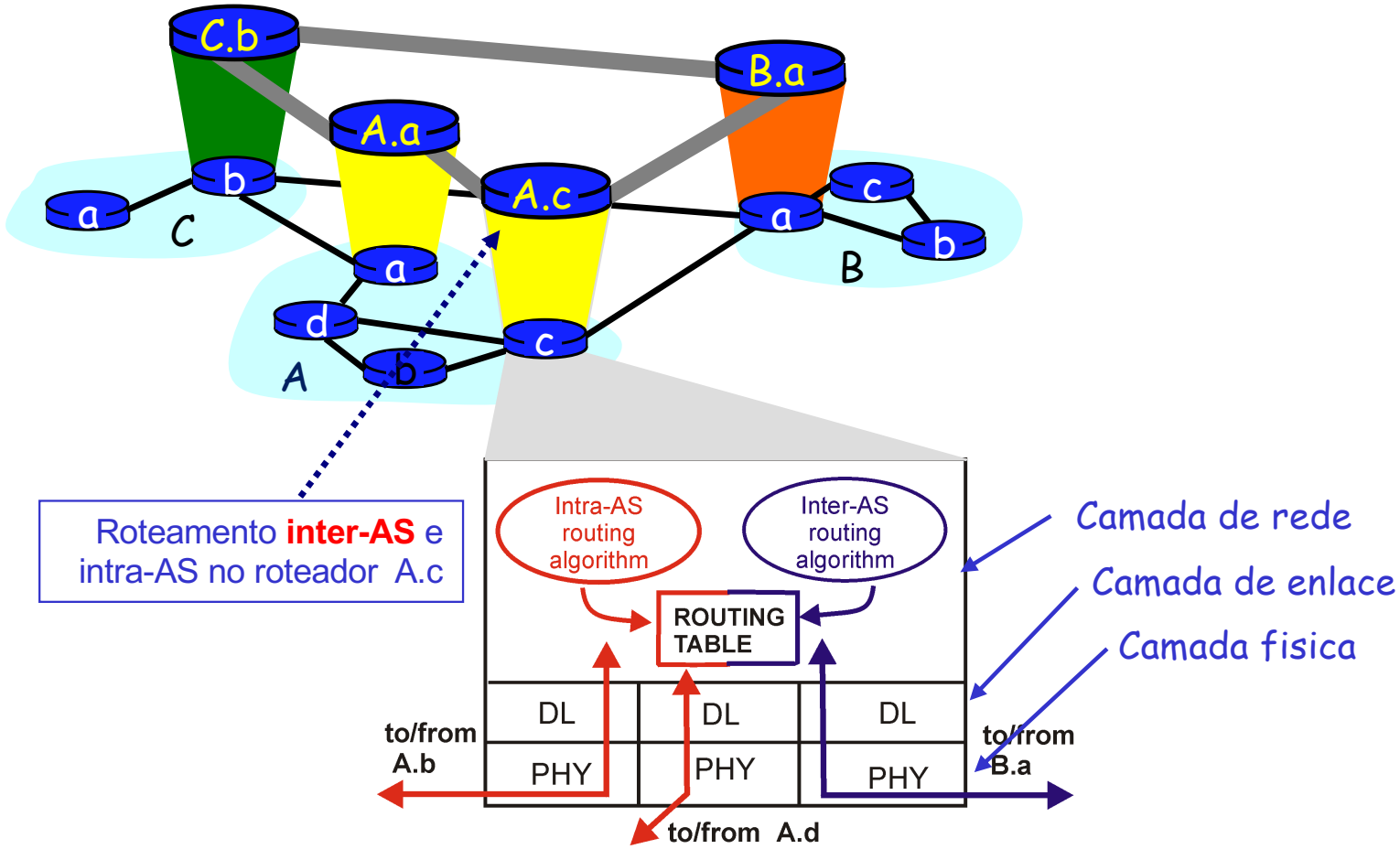
Roteamento Intra-AS e Inter-AS



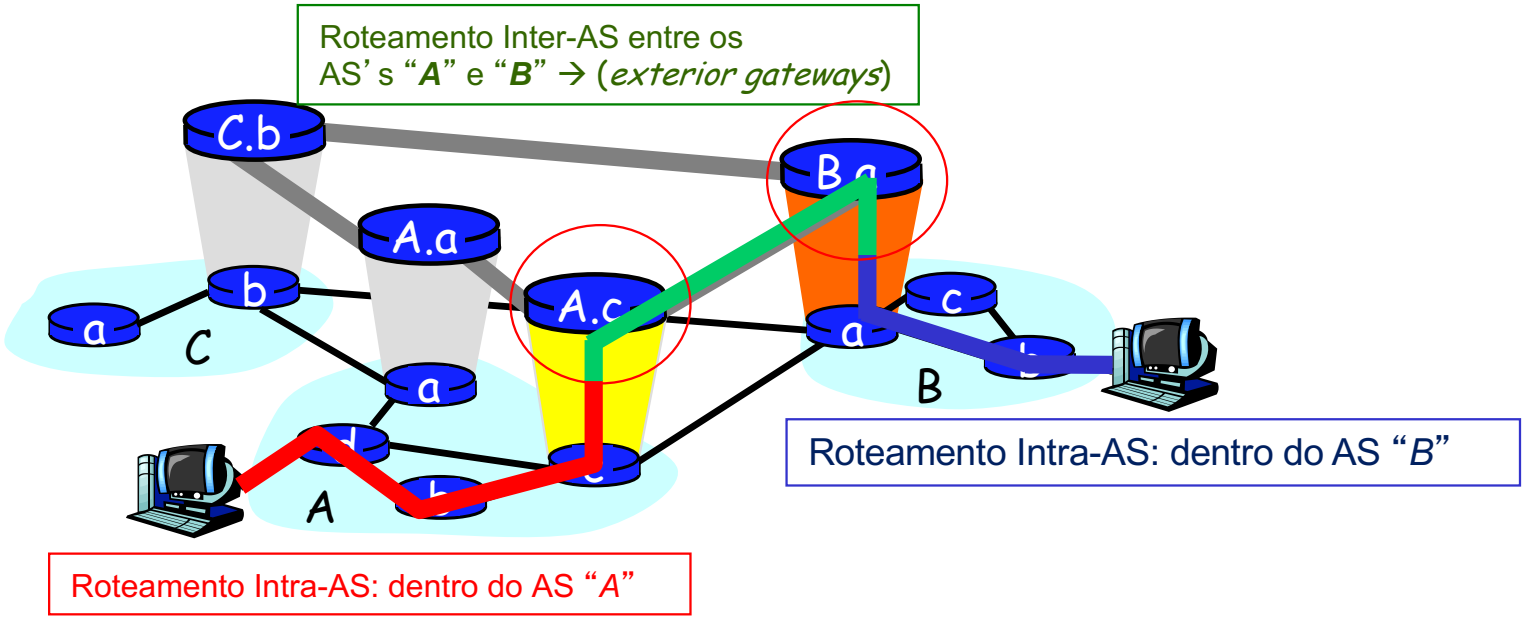
Roteadores de Borda:

- Realizam roteamento **inter-AS** entre instituições diferentes.
- Realizam roteamento **intra-AS** com outros roteadores **dentro do mesmo AS**.

Roteamento Intra-AS e Inter-AS

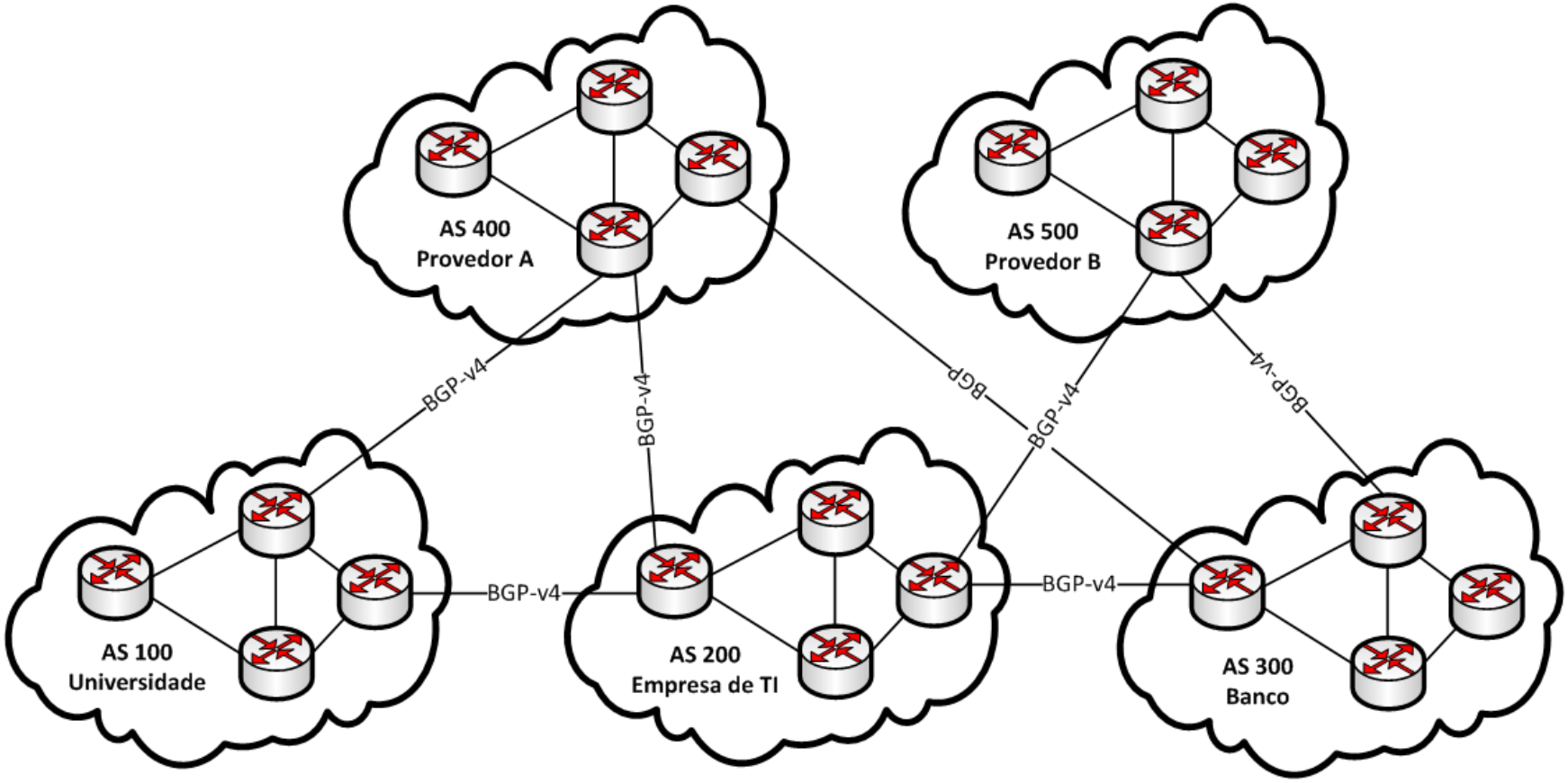


Roteamento Intra-AS e Inter-AS



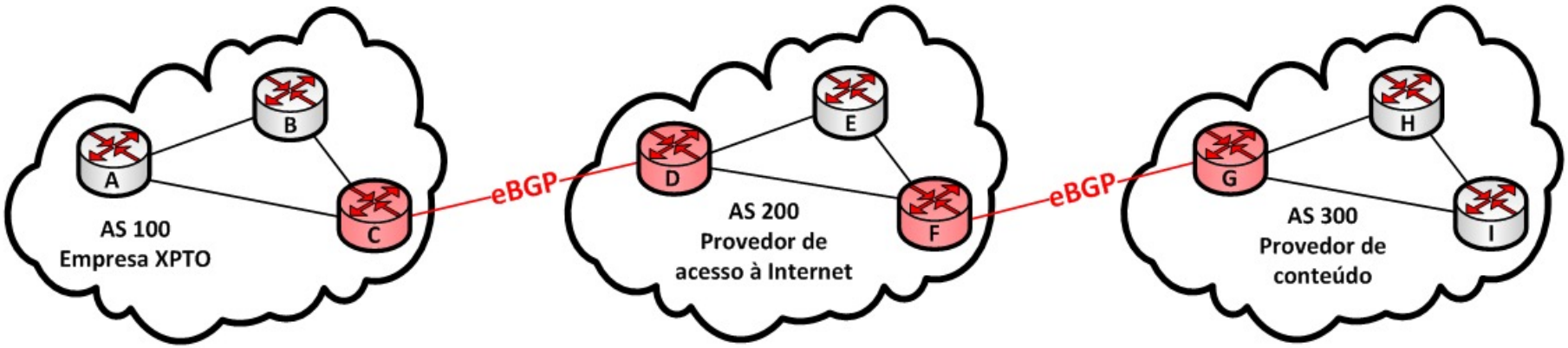
(Nós voltaremos a discutir protocolos de roteamento Inter-AS e Intra-AS e os Sistemas Autônomos mais adiante)

Relacionamento entre os sistemas autônomos



Créditos Figura: https://www.devmedia.com.br/bgp-border-gateway-protocol/29123 (19 May 2025)

Roteamento inter sistemas autônomos



Créditos Figura: https://www.devmedia.com.br/bgp-border-gateway-protocol/29123 (19 May 2025)

Roteamento Intra-AS

Interior Gateway Protocols (*IGP*)

(Cuidado com as siglas! Não confunda IGP / IGRP / EIGRP)

Roteamento Intra-AS

- Também conhecidos como:
  - *Interior Gateway Protocols* (*IGP*).
    - Ou protocolos de roteamento **interno → DENTRO do AS**.
- Os *IGP*s mais comuns são:
  - RIP: *Routing Information Protocol*
  - OSPF: *Open Shortest Path First*
  - IGRP: *Interior Gateway Routing Protocol* (proprietário Cisco)

RIP

Routing Information Protocol

RIP (*Routing Information Protocol*) (1)

- Algoritmo Distribuído ***distance vector*** (equação B-F).
  - Algoritmo de Informação **descentralizada**
- Incluído na distribuição de BSD-UNIX desde1982.
- Distância é **medida apenas em saltos** (*hops*).
  - Enlace **não** tem valor (aresta não tem valor).
  - Não é possível determinar custo para cada enlace.
  - Suporte Máximo = 15 enlaces.
    - Não havia mais do que 15 saltos na internet nos anos 80.

