Redes de Computadores I

Camada de Rede
Parte C: Roteamento Dinâmico

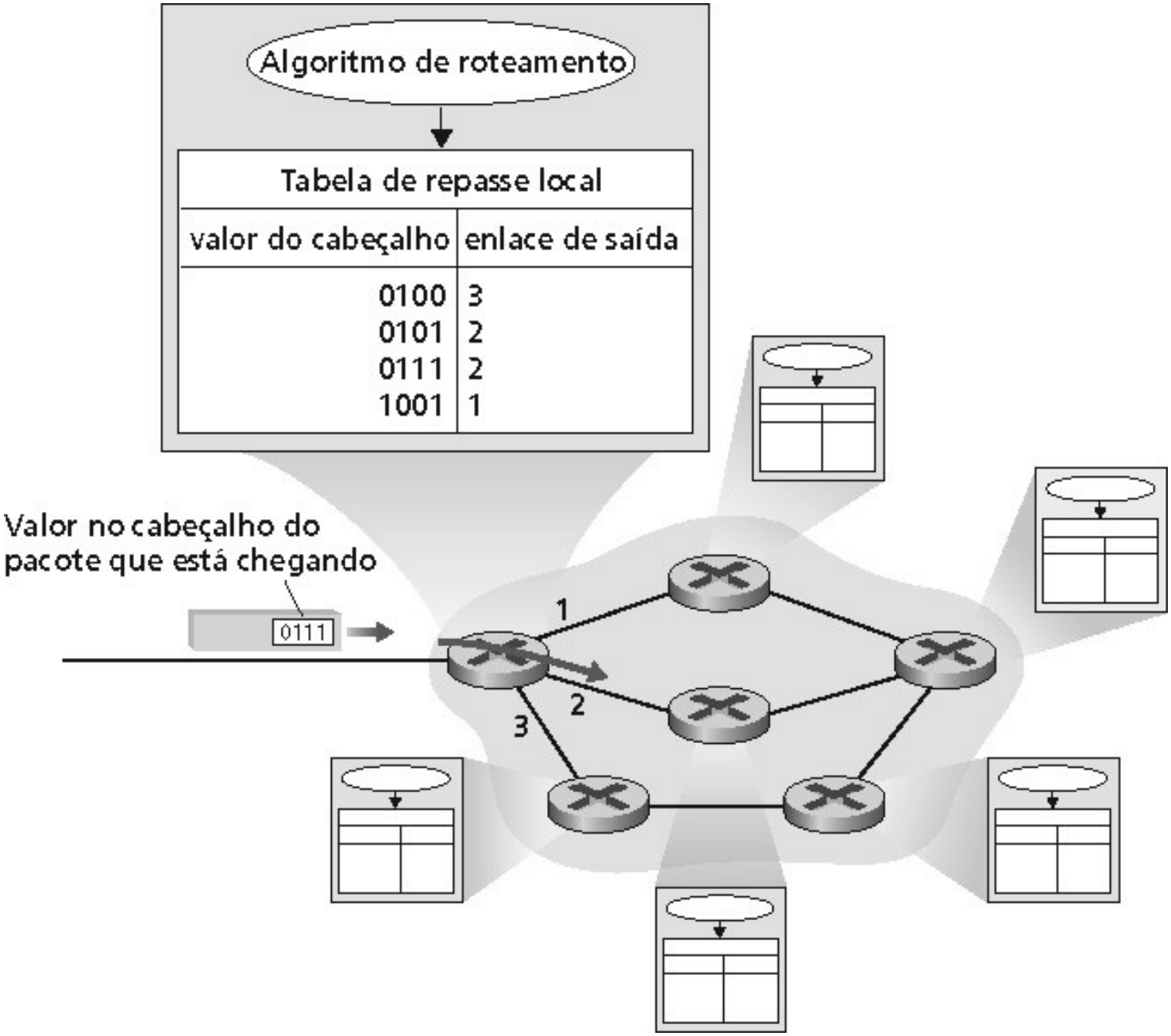
Prof. Ricardo Couto A. da Rocha
rcarocha@ufg.br
UFG – Regional de Catalão

Material baseado em:
- *Slides de referência do livro* ***Redes de Computadores e a Internet. Kurose & Ross****. Pearson*

Roteiro

- Algoritmos de Roteamento Dinâmico
  - Link State
  - Distance Vector
  - Roteamento Hierárquico
- Protocolos de Roteamento na Internet

Roteamento e Comutação



Abstração de Rotas e Custos

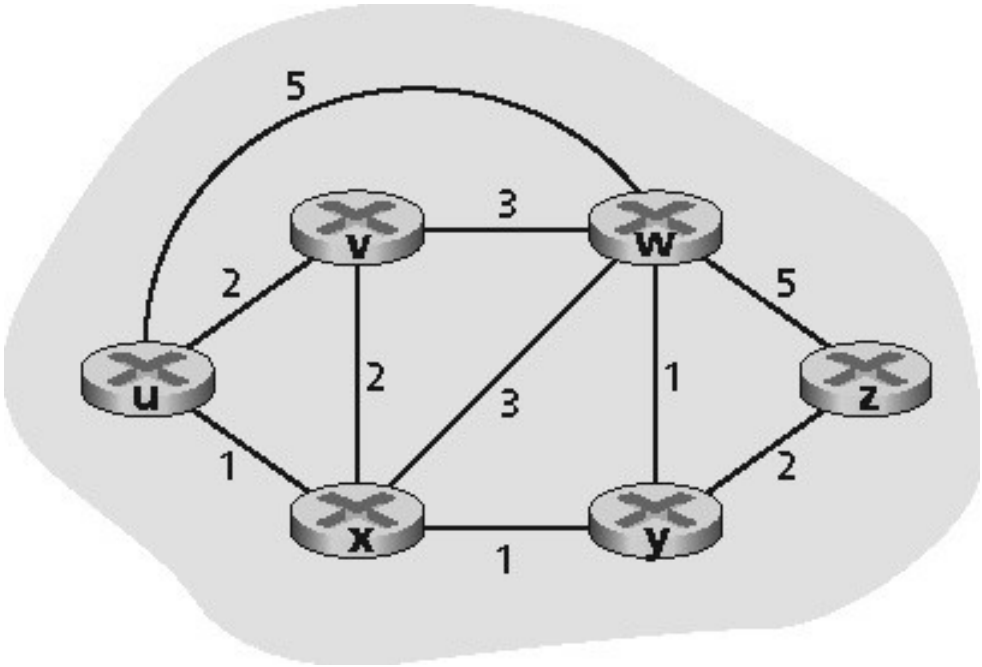


Gráfico: G = (N,E)

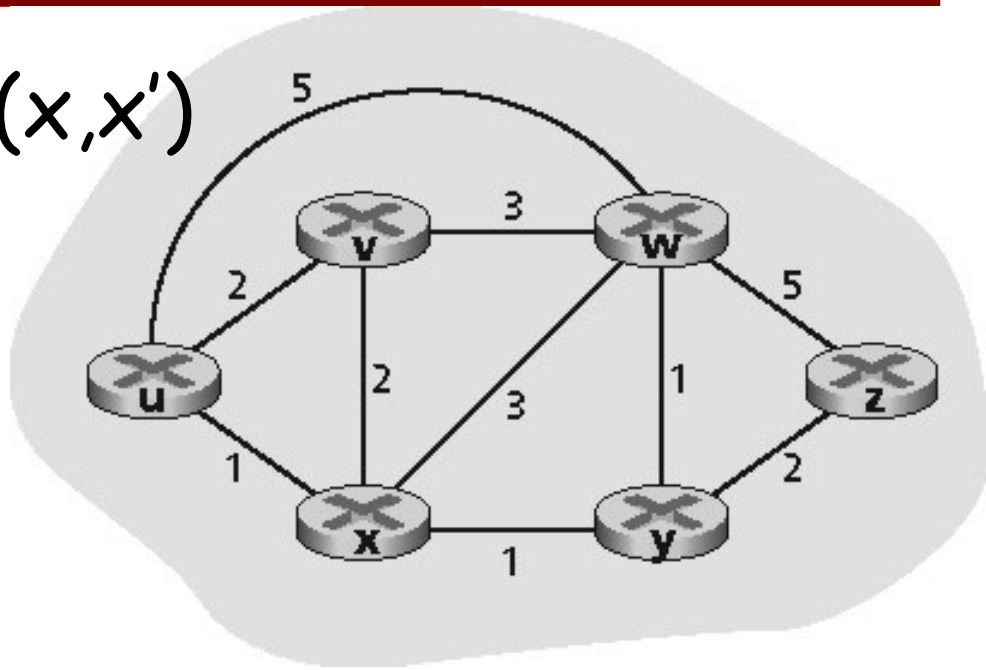
N = conjunto de roteadores = { u, v, w, x, y, z }

E = conjunto de links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Abstração de Rotas e Custos

c(x,x′) = custo do link (x,x′)