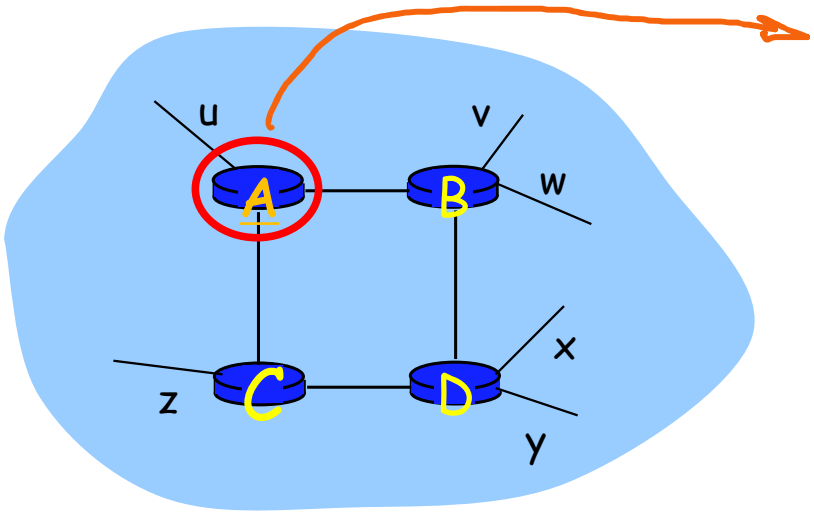
(exemplo a seguir)

distance vector

RIP (*Routing Information Protocol*) (2)

- Métrica de distância: número de saltos
  - Máximo = 15 saltos.

u, v, w, x, y, z → redes acessíveis
A, B, C, D → roteadores (saltos)



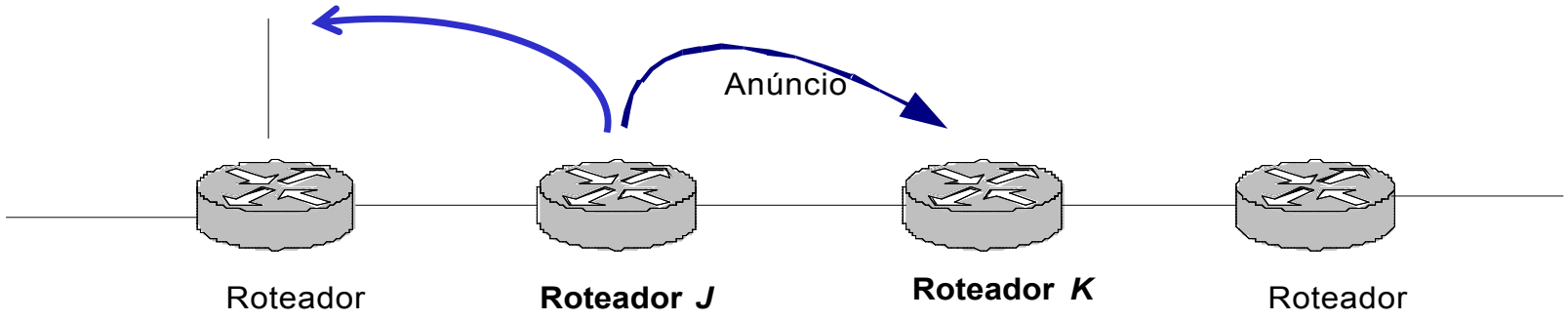
Do roteador ***A*** às sub-redes:

destino	saltos
u	1
v	2
w	2
x	3
y	3
z	2

Anúncios RIP

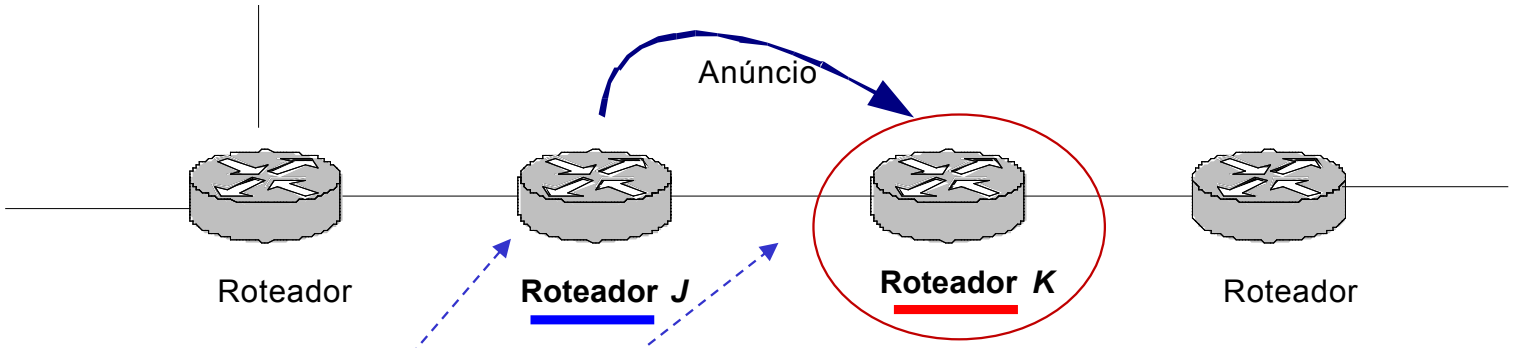
distance vector

- ***Vetores de distância:*** trocados entre vizinhos a cada 30seg.
- Também conhecida como "anúncio" ("*advertize*"):
  - Cada roteador **envia uma cópia de sua tabela de roteamento** para qualquer **outro roteador vizinho.**
  - Cada **anúncio** pode listar até 25 sub-redes de destino dentro do AS.



Operação do RIP (1)

distance vector



- Quando um anúncio chega ao roteador ***K*** vindo do roteador ***J*** :
- ***K*** **examina os destinos que ele conhece, e a distância até cada um destes destinos.** Então:
  - Se ***J*** conhecer caminho **mais curto** para chegar a um destino, ou
  - Se ***J*** listar um destino que **não** existe na tabela de ***K***, ou
  - Se ***K*** estiver no momento roteando para um destino através de ***J*** e a distância de ***J*** até aquele destino mudar,
- ...então ***K*** substitui a sua entrada na tabela.

Operação do RIP (2)

distance vector

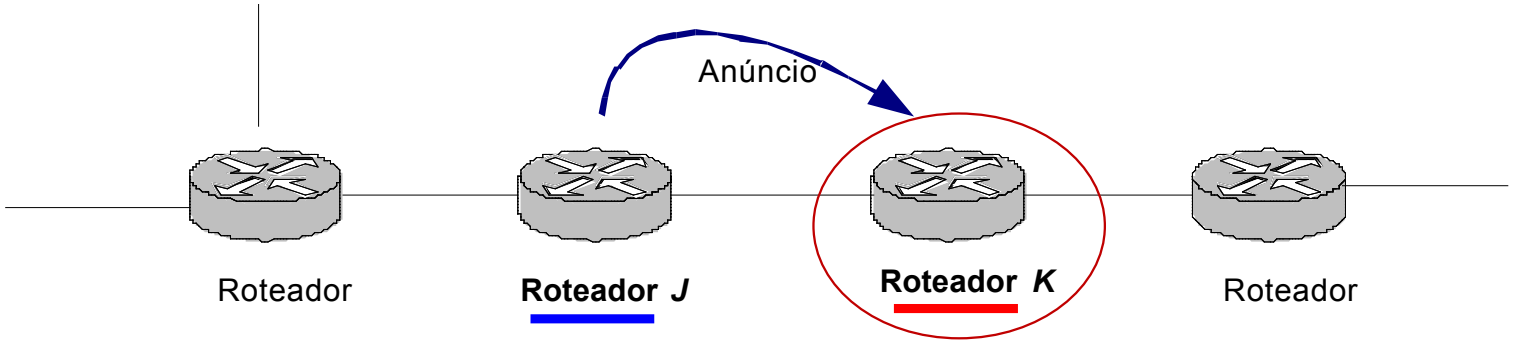


Tabela de K		
Destino	**Distância**	**Rota**
Rede 1	0	Direta
Rede 2	0	Direta
Rede 4	**8**	**Roteador L**
Rede 17	5	Roteador M
Rede 24	6	Roteador J
Rede 30	2	Roteador Q
Rede 42	**2**	**Roteador J**

	Atualização de J	
	Destino	**Distância**
	Rede 1	2
→	**Rede 4**	**3**
	Rede 17	6
→	**Rede 21**	**4**
	Rede 24	5
	Rede 30	10
→	**Rede 42**	**3**

Melhor custo por ***J*** do que por ***L***

Não existia

Custo por ***J*** foi alterado (aumentou)

- À esquerda: uma tabela de roteamento existente num roteador ***K***.
- À direita: uma mensagem de atualização recebida de um vizinho ***J***.

As entradas marcadas com "→" serão usadas para atualizar as entradas existentes (ou acrescentar) na tabela de ***K***.

Operação do RIP (3)

distance vector

Tabela de K		
Destino	**Distância**	**Rota**
Rede 1	0	Direta
Rede 2	0	Direta
Rede 4	**8**	**Roteador L**
Rede 17	5	Roteador M
Rede 24	6	Roteador J
Rede 30	2	Roteador Q
Rede 42	**2**	**Roteador J**

	Atualização de J	
	Destino	**Distância**
	Rede 1	2
→	**Rede 4**	**3**
	Rede 17	6
→	**Rede 21**	**4**
	Rede 24	5
	Rede 30	10
→	**Rede 42**	**3**

Melhor custo por ***J*** do que por ***L***

Não existia

Custo por ***J*** foi alterado (aumentou)

- À esquerda: uma tabela de roteamento existente num roteador ***K***.
- À direita: uma mensagem de atualização recebida de um vizinho ***J***.

As entradas marcadas com "→" serão usadas para atualizar as entradas existentes (ou acrescentar) na tabela de ***K***.

Note que se ***J*** informar uma distância *N*, então uma entrada atualizada em ***K*** terá uma distância ***N+1*** → a distância para acessar o destino a partir de ***J***, mais a distância para acessar ***J*** (lembrando que a única métrica de custo é No. de hops)

Operação do RIP (4)

distance vector

Tabela de K		
Destino	**Distância**	**Rota**
Rede 1	0	Direta
Rede 2	0	Direta
Rede 4	**8**	**Roteador L**
Rede 17	5	Roteador M
Rede 24	6	Roteador J
Rede 30	2	Roteador Q
Rede 42	**2**	**Roteador J**

	Atualização de J	
	Destino	**Distância**
	Rede 1	2
→	**Rede 4**	**3**
	Rede 17	6
→	**Rede 21**	**4**
	Rede 24	5
	Rede 30	10
→	**Rede 42**	**3**

Melhor custo ficou por ***J*** do que por ***L***

Não existia em K

Custo por ***J*** foi alterado (aumentou)

Resultado da Tabela de *K* após a atualização:

Tabela de K		
Destino	**Distância**	**Rota**
Rede 1	0	Direta
Rede 2	0	Direta
Rede 4	**4**	**Roteador J**
Rede 17	5	Roteador M
Rede 21	5	Roteador J
Rede 24	6	Roteador J
Rede 30	2	Roteador Q
Rede 42	**4**	**Roteador J**

Por sua vez, *K* irá avisar seus vizinhos sobre sua atualização.

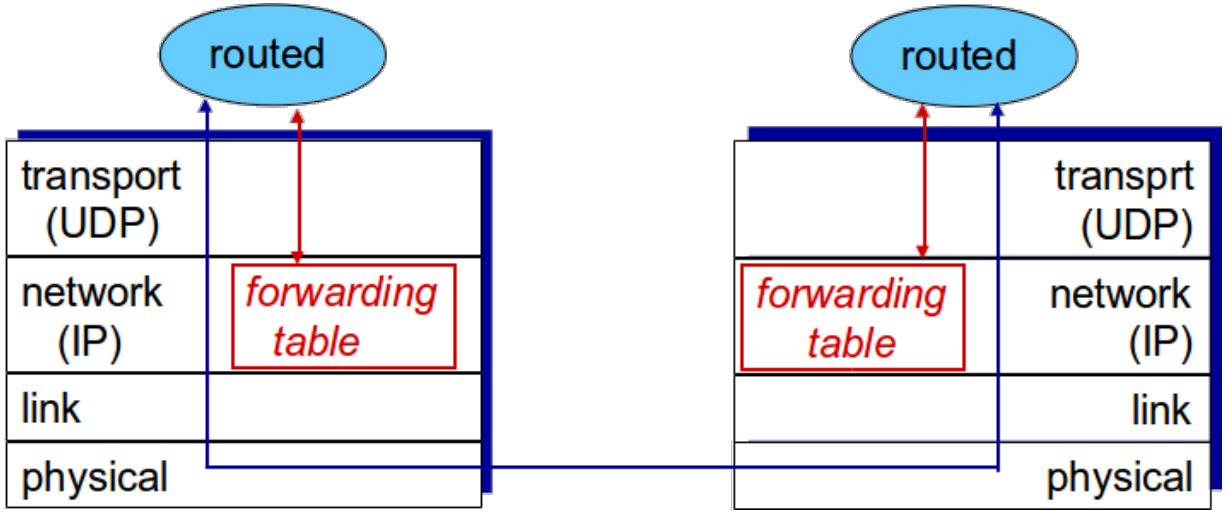
Note que se ***J*** informar uma distância *N*, então uma entrada atualizada em ***K*** terá uma distância ***N+1*** → a distância para acessar o destino a partir de ***J***, mais a distância para acessar ***J***

RIP: Recuperação de falhas

- Lembrando: é *Distance Vector.*
- Se não for recebido anúncio novo durante **180 seg** então o vizinho (e por consequência, o enlace) é declarados morto.
  - Rotas via vizinho são eliminadas.
  - Novos anúncios são enviados aos vizinhos.
  - Por sua vez, os vizinhos publicam novos anúncios, se foram alteradas as suas tabelas.

Processo *Routed*

- Tabelas de roteamento RIP são controladas por processo em **nível de aplicação** chamado ***routed***.
- Anúncios são enviados em pacotes UDP, repetidos periodicamente.



EXERCÍCIO

RIP: Recuperação de falhas - Exercício

- Exercício: já vimos uma técnica para evitar rotas *looping* e contagem ao infinito com algoritmos *Distance Vector*, chamada de *poisoned reverse*. Pesquise e estude as seguintes técnicas de recuperação ou mitigação de rotas cíclicas:
  1. Envenenamento de rotas (*Route Poisoning*).
  2. Estreitamento de horizontes (*Split Horizon*).
  3. Tempo de Espera (*Holddown Timers*).
  4. Atualizações Imediatas (*Triggered Updates*).