- Ex: c(w, z) = 5



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

- **Questão**: Qual é o caminho de menor custo entre **u** e **z** ?
- **Objetivo do Algoritmo de roteamento** → encontrar o caminho de menor custo

Classificação dos Algoritmos

- Usam **Informação global** ou **descentralizada**
- **Global**
  - Todos os roteadores têm informações completas da topologia e do custos dos enlaces
  - Algoritmos "link state"
- **Descentralizada**:
  - Roteadores só conhecem informações sobre seus vizinhos e os enlaces para eles
  - Processo de computação interativo, troca de informações com os vizinhos
  - Algoritmos "distance vector"
- Estático ou dinâmico?
  - **Estático** → As rotas mudam lentamente ao longo do tempo
  - **Dinâmico**:
    - As rotas mudam mais rapidamente
    - Podem responder a mudanças no custo dos enlaces
    - Usam Atualizações periódicas

Roteiro

- Algoritmos de Roteamento Dinâmico
  - **Link State**
  - Distance Vector
  - Roteamento Hierárquico
- Protocolos de Roteamento na Internet

Algoritmo de roteamento Link-state

- Usa **Algoritmo de Dijkstra**
- Topologia de rede e custo dos enlaces são conhecidos por todos os nós
  - Implementado via **"link state broadcast"**
  - **Todos os nós têm a mesma informação**
- Computa caminhos de menor custo de um nó (fonte) para todos os outros nós
  - Fornece uma tabela de roteamento para aquele nó
- **Convergência**: após **k** iterações, conhece o caminho de menor custo para k destinos
- Notação:
  - **C(i,j)**: custo do enlace do nó **i** ao nó **j**. Custo é infinito se não houver ligação entre **i** e **j**
  - **D(v)**: valor atual do custo do caminho da fonte ao destino **V**
  - **P(v)**: nó predecessor ao longo do caminho da fonte ao nó **v**, isto é, antes do **v**
  - **N'**: conjunto de nós cujo caminho de menor custo é definitivamente conhecido

Algoritmo de Dijsktra

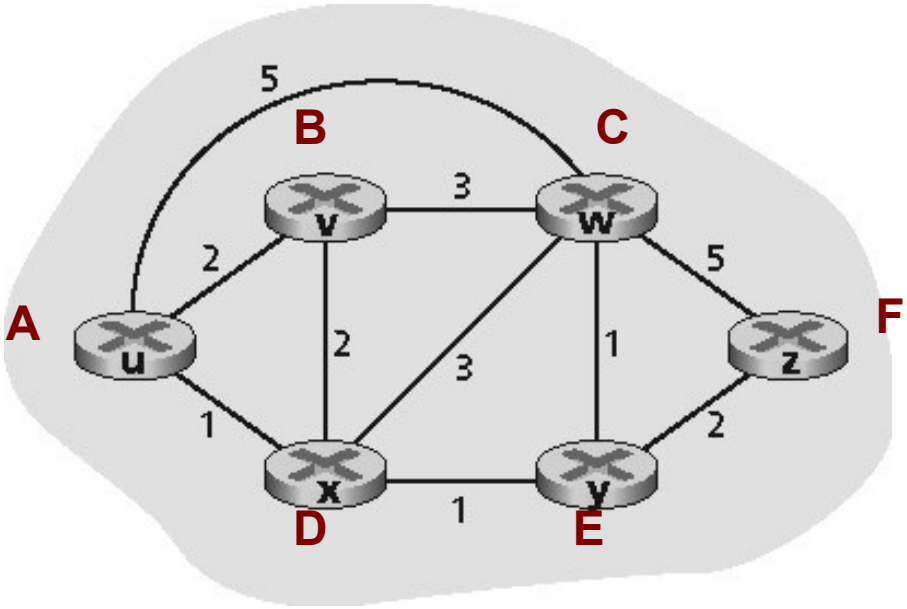
Inicialização:

```
   N' = {u}
   para todos os nós v
     se v é adjacente a u
       então D(v) = c(u,v)
       senão D(v) = ∞

Loop
   ache w não em N' tal que D(w) é um mínimo
   acrescente w a N'
   atualize D(v) para todo v adjacente a w e não em N':
      D(v) = min( D(v), D(w) + c(w,v) )
      /* novo custo para v é ou o custo anterior para v ou
         o menor custo de caminho conhecido para w mais o
         custo de w a v */
até que todos os nós estejam em N'
```