OSPF

Open Shortest Path First

OSPF (*Open Shortest Path First*)

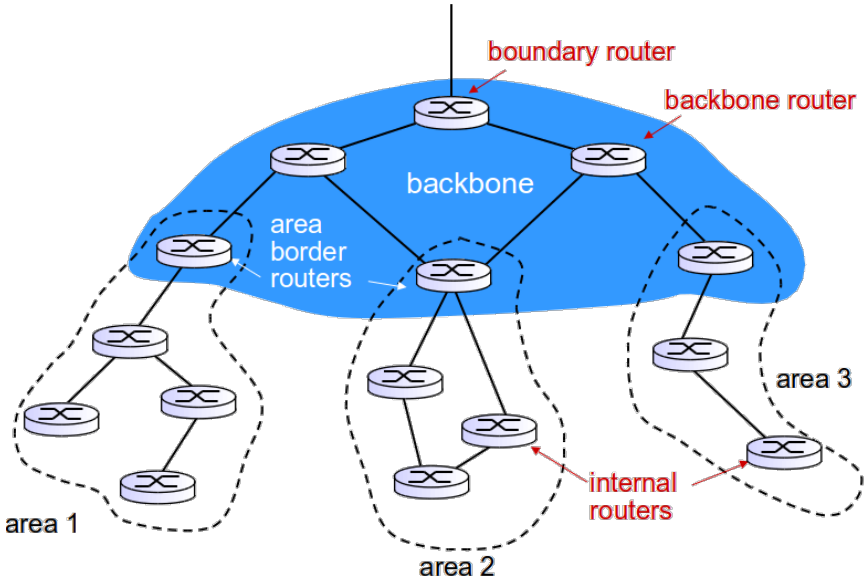
Link State (**Dijkstra**)
Global

- "*Open*" → publicamente disponível.
- Usa algoritmo Global de *Link State* (**Dijkstra**).
  - Informação global.
  - Disseminação de "Pacotes de Estado de Enlace".
  - Mapa da topologia está presente em cada nó.
  - Cálculo de rotas é feito usando o algoritmo de Dijkstra.
- Anúncio de OSPF transporta uma entrada por roteador vizinho.
  - Anúncios são disseminados para o AS inteiro
  - Via inundação – *flooding*.
  - Transportados em mensagens OSPF diretamente sobre IP (em vez de TCP ou UDP).

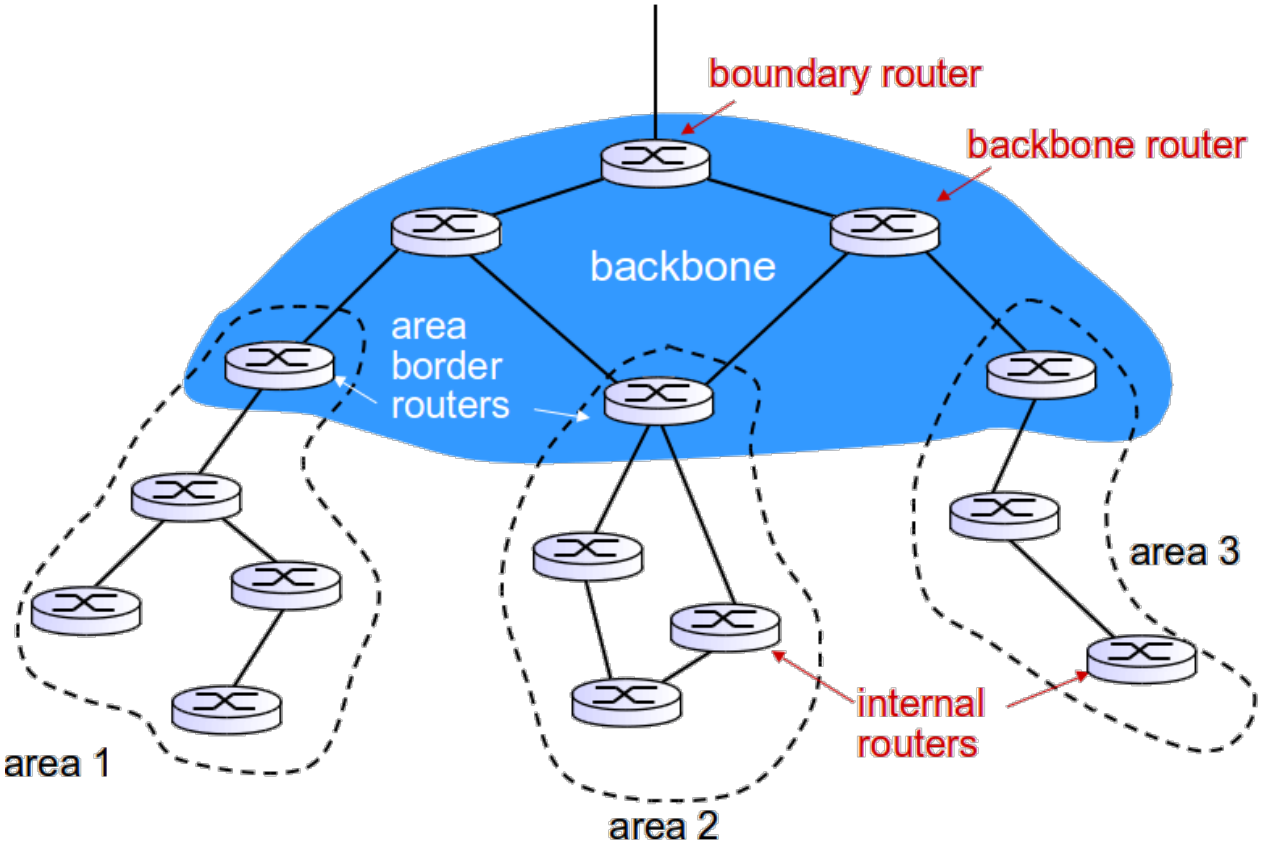
OSPF: características “avançadas” (não presentes em RIP)

Link State (**Dijkstra**)
Global

- Segurança: todas mensagens OSPF são autenticadas.
  - Utiliza conexões TCP para autenticação.
- Aceita caminhos Múltiplos, com custos iguais permitidos.
  - (O RIP aceita **apenas** uma rota, quando custos são iguais).
- OSPF pode ser hierárquico em domínios grandes.



Referência de OSPF Hierárquico (opcional; para quem deseja se aprofundar):



https://www.ciscopress.com/articles/article.asp?p=2294214 (19 May 2025)

Home > Articles > Cisco Certification > CCNP > OSPF Implementation

OSPF Implementation - Sample Chapter is provided courtesy of Cisco Press. - Date: Feb 3, 2015.

IGRP e EIGRP

Interior Gateway Routing Protocol

Enhanced Interior Gateway Routing Protocol

IGRP (Interior Gateway Routing Protocol)

- Proprietário da CISCO.
- Sucessor do RIP (final dos anos 80).
- Usa *Distance Vector* (B-F)→ assim como RIP.
  - Algoritmo de Informação **descentralizada**
- Diversas métricas de custo:
  - Atraso, largura de banda, confiabilidade, carga, etc...
- Usa TCP para trocar mudanças de rotas.

EIGRP (*Enhanced Interior Gateway Routing Protocol*)

- O EIGRP é uma versão aprimorada do IGRP.
  - Protocolo avançado de roteamento por *Distance Vector* (B-F).
    - Desenvolvido com base no protocolo proprietário IGRP da Cisco.
- Funcionalidades básicas do IGRP foram convertidas para padrão aberto em 2013, e foram especificadas na RFC7868 em 2016.

EXERCÍCIO

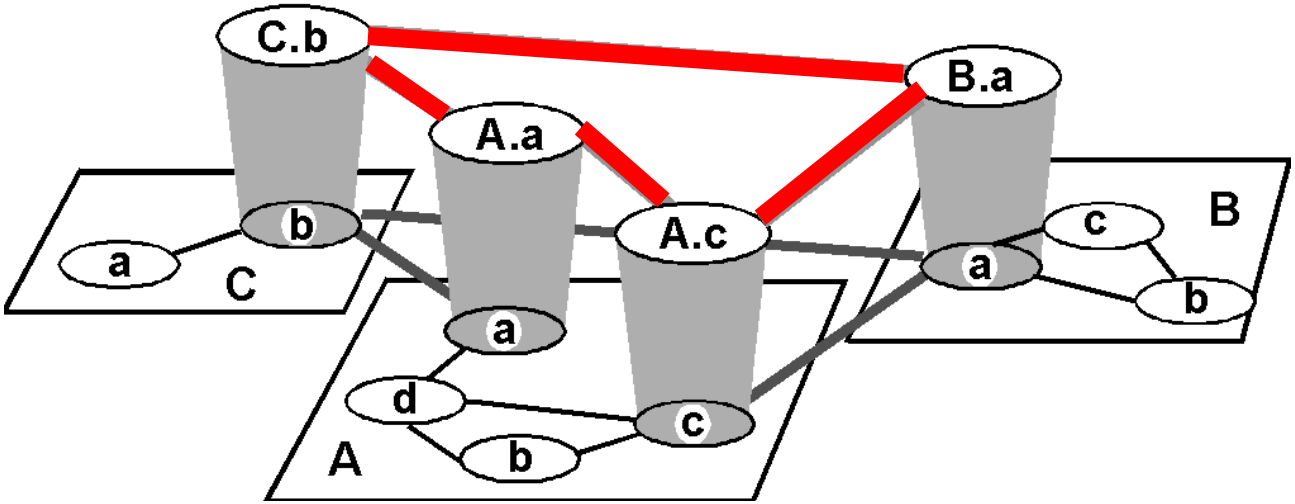
IGPs - Resumo

- Os IGPs suportam os roteadores instalados dentro de um mesmo AS.
- São agrupados em duas principais categorias:
- Vetor de distância (*Distance Vector*):
  - Os roteadores possuem uma tabela que armazena as menores distâncias para os destinos conhecidos e por onde os pacotes deverão ser enviados, a fim de determinar as melhores rotas.
  - *Routing Information Protocol* (RIP), *Interior Gateway Routing Protocol* (IGRP) e *Enhanced Interior Gateway Routing Protocol* (EIGRP).
- Estado de Enlace (*Link State*):
  - Todos os roteadores mapeiam a topologia lógica da rede e se baseiam nesta para determinar os melhores caminhos.
  - Protocolos mais conhecidos neste subconjunto: *Open Shortest Path First (OSPF)* e *Intermediate System to Intermediate System* (IS-IS; não discutido aqui).

Roteamento Inter-AS

Exterior Gateway Protocols (EGP)

Roteamento Inter-AS: o BGP V.4



Sistemas Autônomos

- Já vimos até aqui que um **AS** é composto por uma ou mais redes gerenciadas de acordo com uma política de roteamento única.
  - Define as regras para o encaminhamento dos pacotes IP à Internet.
- Cada AS pode determinar o protocolo de roteamento interno:
  - **IGP** (*Interior Gateway Protocol*) → Intra-AS
  - Realiza a integração dos componentes de sua infraestrutura, com base em seus próprios critérios operacionais, administrativos, e técnicos.
- Por outro lado, para que seja viável a troca de dados entre os diferentes sistemas autônomos, é necessário que todos optem por um único protocolo de roteamento externo:
  - **EGP** (*Exterior Gateway Protocol*) → Inter-AS

BGP: o roteamento inter-AS na Internet:

- O protocolo BGP foi criado em meados de **1989**.
- Propósitos:
  - Permitir diálogo entre os diferentes roteadores.
  - Suportar as diferentes infraestruturas já existentes.
  - Fornecer escalabilidade, flexibilidade e redundância.
  - Possibilitar a criação de políticas de roteamento que respeitassem as particularidades e os anseios de cada organização.

Estas características tornaram o protocolo mais complexo, pois foi necessária a inclusão de diversos recursos.

Roteamento inter-AS na Internet: BGP

- BGP (***Border Gateway Protocol***)
- Padrão "de fato" na Internet para roteamento inter-AS.
  - ***Border gateway protocol version* 4** (BGP-4).
- Como um AS é identificado:
  - Número global e único ***Autonomous System Number*** – "ASN" ou "AS#"
  - É atribuído às organizações por um ***Regional Internet Registry*** – RIR (*).
    - **Mediante atendimento de critérios técnicos bastante restritos.**
- Inicialmente, o ASN foi definido como um número inteiro de 16 bits (**2 bytes**), possibilitando associar números desde **1 a 65.535** (mas foi insuficiente).
- Posteriormente, a faixa foi ampliada para 32 bits (**4 bytes**), expandindo para **4.294.967.295** ASNs possíveis.

(*) Órgão responsável pela alocação e reserva dos recursos de numeração da Internet em uma determinada região do mundo.

EXERCÍCIO

Quem pode ser um AS?

- As instituições públicas e privadas que desejam se tornar um sistema autônomo devem apresentar suas justificativas técnicas ao RIR e descrever sua infraestrutura de comunicação de dados.
- Além disso, precisam possuir:
  - Equipe técnica habilitada para projetar, implantar e operar sua política de roteamento.
  - Software e hardware compatíveis com o protocolo BGP-4.
  - Condições financeiras para a manutenção dos recursos de numeração (ASN e endereços IP).
- Organizações mais comuns que operam AS:
  - Operadoras de telecomunicações, provedores de conteúdo, empresas de tecnologia da informação, fornecedores de hardware e software, órgãos de ensino e pesquisa, instituições financeiras, entre outros, são alguns exemplos de sistemas autônomos.

EXERCÍCIO

Quais as vantagens de ser um AS?

- **Disponibilidade**:
  - Remove a restrição de conexão em um único provedor de acesso à Internet, ou seja, possibilita o encaminhamento dos dados para múltiplas operadoras de telecomunicações (*multi-homed system*), além de contingenciamento.
- **Espaço de endereçamento IP próprio:**
  - Os endereços IP podem ser atribuídos diretamente aos clientes, reduzindo o uso de NAT e facilitando a rastreabilidade dos eventos na rede.
- **Portabilidade de endereços IP:**
  - Independência do provedor de acesso, permitindo a manutenção das configurações IP mesmo quando ocorrerem mudanças nos fornecedores de acesso à Internet.
- **Políticas de roteamento:**
  - A engenharia de tráfego determina a melhor opção técnica-financeira para o encaminhamento de dados IP aos diversos destinos na rede mundial de computadores.

Histórico

- 1970: *Exterior Gateway Protocol* – EGP (*)
  - Primeiro protocolo de roteamento inter-AS.
  - Não se mostrou flexível para a configuração das políticas de roteamento, e para suportar diferentes topologias, evitando *loops* no encaminhamento dos pacotes IP.
- 1989: Protocolo BGP (RFC 1105):
  - Visava substituir o EGP e suplantar suas deficiências.
  - A proposta inicial sofreu diversas alterações até a **Versão 4 atual** (RFC 4271)

(*) Infelizmente, é mesmo nome da categoria de roteamento inter-AS, mas tente não confundir).

Roteamento inter-AS na Internet: BGP

- BGP oferece a cada AS um meio de:

1. Obter informação de acesso a uma sub-rede, a partir de ASs vizinhos.
   - Via "sessão eBGP"
2. Propagar informação de acessibilidade a todos os roteadores **internos** ao AS.
   - Via "sessão iBGP"
3. Determinar rotas "boas" para sub-redes com base na informação e **política** de acessibilidade.
   - Permite que uma sub-rede anuncie sua existência ao resto da Internet.

EXERCÍCIO

BGP - Características principais:

- Suporte às diferentes arquiteturas de comunicação de dados.
  - Permitindo flexibilidade na concepção dos projetos de infraestrutura.
- **Atualização incremental das tabelas** de roteamento.
  - Apenas no momento inicial são enviados todos os dados.
- Uso de um protocolo confiável TCP na troca das mensagens.
- Disponibilização do *Classless Interdomain Routing* (**CIDR**).
  - Recurso que possibilita a agregação ou sumarização de rotas IP.
- Opção de autenticação da conexão entre os roteadores envolvidos (chamados de BGP *speakers*).
  - Por meio de senhas ou certificados digitais.

Cenários BGP:

- Os 3 cenários fundamentais de roteamento suportados pelo protocolo BGP-4 são:
  1. Roteamento inter sistemas autônomos (via sessões ***eBGP***).
  2. Roteamento intra sistemas autônomos (via sessões ***iBGP***).
  3. Roteamento de trânsito entre ASs.

Definem:

- O relacionamento entre os roteadores;
- As regras que serão utilizadas para propagação das tabelas de roteamento; e
- As políticas que serão implantadas por cada um dos sistemas autônomos.

1. Vizinhança "*external BGP*" (eBGP)

- Dois ou mais roteadores em ASs distintos se comunicam pelo roteamento inter-sistemas autônomos.
  - Gera uma relação de vizinhança ***external* BGP** (***eBGP***).
  - É a aplicação mais comum do protocolo BGP-4.

É o alicerce de funcionamento da Internet.

A Internet pode ser definida como uma coleção de ASs interconectados.

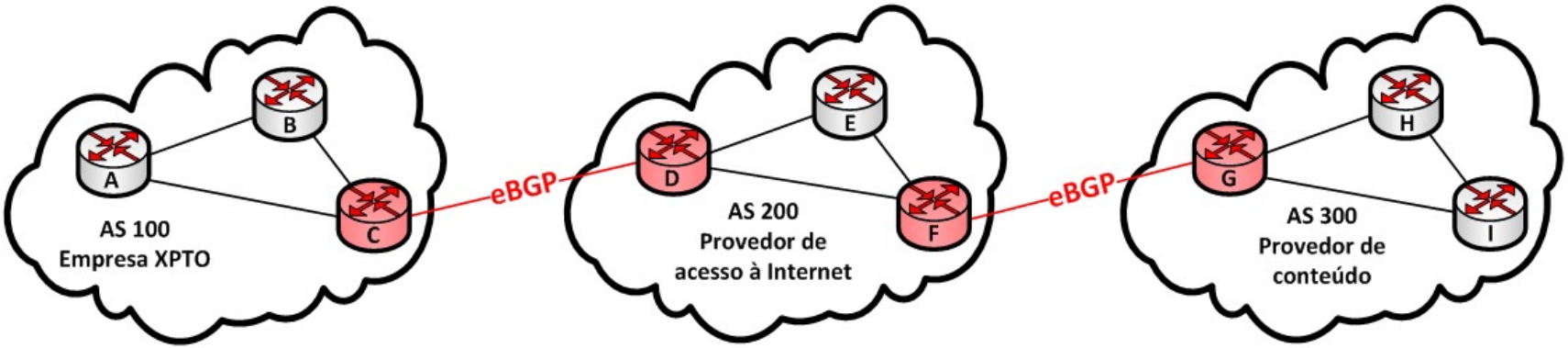
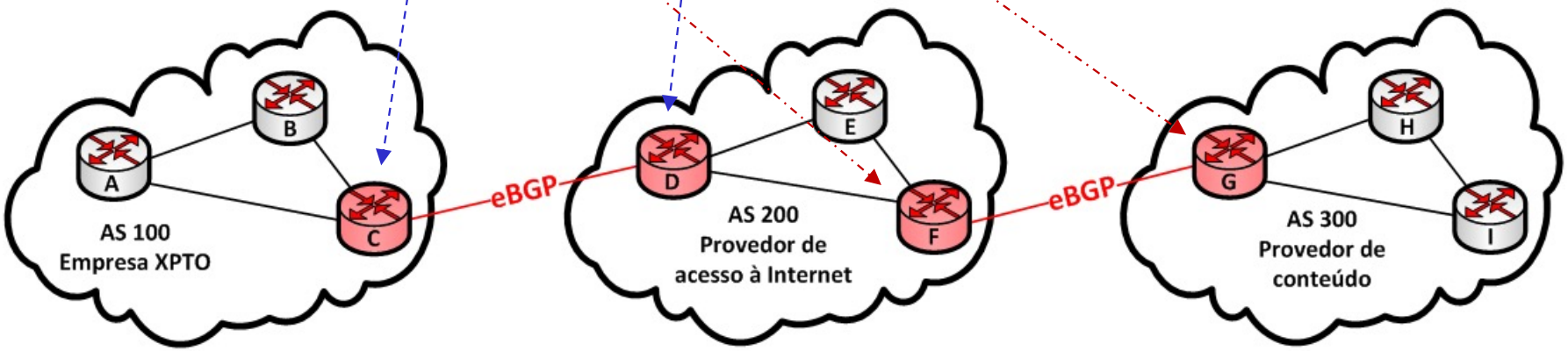


Figura adaptada de https://www.devmedia.com.br/bgp-border-gateway-protocol/29123 (21/6/2023)

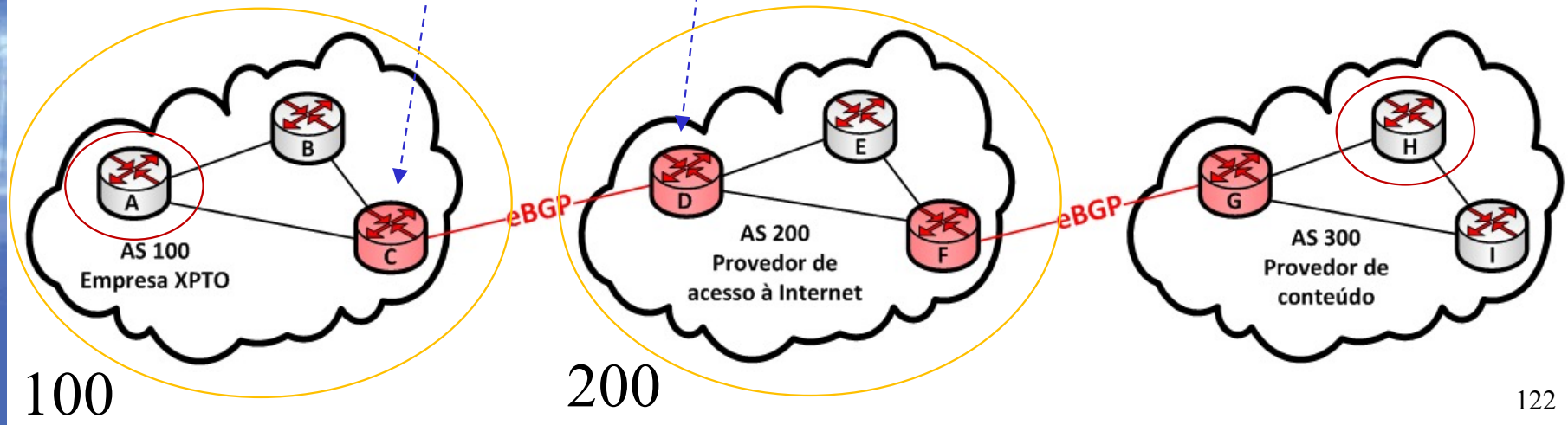
Exemplo de relações eBGP (1)

- Entre os roteadores ***C*** (AS 100) e ***D*** (AS 200)
- E entre ***F*** (AS 200) e ***G*** (AS 300).



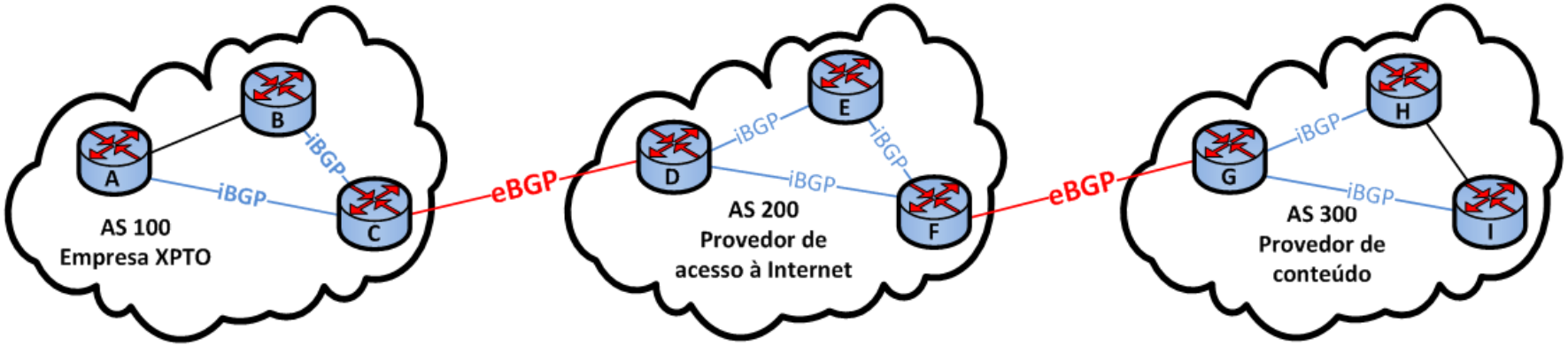
Exemplo de relações eBGP (2)

- Entre os roteadores ***C*** (AS 100) e ***D*** (AS 200)
- E entre ***F*** (AS 200) e ***G*** (AS 300).
- Os dados de um usuário conectado ao segmento de rede do **roteador A** da empresa XPTO, direcionados a um servidor web localizado no provedor de conteúdo à Internet, necessariamente serão encaminhados pela rede interna do **AS 100** ao **roteador C**, segundo sua política interna de roteamento, pois esta é a única interface entre os ASs 100 e 200.



2. Vizinhança "*internal BGP*" (iBGP)

- A comunicação **intra-sistemas autônomos** ocorre quando o protocolo BGP-4 é configurado entre dois ou mais roteadores pertencentes ao mesmo AS.
- Vizinhança internal BGP (iBGP):
  - Esta aplicação permite o compartilhamento das tabelas de rotas entre os roteadores do AS que **possuem conexões com outros sistemas autônomos,** e aqueles localizados internamente.



Exemplo de relações iBGP (1)

- Em **azul** estão as relações ***iBGP*** entre os roteadores internos dos ASs 100, 200 e 300.
- Os roteadores ***C*** (AS 100), ***D*** (AS 200), ***F*** (AS 200) e ***G*** (AS 300) **possuem ambos os tipos de vizinhanças**: ***iBGP*** e ***eBGP***.
- São estes os roteadores responsáveis pela propagação das rotas externas aos demais dispositivos dentro de seus sistemas autônomos.

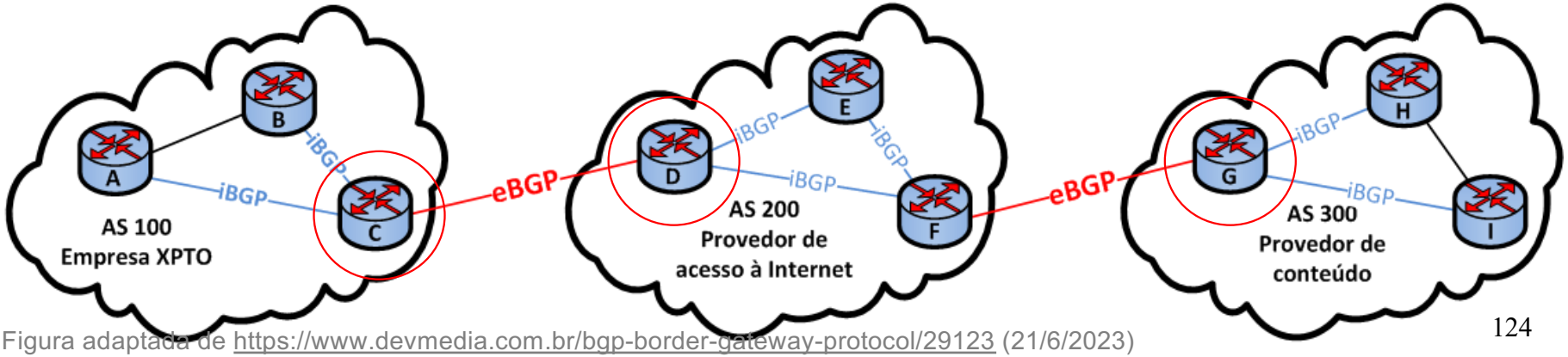
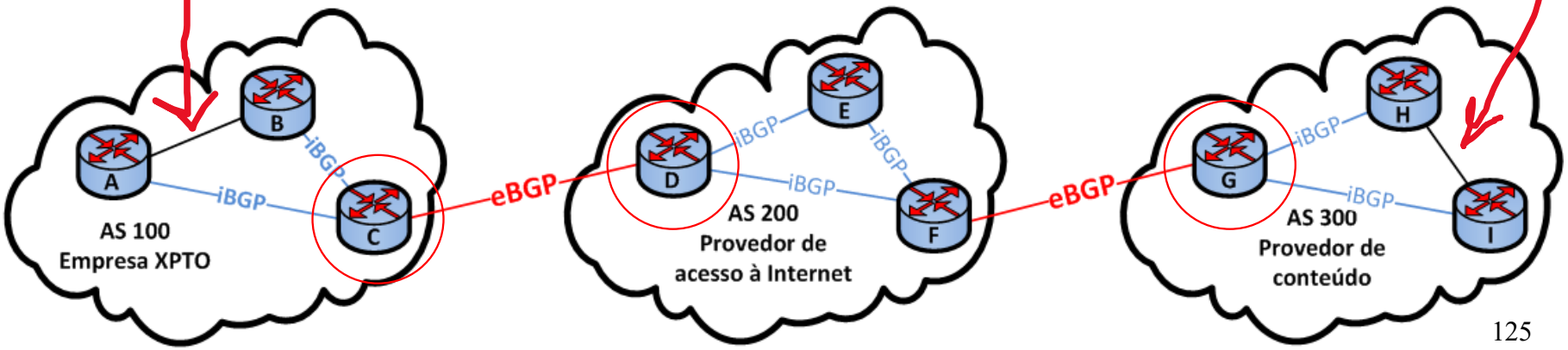


Figura adaptada de https://www.devmedia.com.br/bgp-border-gateway-protocol/29123 (21/6/2023)

Exemplo de relações iBGP (2)

- Em **azul** estão as relações iBGP entre os roteadores internos dos ASs 100, 200 e 300.
- Os roteadores ***C*** (AS 100), ***D*** (AS 200), ***F*** (AS 200) e ***G*** (AS 300) **possuem ambos os tipos de vizinhanças**: ***iBGP*** e ***eBGP***.