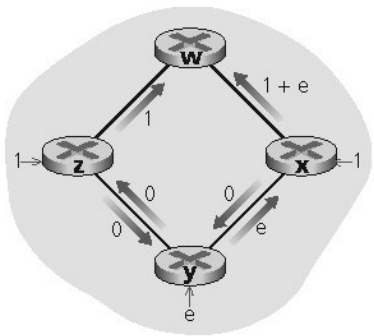
Exemplo: Algoritmo de Dijsktra

Passo	início N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	u	2,u	5,u	1,u	infinito	infinito
1	ux	2,u	4,x		2,x	infinito
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					4,y

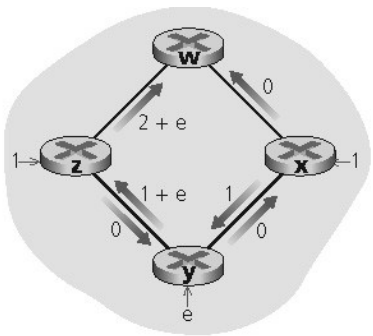


Discussão do algoritmo de Dijsktra

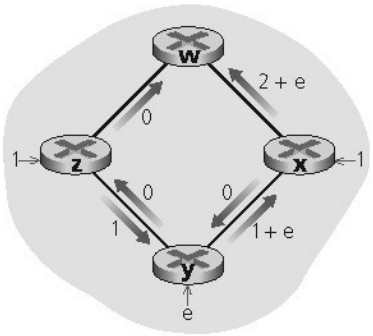
- Complexidade do algoritmo: **n** nós
  - Cada iteração precisa verificar todos os nós **w**, que não estão em **N**
  - **n(n+1)/2** comparações: **$O(n^2)$**
  - Implementações mais eficientes: **O(nlogn)**
- Variações possíveis:
  - Exemplo: custo do link = quantidade de tráfego transportado



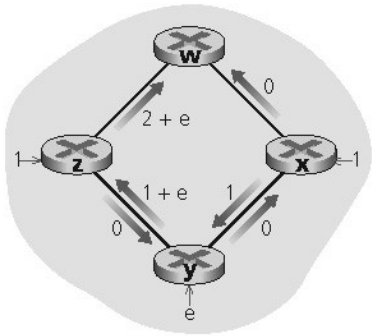
a. Roteamento inicial



b. *x*, *y* detectam melhor caminho até *w* em sentido horário



c. *x*, *y*, *z* detectam melhor caminho até *w* em sentido anti-horário



d. *x*, *y*, *z*, detectam melhor caminho até *w* em sentido horário

Roteiro

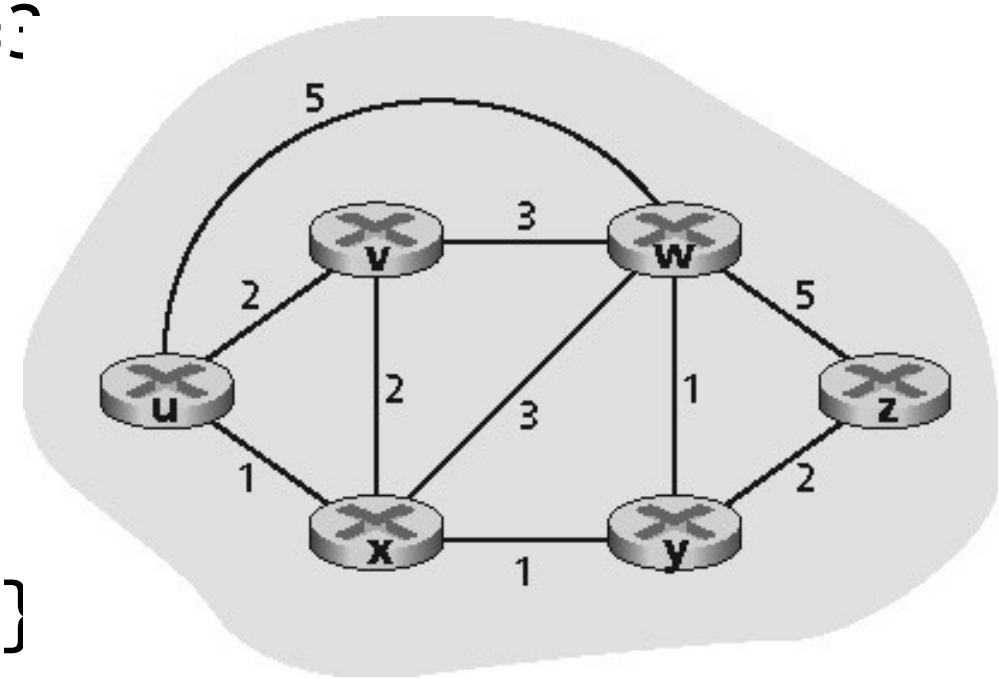
- Algoritmos de Roteamento Dinâmico
  - Link State
  - **Distance Vector**
  - Roteamento Hierárquico
- Protocolos de Roteamento na Internet