Observe que aqui **não há nem iBGP nem eBGP**. É uma situação que pode ocorrer por decisão administrativa

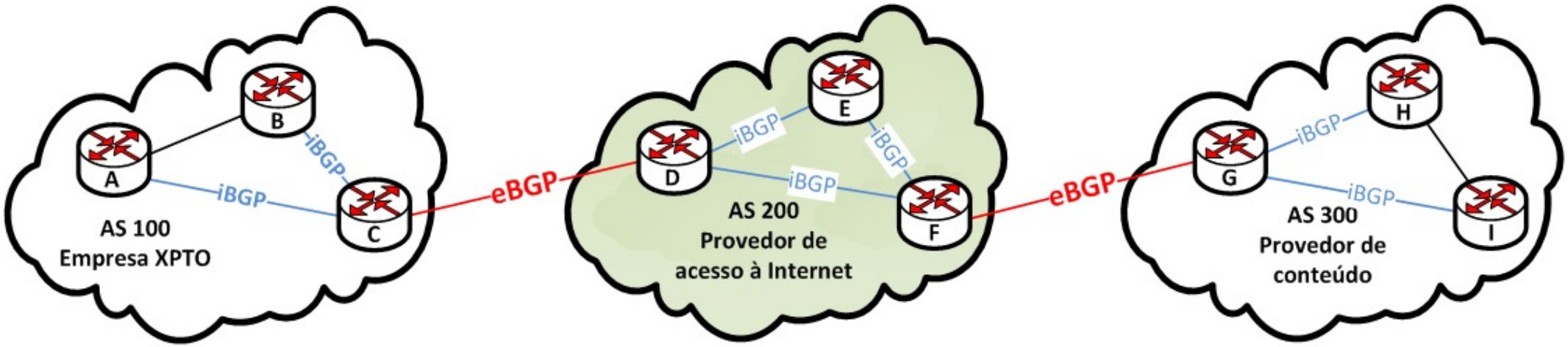


3. Trânsito entre ASs

- O trânsito entre os sistemas autônomos é realizado por **ASs que não possuem relação direta com o remetente ou destinatário dos pacotes IP**.
  - Ou seja, um AS de trânsito apenas encaminha os dados de terceiros e não para si mesmo.
  - No geral são grandes ISPs e operadoras de telecomunicações que oferecem trânsito IP aos seus clientes,
  - Criam uma malha de interconexão entre os diferentes sistemas autônomos espalhados pela região geográfica na qual possuem cobertura.

Exemplo de trânsito entre ASs

- Essa figura representa o **AS 200** (Provedor de Acesso à Internet) como uma entidade de trânsito, que conecta a Empresa XPTO (**AS 100**) ao Provedor de Conteúdo (**AS 300**).



Provedor de Trânsito

Regras de Topologia BGP-4

Por padrão, três importantes regras devem ser consideradas no projeto de uma topologia de rede com suporte ao protocolo BGP-4.

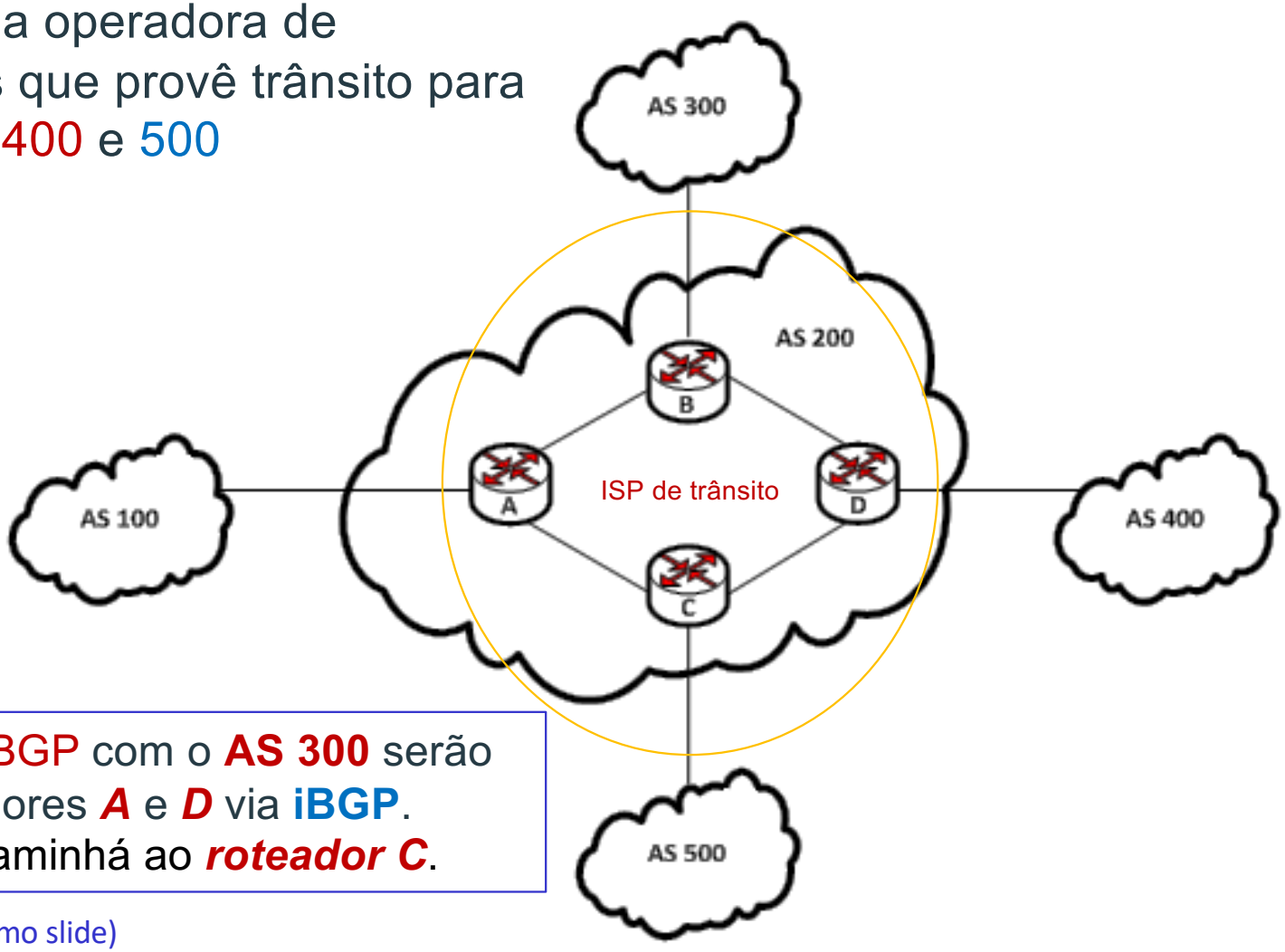
Veremos em seguida...

Regra de Topologia BGP No 1.

- As rotas externas recebidas por uma conexão **eBGP** podem ser propagadas para vizinhanças **eBGP** e **iBGP**.
- Entretanto, para evitar *loops* de roteamento, por padrão, **as rotas aprendidas por iBGP** **não** são encaminhadas para outros roteadores em sessões iBGP (ver figura adiante).
  - Ou seja, o que se APRENDE por iBGP não PROPAGA por iBGP.
- Portanto, um AS **possivelmente empregará um IGP** para o compartilhamento interno das tabelas de roteamento de seus dispositivos.

(Mas isso pode não ser verdade sempre; veja o Apêndice no final desses slides para mais informações.)

O AS200 é de uma operadora de telecomunicações que provê trânsito para os **AS**s 100, 300, 400 e 500

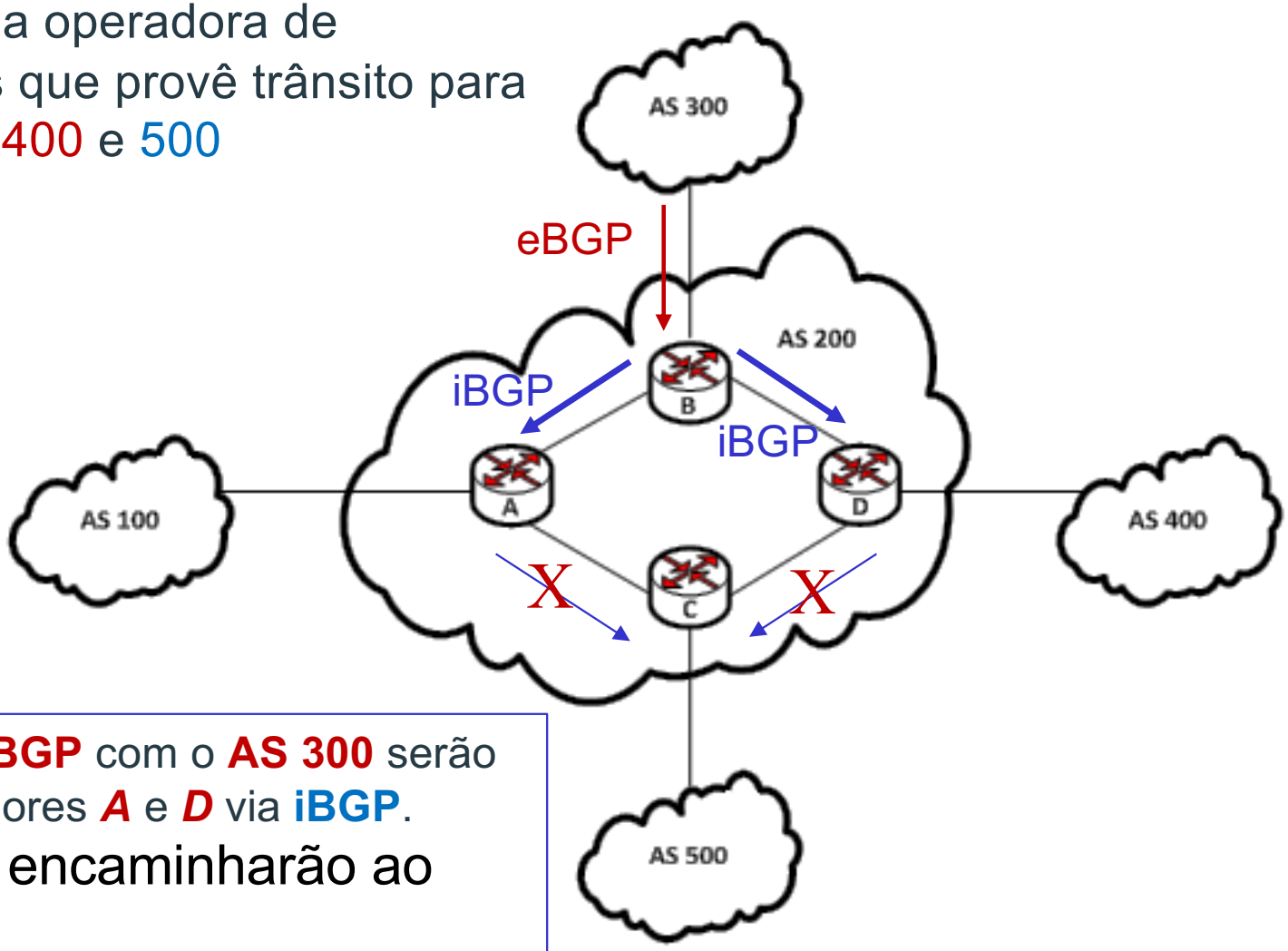


- As rotas na conexão eBGP com o **AS 300** serão propagadas aos roteadores ***A*** e ***D*** via **iBGP**.
- Mas *A* e *D* **não** as encaminhá ao ***roteador C***.