Algoritmo de Vetor Distância

- Utiliza Equação de Bellman-Ford (programação dinâmica)
- Considere
  - $d_x(y)$:= custo do caminho de menor custo de x para y
- Então
  - $d_x(y) = \min \{c(x,v) + dv(y) \}$
  - min é calculado sobre todos os vizinhos de x

Exemplo: Bellman-Ford

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



- Segundo equação B-F:

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

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

- O nó que **atinge o mínimo é o próximo salto** no caminho mais curto → tabela de roteamento

Algoritmo vetor de distância

- $D_x(y)$ = estima o menor custo de x para y
- Vetor de distância: $D_x = [D_x(y): y \in N]$
- O nó x conhece o custo para cada vizinho v: $c(x,v)$
- O nó x mantém $D_x = [D_x(y): y \in N]$
- O nó x também mantém os vetores de distância de seus vizinhos
- Para cada vizinho v, x mantém $D_v = [D_v(y): y \in N]$

Algoritmo vetor de distância

Idéia básica:

- Cada nó envia periodicamente sua própria estimativa de vetor de distância aos vizinhos
- Quando o nó x recebe nova estimativa de DV do vizinho, ele atualiza seu próprio DV usando a equação B-F:

  $D_x(y) = \min_v\{c(x,v) + D_v(y)\}$ para cada nó $y \in N$
- Ao menos em condições naturais, a estimativa $D_x(y)$ converge para o menor custo atual $d_x(y)$

Algoritmo vetor de distância

Iterativo, assíncrono: cada iteração local é causada por:

- Mudança no custo do enlace local
- Mensagem de atualização DV do vizinho

Distribuído:

- Cada nó notifica os vizinhos apenas quando seu DV mudar
- Os vizinhos então notificam seus vizinhos, se necessário

Passos em Cada nó

1) Espera por (mudança no custo do enlace local na mensagem do vizinho)
2) Recalcula estimativas de distâncias
3) Se o DV para qualquer destino mudou, notifica os vizinhos
4) Volta para (1)

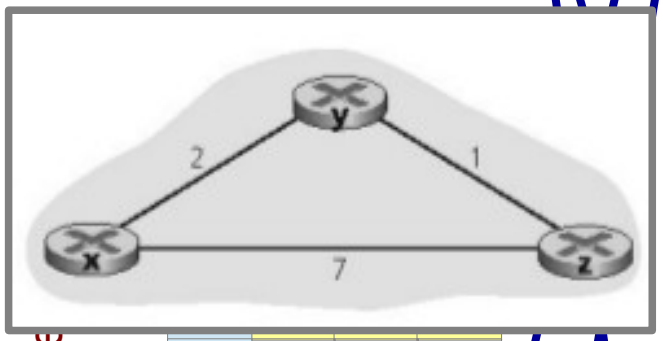
Exemplo

Tabela do nó **x**

De \ Custo até	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

De \ Custo até	x	y	z
x	0	2	**3**
y	2	0	1
z	7	1	0

De \ Custo até	x	y	z
x	0	2	**3**
y	2	0	1
z	3	1	0



De \ Custo até	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

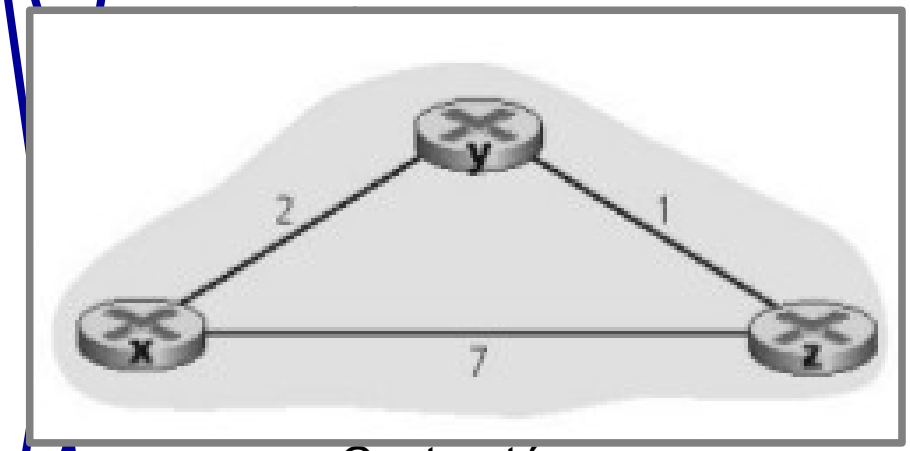


Tabela do nó **y**

De \ Custo até	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

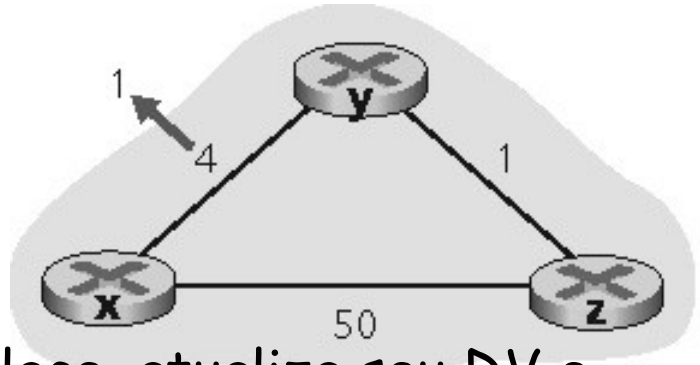
De \ Custo até	x	y	z
x	0	2	7
y	2	0	1
z	**3**	1	0

De \ Custo até	x	y	z
x	0	2	3
y	2	0	1
z	**3**	1	0

Mudanças no custo do enlace

Mudanças no custo do enlace:

- Nó detecta mudança no custo do enlace local
- Atualiza informações de roteamento, recalcula o vetor de distância
- Se o DV muda, notifica vizinhos



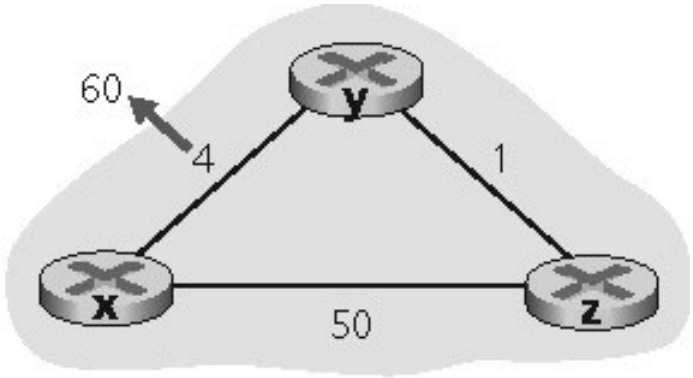
a.

- Em t_0 → y detecta a mudança no custo do enlace, atualiza seu DV e informa seus vizinhos.
- Em t_1 → z recebe a atualização de y e atualiza sua tabela.
  - Ele calcula o menor custo novo para x e envia seu DV para os vizinhos.
- Em t_2 → y recebe a atualização de z a atualiza sua tabela de distância. O menor custo de y's não muda e então y não envia nenhuma mensagem para z.

Mudanças no custo do enlace

Mudanças no custo do enlace:

- Boas notícias viajam rápido
- Más notícias viajam devagar - problema da "contagem ao infinito"!
- 44 iterações antes de o algoritmo estabilizar



b.

Reversão envenenada:

- Se Z roteia por Y para alcançar X :
- Z diz a Y que sua distância (de Z) para X é infinita (então Y não roteará até X via Z)
- Isso resolverá completamente o problema da contagem ao infinito?

Link State vs. Vector Distance

Complexidade

- **LS**: com n nós, E links, O(NE) mensagens enviadas
- **DV**: trocas somente entre vizinhos
  - Tempo de convergência varia

Tempo de convergência

- **LS**: algoritmo $O(N^2)$ exige mensagens O(NE)
  - Pode ter oscilações
- **DV**: tempo de convergência varia
  - Pode haver loops de roteamento
  - Problema da contagem ao infinito

Robustez: o que acontece se um roteador funciona mal?

- **Ls**:
  - Nós podem informar custos de link incorretos.
  - Cada nó calcula sua própria tabela de roteamento
- **Dv**:
  - Nó DV pode informar custo de caminho incorreto
  - Tabela de cada nó é usada por outros
  - Propagação de erros pela rede

Roteiro

- Algoritmos de Roteamento Dinâmico
  - Link State
  - Distance Vector
  - **Roteamento Hierárquico**
- Protocolos de Roteamento na Internet

Roteamento hierárquico

Nosso estudo é uma idealização

- Roteadores são todos idênticos
- Redes "flat"
- ... na prática, isso não é verdade

Escala: com 200 milhões de destinos:

- Não é possível armazenar todos os destinos numa única tabela de rotas!
- As mudanças na tabela de rotas irão congestionar os enlaces!