(Veja próximo slide)

Figura adaptada de https://www.devmedia.com.br/bgp-border-gateway-protocol/29123 (21/6/2023)

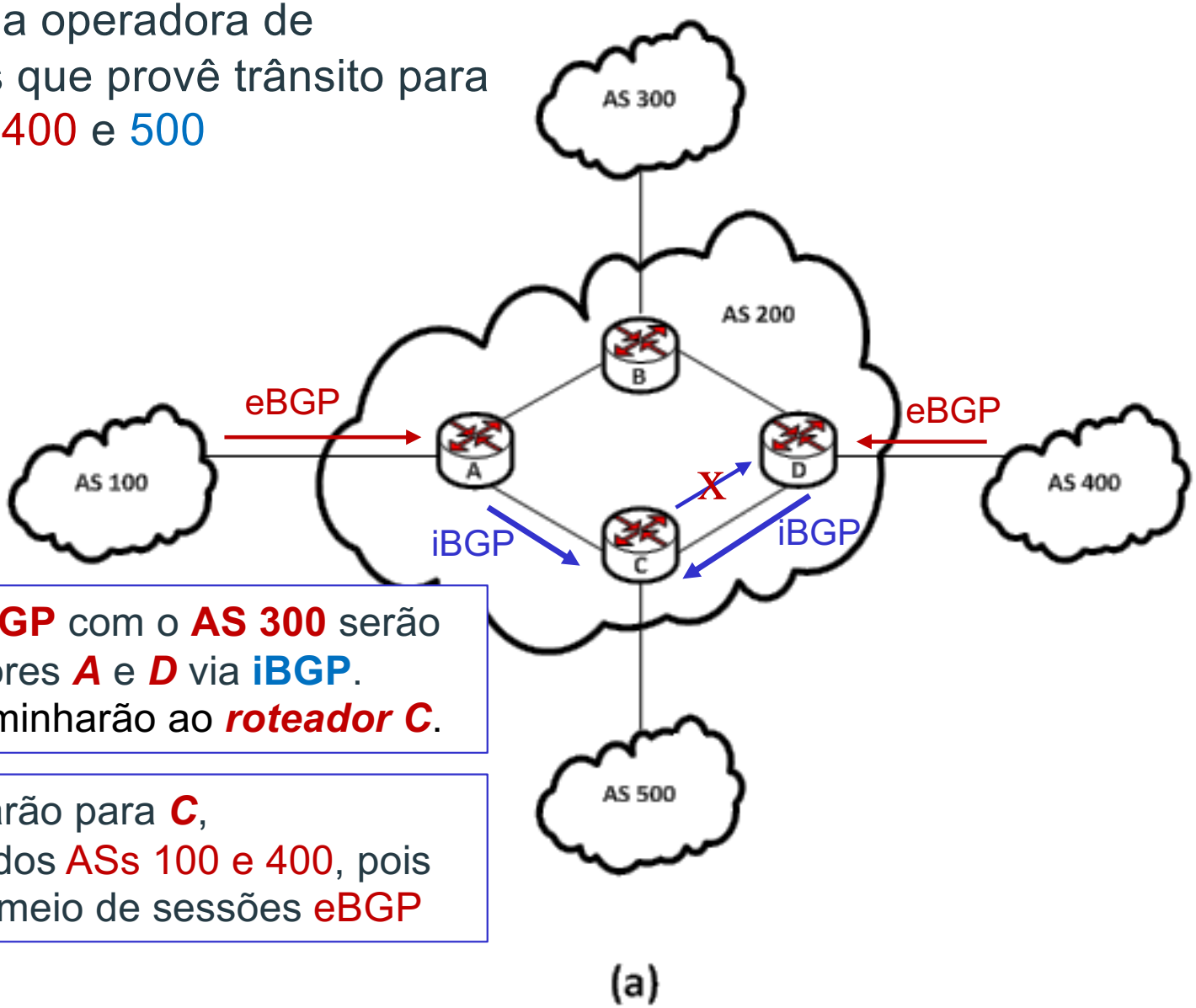
O AS200 é de uma operadora de telecomunicações que provê trânsito para os **AS**s 100, 300, 400 e 500



- As rotas na conexão **eBGP** com o **AS 300** serão propagadas aos roteadores ***A*** e ***D*** via **iBGP**.
- Mas *A* e *D* **<u>não</u>** as encaminharão ao ***roteador C***.

Figura adaptada de https://www.devmedia.com.br/bgp-border-gateway-protocol/29123 (21/6/2023)

O AS200 é de uma operadora de telecomunicações que provê trânsito para os **AS**s 100, 300, 400 e 500



(a)

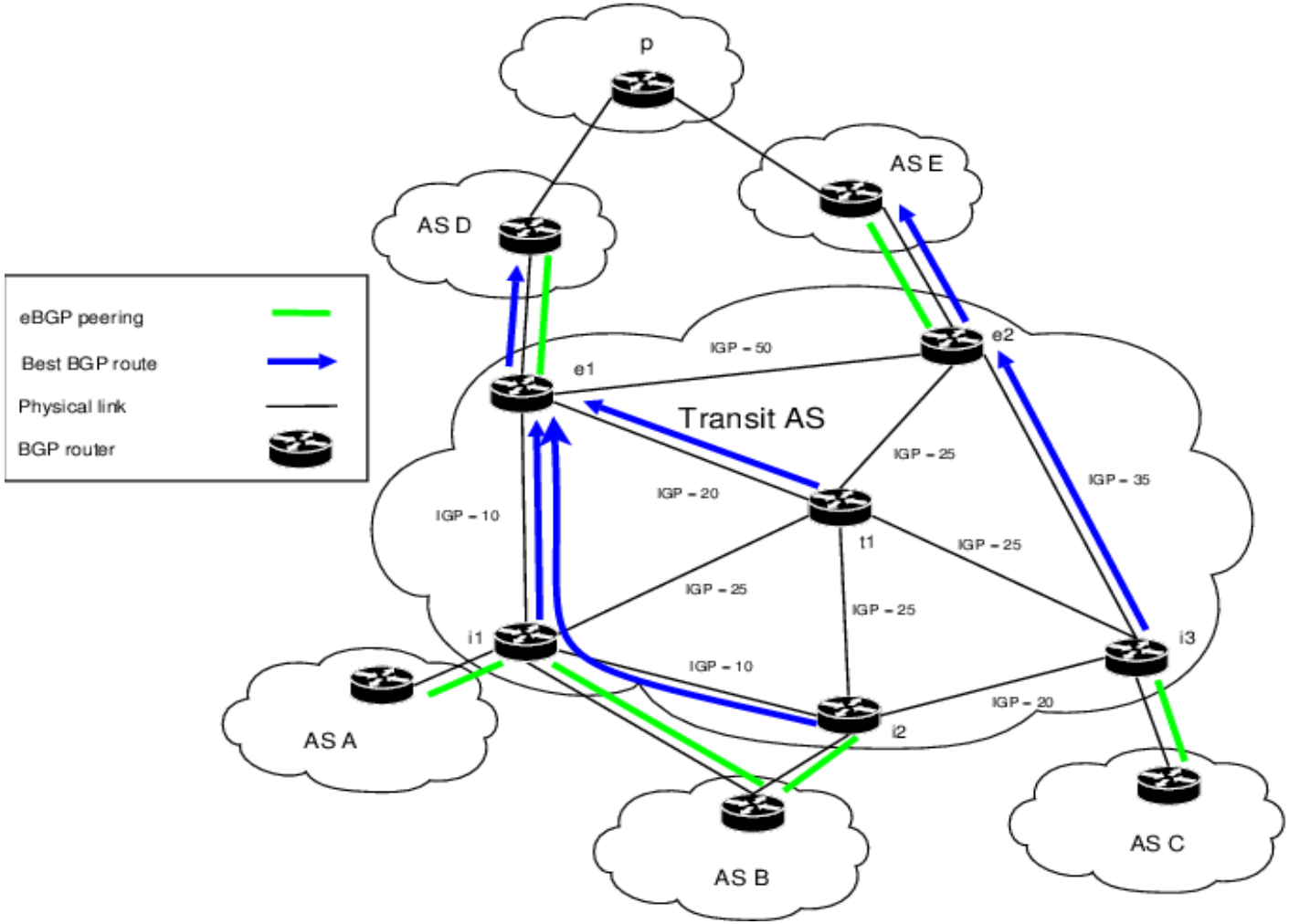
- As rotas na conexão **eBGP** com o **AS 300** serão propagadas aos roteadores ***A*** e ***D*** via **iBGP**.
- Mas estes **não** as encaminharão ao ***roteador C***.

Por outro lado, ***A*** e ***D*** enviarão para ***C***, respectivamente, as rotas dos ASs 100 e 400, pois estas foram recebidas por meio de sessões eBGP

Regra de Topologia BGP No. 2

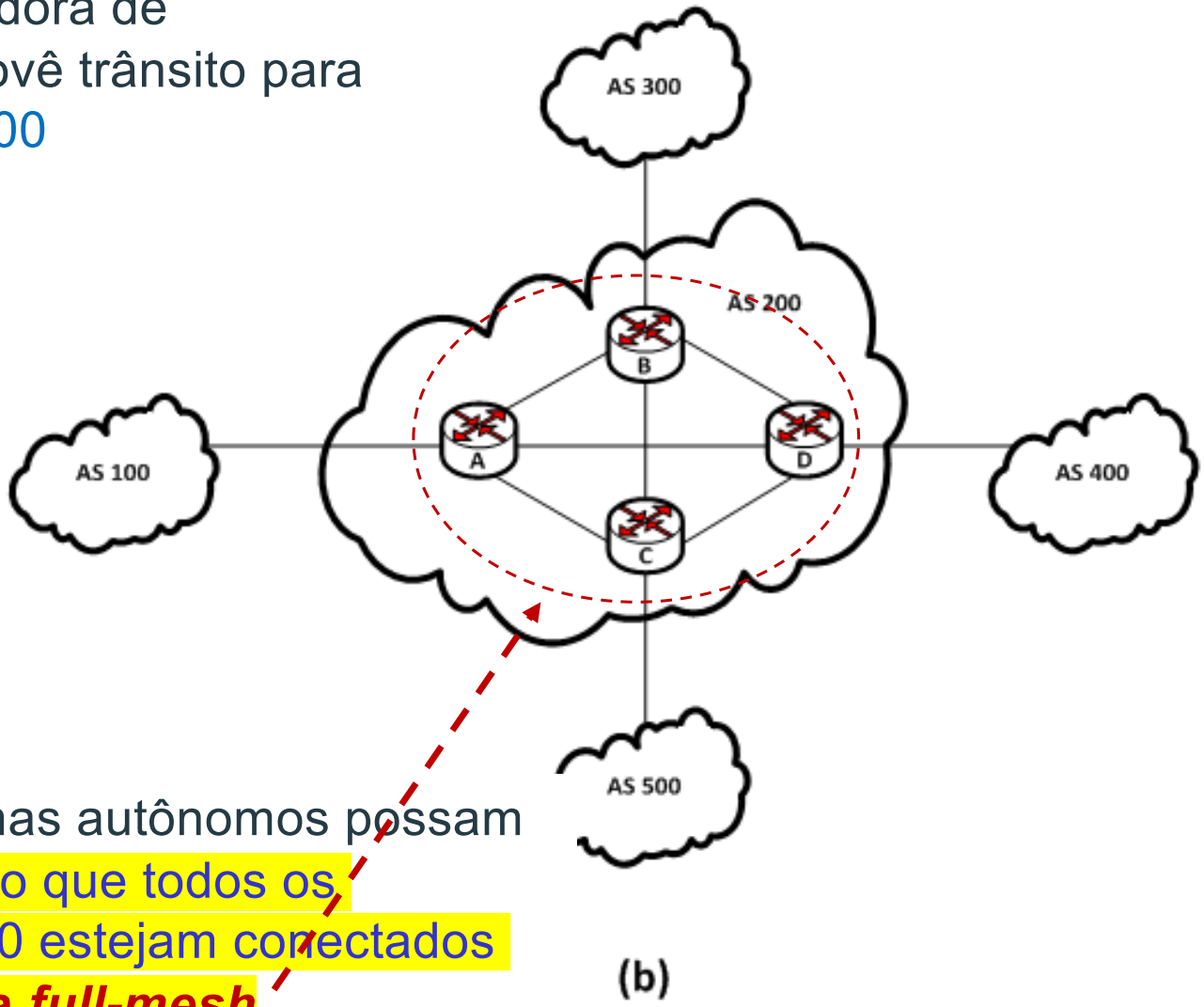
- Todos os roteadores configurados com o protocolo BGP-4 devem estar conectados logicamente entre si (em vizinhanças *iBGP*).
  - **Formando uma topologia de malha completa (*full-mesh*).**
- Esta arquitetura é necessária para atender a restrição imposta pelo comportamento da Regra 1.
- Veja: https://notes.networklessons.com/bgp-ibgp-full-mesh-peering

Topologia *BGP Full Mesh*



Créditos: https://www.researchgate.net/figure/Example-topology-with-iBGP-full-mesh_fig1_226810646

O AS200 é de uma operadora de telecomunicações que provê trânsito para os **AS**s 100, 300, 400 e 500



(b)

Assim, para que os sistemas autônomos possam se comunicar, é necessário que todos os roteadores BGP do AS 200 estejam conectados logicamente em **topologia *full-mesh***.

Regra de Topologia BGP No. 3

- Somente são propagadas para os *BGP peers* (vizinhos) as rotas que:
  - Que estejam na tabela de roteamento dos dispositivos;
  - **Sejam redes diretamente conectadas;**
  - **Sejam prefixos IP configurados estaticamente;** ou
  - Aprendidos por intermédio de protocolos dinâmicos IGP.
- Para evitar *BGP Hijacking* ou *IP Spoofing*.
  - Roteamento de pacotes que não pertençam à rede interna.
    - *Egress Filtering*: não permitir que pacotes IP de endereços que não pertencem à rede possa sair dela.
    - Exemplo: se a rede interna da UNESP possui o range `200.145/16` , então **não** podem sair delas pacotes IP que tenham como endereço de origem, por exemplo, `200.9.78/24`.

Atributos do BGP-4

- O protocolo BGP-4 é sustentado pelo **algoritmo vetor de caminhos** (*path vector*).
- A partir de dados recebidos dos demais sistemas autônomos, cria-se um vetor que armazena os ASNs que formam o caminho até cada uma das redes remotas.
- Uma vez que as diversas vizinhanças BGP divulgam estas informações, **é possível calcular o menor caminho para uma determinada rede IP**
- Esse é chamado de "*AS-path*".