Autonomia administrativa

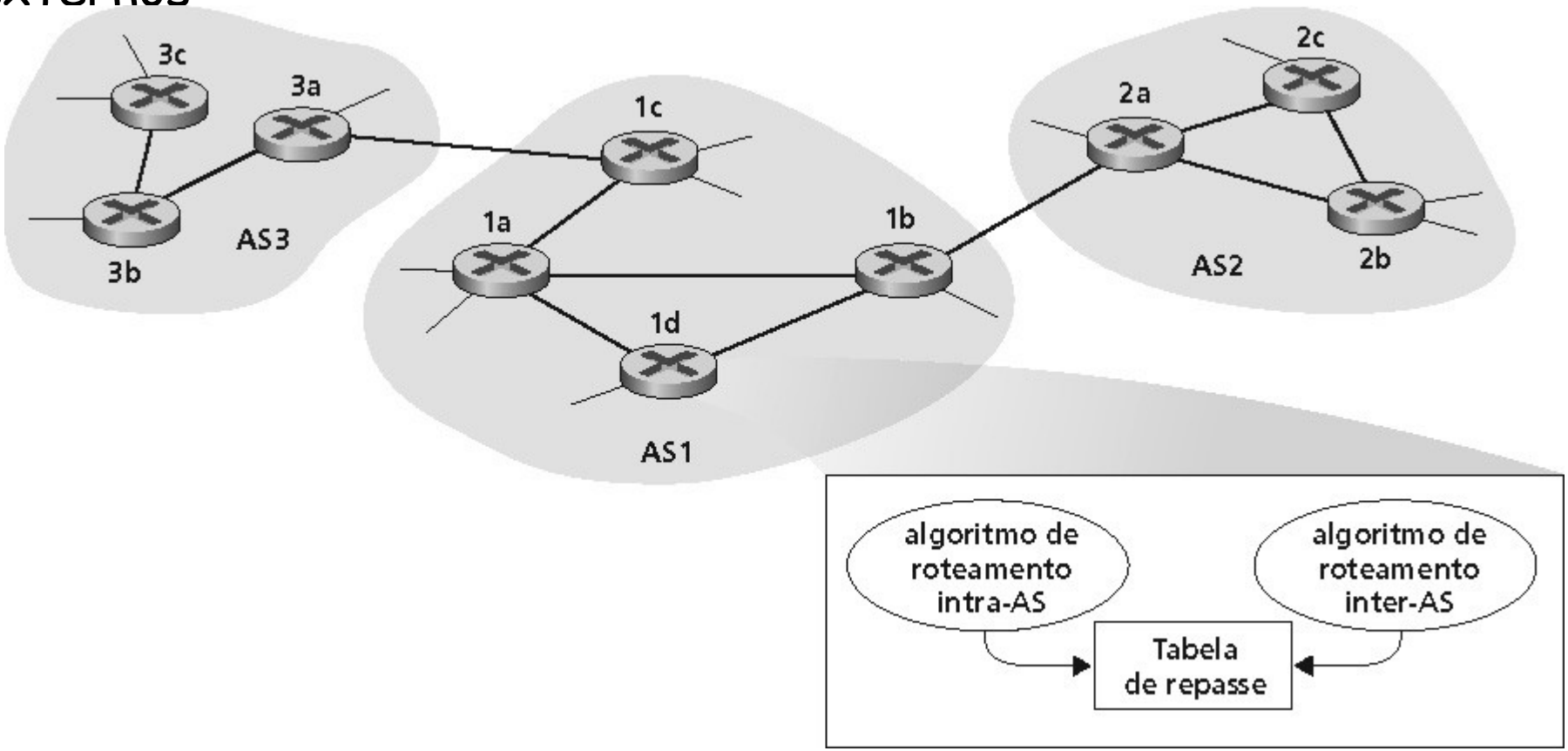
- Internet = rede de redes
- Cada administração de rede pode querer controlar o roteamento na sua própria rede

Roteamento hierárquico

- Agrega roteadores em regiões → **sistemas autônomos** (***AS***)
- Roteadores no mesmo AS rodam o mesmo protocolo de roteamento
  - Protocolo de roteamento **intra-AS**
  - Roteadores em diferentes AS podem rodar diferentes protocolos de roteamento
- Roteador Gateway
  - Link direto para um roteador em outro AS

ASs interconectadas

- Tabela de roteamento é configurada por ambos algoritmos, intra- e inter-AS
- Intra-AS estabelece entradas para destinos internos
- Inter-AS e intra-As estabelecem entradas para destinos externos