Diferentes caminhos (*AS-paths*) entre as empresas *A* (AS 100) e *B* (AS 500)

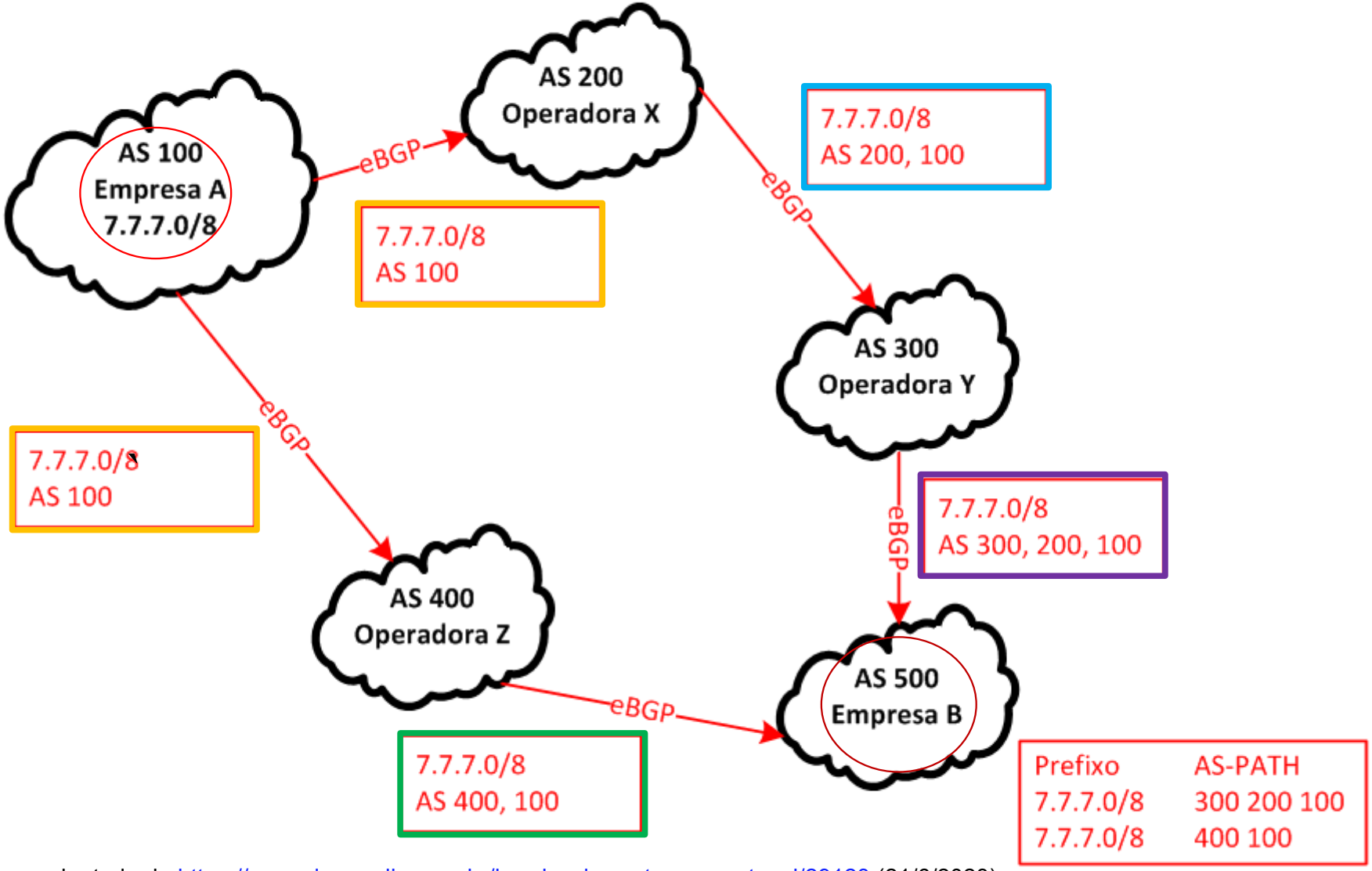
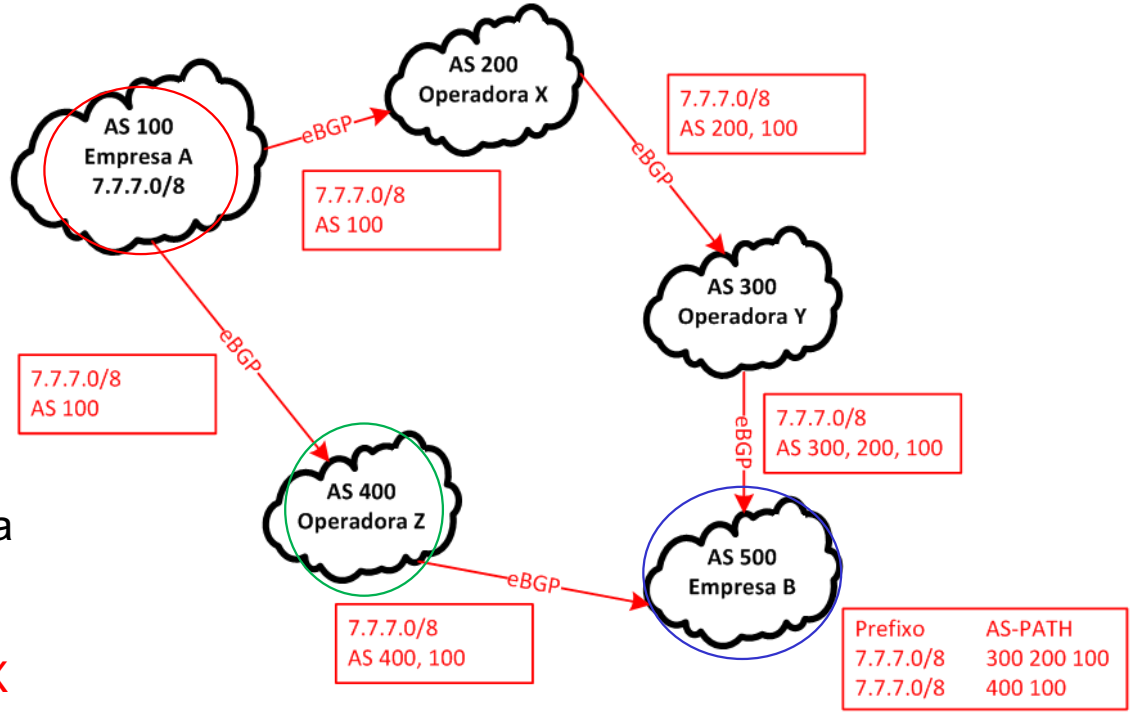


Figura adaptada de https://www.devmedia.com.br/bgp-border-gateway-protocol/29123 (21/6/2023)

Diferentes caminhos (*AS-paths*) entre as empresas *A* (AS 100) e *B* (AS 500)

A **Empresa B** pode acessar o prefixo `7.7.7.0/8` (**Empresa A**) por intermédio das operadoras ***Y (AS 300)*** ou ***Z (AS 400).***



É importante notar que a **Operadora Z** possui uma conexão eBGP direta com a Empresa A (AS100), enquanto que a operadora Y depende do serviço de trânsito BGP prestado pela operadora X (AS 200).

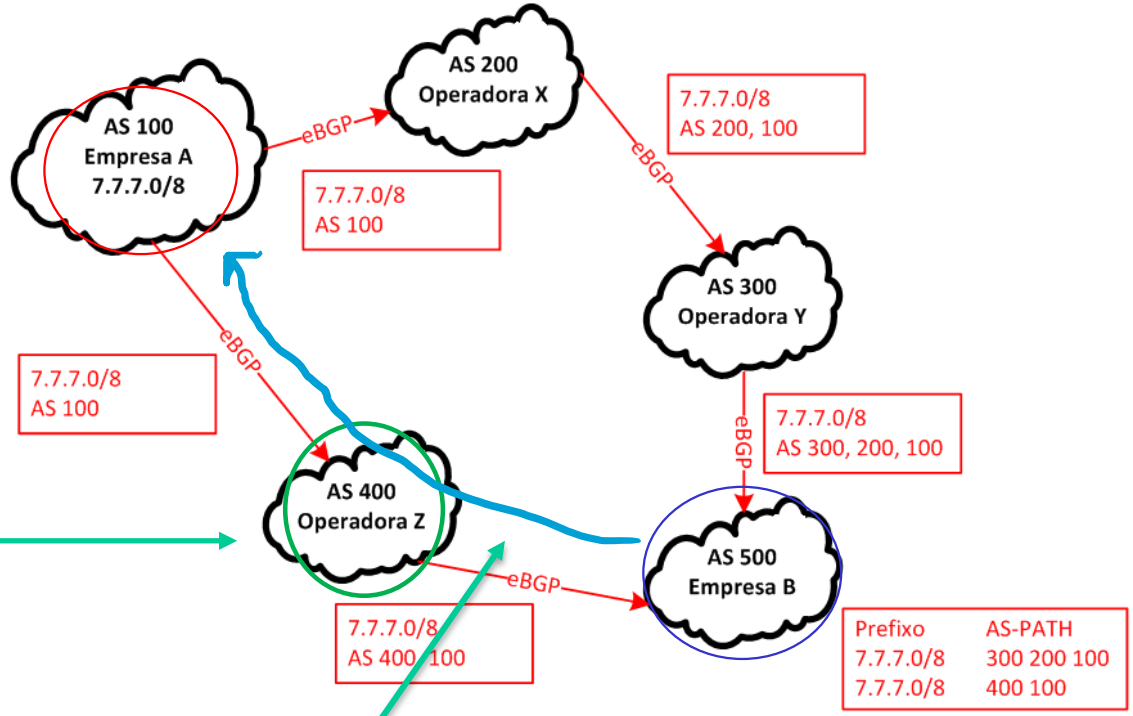
Neste cenário, a rota de menor caminho é pela **Operadora Z**, pois os **dados passam somente por um sistema autônomo até o destino final**.

Figura adaptada de https://www.devmedia.com.br/bgp-border-gateway-protocol/29123 (21/6/2023)

Diferentes caminhos (*AS-paths*) entre as empresas *A* (AS 100) e *B* (AS 500)

A **Empresa B** pode acessar o prefixo `7.7.7.0/8` (**Empresa A**) por intermédio das operadoras ***Y (AS 300)*** ou ***Z (AS 400).***

É importante notar que a **Operadora Z** possui uma conexão eBGP direta com a Empresa A (AS100), enquanto que a operadora Y depende do serviço de trânsito BGP prestado pela operadora X (AS 200).



Neste cenário, a rota de menor caminho é pela **Operadora Z**, pois os **dados passam somente por um sistema autônomo até o destino final**.

Exemplo de uma tabela BGP real

Prefixo	Próximo Salto	AS-PATH
192.0.2.0/24	198.51.100.1	64500
203.0.113.0/24	198.51.100.2	64501 64500
198.51.100.0/24	203.0.113.1	64502 64501 64500
198.18.0.0/15	203.0.113.2	64503 64502 64501 64500
10.0.0.0/8	192.0.2.1	64504

EXERCÍCIO

Resumo BGP (1)

1. Anúncio de Rotas via BGP:

- Quando um roteador de borda de um ASN deseja anunciar suas redes para outros roteadores em diferentes ASNs, ele utiliza o BGP.
- O roteador de borda envia mensagens BGP que contêm informações sobre os prefixos de rede (ou blocos de endereços IP) que ele pode alcançar, juntamente com o ASN associado.

EXERCÍCIO

Resumo BGP (2)

2. Tabelas de Roteamento BGP:

- Roteadores BGP mantêm tabelas de roteamento que listam todos os prefixos conhecidos e os caminhos AS associados a cada prefixo.
- Estas tabelas são atualizadas dinamicamente à medida que novos anúncios BGP são recebidos.
- Cada entrada na tabela de roteamento inclui:
  - O prefixo de rede;
  - O ASN de origem; e
  - A sequência de ASNs pelos quais o prefixo foi anunciado (chamado de *AS-PATH*).

EXERCÍCIO

Resumo BGP (3)

3. Processamento de Anúncios BGP:

- Quando um roteador BGP recebe um anúncio de rota, ele:
  - Atualiza sua tabela de roteamento;
  - E repassa o anúncio para seus vizinhos BGP, ajustando o AS-PATH para incluir seu próprio ASN.
- O AS-PATH ajuda a evitar loops de roteamento.
- Pois os roteadores podem detectar se um anúncio passou por um ASN anteriormente.

EXERCÍCIO

Resumo BGP (4)

4. Tomada de Decisões de Roteamento:

- Com base nas tabelas de roteamento BGP, o roteador decide o melhor caminho para encaminhar pacotes destinados a redes específicas.
- As decisões são baseadas em critérios como comprimento do AS-PATH, políticas locais de roteamento, entre outros.

EXERCÍCIO

Exemplo prático BGP

- Imagine que o `ASN 64500` deseja anunciar que ele pode rotear pacotes para a rede `192.0.2.0/24`.
- O roteador de borda no `ASN 64500` enviará um anúncio BGP para seus vizinhos (outros ASNs), indicando que `192.0.2.0/24` pode ser alcançada através do `ASN 64500`.
  - Os roteadores vizinhos receberão este anúncio, atualizarão suas tabelas de roteamento BGP para incluir a nova rota, e repassarão o anúncio para outros vizinhos.
- Um roteador no `ASN 64501` que recebeu o anúncio pode agora saber que a rede `192.0.2.0/24` é roteada pelo `ASN 64500`, e ajustará suas rotas de acordo.

EXERCÍCIO

BGP-4 e *Looping Avoidance*

- Uma importante melhoria introduzida no algoritmo vetor de caminhos está relacionada à prevenção dos loops de roteamento.
- Os roteadores verificam no ***as-path*** de cada prefixo se ele foi originado ou propagado pelo seu próprio sistema autônomo.
- Se esta observação for verdadeira o ***as-path*** é DESCARTADO para evitar *loopings*.

EXERCÍCIO

BGP-4: menor caminho não é o único atributo

- A escolha da melhor rota não é necessariamente baseada no menor caminho.
- Outros atributos podem ser usados.
- Estes atributos descrevem um conjunto de propriedades dos prefixos IP.
- São empregados na decisão da melhor opção para o encaminhamento dos pacotes, quando há múltiplos caminhos.
- Os atributos que podem ser usados no BGP-4 não serão discutidos aqui.

BGP-4 Conclusão

- Os 3 cenários básicos de roteamento do BGP sustentam:

1. A comunicação entre roteadores internos por meio de vizinhanças iBGP.
2. O envio e o recebimento das tabelas de roteamento entre sistemas autônomos distintos, desde que estes possuam conexões eBGP estabelecidas.
3. O transporte de dados por organizações que não estejam diretamente relacionadas à origem ou ao destino dos pacotes IP, por intermédio dos sistemas autônomos de trânsito.

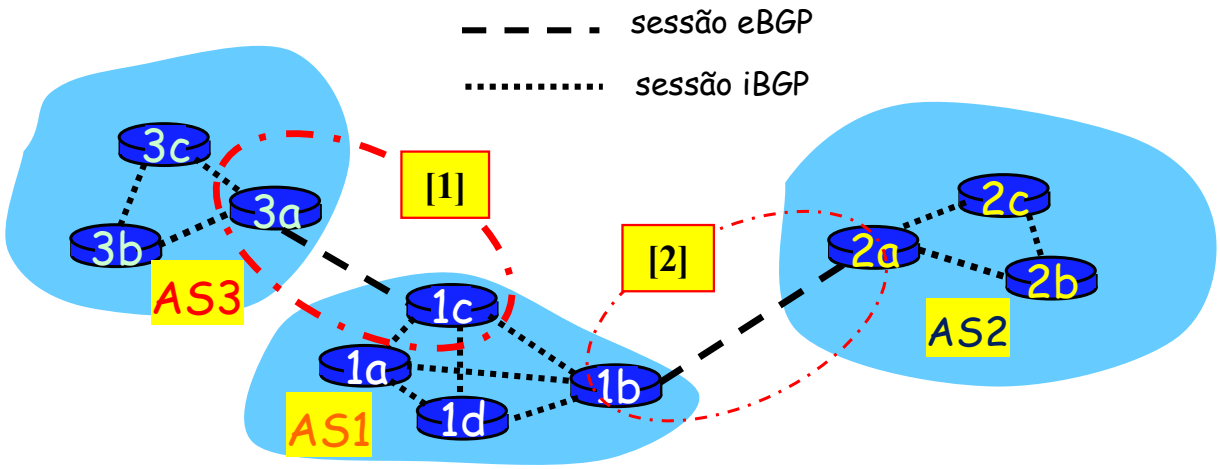
Tópicos não cobertos em BGP aqui

- A discussão sobre BGP-4 **não** cobriu os seguintes tópicos:
  - Mensagens BGP-4.
  - Atributos do BGP-4.
  - Máquina de Estado do BGP-4.
  - Comunicação entre os roteadores BGP-4 e as ações resultantes.
- Entretanto esses podem ser de interesse aos alunos que desejarem se aprofundar ou seguir sua formação na área de Redes e Internet.
- Para uma introdução sobre esses tópicos não cobertos, recomenda-se a leitura do artigo introdutório "*BGP - Border Gateway Protocol*" da Revista Infra Magazine No. 13
  - Disponível em https://www.devmedia.com.br/bgp-border-gateway-protocol/29123 (20/6/2023)

EXERCÍCIO

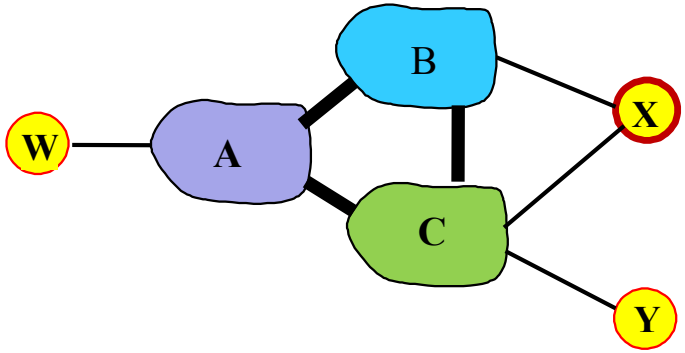
Distribuindo informações de atingibilidade

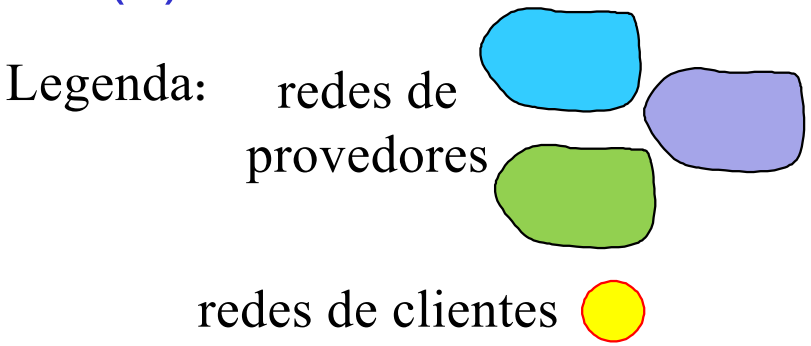
- [1] Usando sessão eBGP entre ***3a*** e ***1c***, ***AS3*** envia informação de atingibilidade do prefixo para ***AS1***.
  - ***1c*** pode então usar **iBGP** para distribuir nova informação de prefixo a todos os roteadores em ***AS1***
- [2] ***1b*** pode então reanunciar nova informação de atingibilidade para ***AS2*** por sessão **eBGP** ***1b***-para-***2a***
- Quando roteador descobre novo prefixo, ele cria entrada para prefixo em sua tabela de repasse.



EXERCÍCIO

Exemplo: Política de roteamento BGP (1)

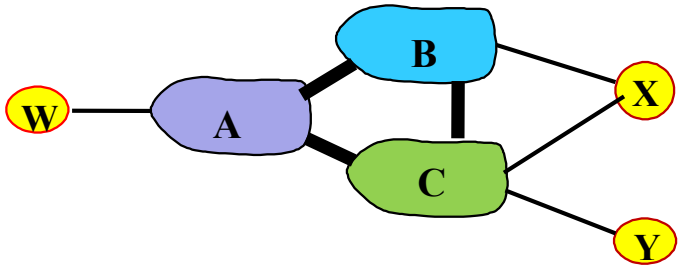


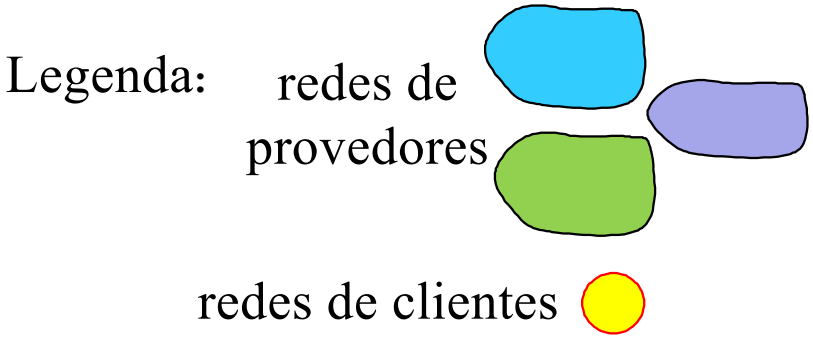


- ***A, B, C*** são redes de provedores.
- *X, Y, W* são redes de clientes (de provedores).
- ***X*** é *dual-homed*: conectada a duas redes de provedores.
  - ***X* não** quer deixar *B* rotear por meio dele (X) para chegar a ***C***.
  - Então, ***X* não anunciará a *B* uma rota para *C*.**

EXERCÍCIO

Exemplo: Política de roteamento BGP (2)



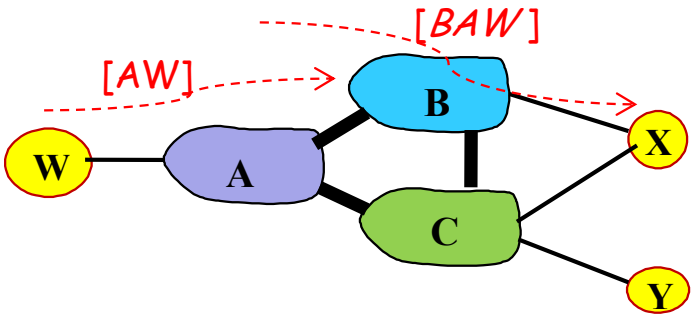


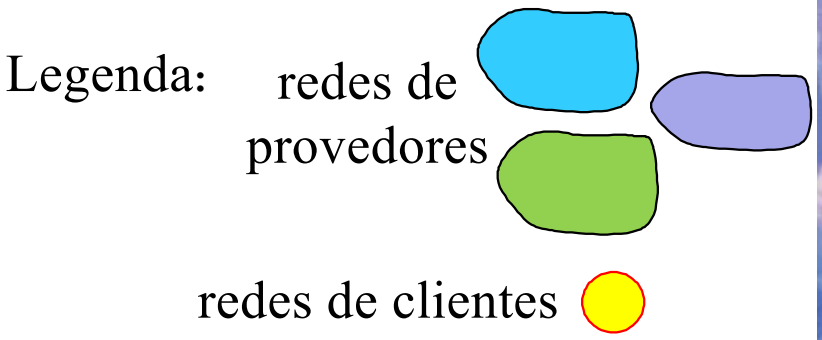
- ***A*** anuncia caminho ***AW*** para *B.*
- ***B*** anuncia caminho ***BAW*** para ***X***.
- ***Pergunta: B*** deve anunciar caminho ***BAW*** para ***C*** ?

?

EXERCÍCIO

Exemplo: Política de roteamento BGP (2)





- ***A*** anuncia caminho ***AW*** para *B.*
- ***B*** anuncia caminho ***BAW*** para ***X***.
- ***B*** deve anunciar caminho ***BAW*** para ***C*** ?
- **De forma alguma!**
  - ***B*** não recebe "retorno" para roteamento ***CBAW***.
  - Nem ***W*** nem ***C*** são clientes de ***B***
    - ***B*** quer forçar ***C*** a rotear para ***W*** por meio de ***A.***
    - ***B*** quer rotear *apenas* **para / de** seus clientes!

Final da Camada de Rede

Vimos neste capítulo:

- Serviços da camada de rede.
- Roteamento e Roteador.
- Endereços IP.
- Fragmentação e MTU.
- Sub-redes e máscaras.
- Princípio de roteamento: seleção de caminhos.
- Algoritmos de roteamento.
  - *Link state e distance vector.*
- Roteamento hierárquico.
- Sistemas Autônomos.
- Protocolos de roteamento da Internet.

Apêndice 1

Um sistema autônomo sempre empregará um IGP para o compartilhamento interno das tabelas de roteamento?

Apêndice 1/1 – IGP versus iBGP

Um sistema autônomo sempre empregará um IGP para o compartilhamento interno das tabelas de roteamento?

- Não necessariamente.
- Embora a prática comum em muitas redes de sistemas autônomos (AS) seja empregar um protocolo de roteamento interno (IGP) para o compartilhamento de informações de roteamento internas, nem todos os sistemas autônomos precisam utilizar um IGP.
  - O uso de IGP é uma prática comum em muitos sistemas autônomos para garantir eficiência, escalabilidade e rápida convergência, mas nem todos os AS precisam de um IGP.
- A necessidade de um IGP depende da complexidade, tamanho e dinâmica da rede.
  - Em redes pequenas, ou ambientes controlados, podem ser suficiente usar rotas estáticas, ou a utilização exclusiva de iBGP.

Apêndice 1/2 – IGP versus iBGP

Situações em que um **IGP** é Comumente Utilizado:

- **Redes Grandes e Complexas:**
  - Em grandes sistemas autônomos com muitos roteadores e links, **um IGP** como OSPF (*Open Shortest Path First*) ou IS-IS (*Intermediate System to Intermediate System*) é utilizado para fornecer rápida convergência e descoberta dinâmica de rotas internas.
    - Ele garante que todos os roteadores dentro do AS tenham conhecimento completo da topologia interna e possam rapidamente recalcular rotas em caso de falhas.
- **Redes com Vários Caminhos Internos:**
  - Quando há múltiplos caminhos entre pontos dentro do AS, um IGP ajuda a escolher o caminho mais eficiente com base em métricas como custo, banda, atraso, etc.
- **Interoperabilidade e Escalabilidade:**
  - Em redes que precisam ser altamente escaláveis e interoperáveis, um IGP é necessário para manter a consistência e a eficiência da distribuição de rotas internas, especialmente quando integradas com BGP para o roteamento externo.

Apêndice 1/3 – IGP versus iBGP

Situações em que um IGP pode **não** ser necessário:

- **Redes Pequenas e Simples:**
  - Em redes muito pequenas e estáticas, onde há poucos roteadores e a topologia é simples, a necessidade de um IGP pode ser minimizada ou até mesmo eliminada. Neste caso, rotas estáticas podem ser configuradas manualmente sem a necessidade de um protocolo dinâmico.
- **Provedores de Serviços de Data Center:**
  - Alguns provedores de serviços de data center ou ambientes fortemente controlados podem usar apenas BGP internamente (iBGP) com uma abordagem de roteamento estático, especialmente se a topologia é bem conhecida e raramente muda.
- **Redes com Camada de Sobreposição (*Overlay Networks*):**
  - Em algumas arquiteturas de rede modernas, como SD-WAN (Software-Defined Wide Area Network) ou certas implementações de redes definidas por software (SDN), uma camada de controle separada pode gerenciar o roteamento sem a necessidade de um IGP tradicional. Nessas redes, os túneis de sobreposição (como GRE, VXLAN) podem ser gerenciados por controladores SDN que mantêm a tabela de roteamento.

Apêndice 2

Utilizando OSPF em conjunto com iBGP

Apêndice 2/1 – OSPF e BGP

- O protocolo OSPF (Open Shortest Path First) não é utilizado para sessões iBGP, mas ele pode trabalhar em conjunto com o iBGP dentro de uma rede. Vamos esclarecer como esses dois protocolos se complementam e como eles são utilizados em um sistema autônomo (AS).

Diferenças Fundamentais:

1. **Finalidade:**
   - **OSPF:** É um protocolo de roteamento interno (IGP - Interior Gateway Protocol) usado para descobrir automaticamente e distribuir rotas dentro de um único sistema autônomo. Ele é baseado no algoritmo de estado de link e calcula o caminho mais curto para todos os destinos dentro da rede.
   - **iBGP:** É uma variante do BGP usada para trocar informações de roteamento entre roteadores BGP dentro do mesmo AS. Ele lida com rotas inter-AS (rotas aprendidas de outros sistemas autônomos) e precisa propagar essas informações internamente dentro do AS.
2. **Propagação de Rotas:**
   - **OSPF:** Distribui informações de roteamento sobre sub-redes e links internos dentro do AS.
   - **iBGP:** Distribui informações de rotas BGP, que podem incluir rotas externas aprendidas via eBGP e internas propagadas via iBGP.

Apêndice 2/2 – OSPF e BGP

Como OSPF e iBGP operam Juntos:

1. **Complementaridade:**
   - O OSPF é usado para fornecer conectividade IP básica e para garantir que todos os roteadores dentro do AS possam alcançar uns aos outros. Ele é responsável por propagar rotas para sub-redes internas e manter a rede interna convergida.
   - O iBGP é usado para distribuir rotas BGP (incluindo rotas externas) dentro do AS. O iBGP garante que as rotas recebidas de outros AS via eBGP sejam conhecidas por todos os roteadores BGP no AS.
2. **Configurando Sessões iBGP:**
   - **Estabelecendo Conectividade:** O OSPF pode ser usado para estabelecer conectividade IP básica entre os roteadores que participarão do iBGP. Isso garante que os roteadores BGP possam se comunicar diretamente.
   - **Distribuição de Rotas BGP:** Uma vez que a conectividade IP é garantida pelo OSPF, os roteadores podem estabelecer sessões iBGP entre si para trocar informações de roteamento BGP.
3. **Route Reflectors e OSPF:**
   - Em redes grandes onde um full mesh iBGP não é prático, são usados route reflectors para reduzir o número de sessões iBGP necessárias. O OSPF pode ajudar a garantir que todos os route reflectors e clientes estejam conectados de maneira eficiente.