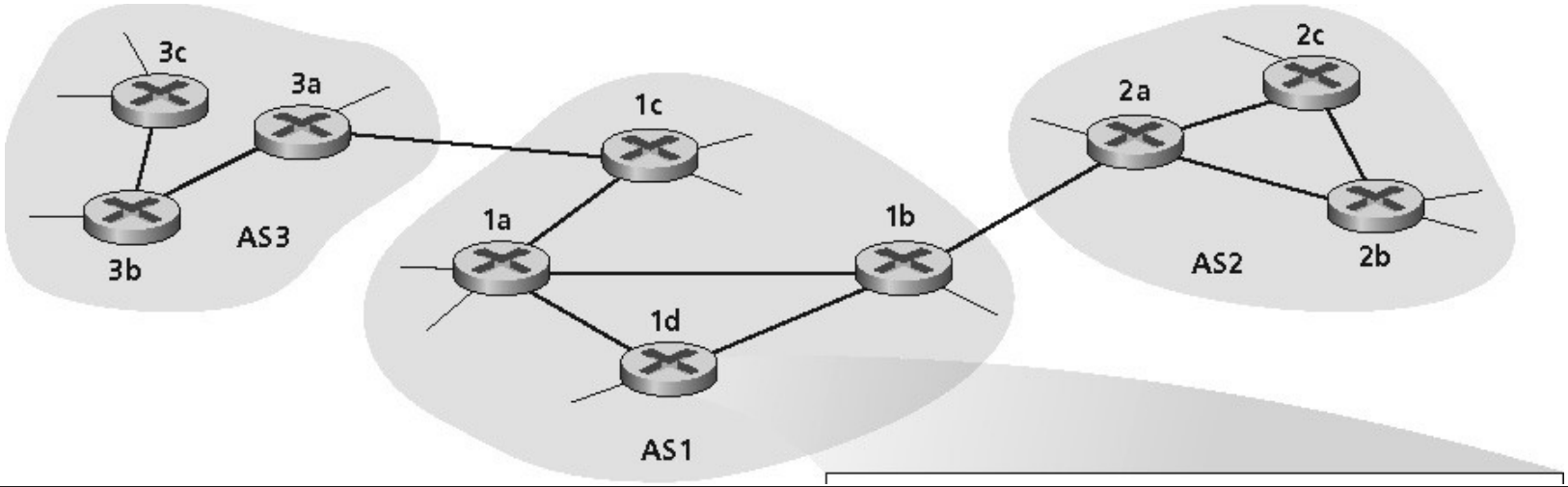
Tarefas Inter-AS

- Suponha que um roteador no AS1 receba um datagrama cujo destino seja fora do AS1
  - O roteador deveria encaminhar o pacote para os roteadores gateway, mas qual deles?

AS1 precisa:

1. Aprender quais destinos são alcancáveis através de AS2 e através de AS3.
2. Propagar suas informações de alcance para todos os roteadores em AS1.

- Tarefa para o roteamento inter-AS routing!

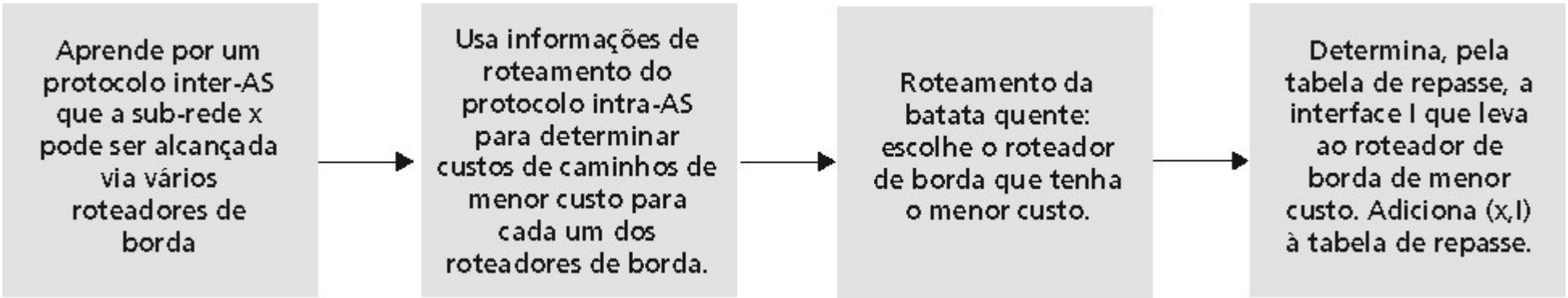


Exemplo: Ajuste em tabela

- Exemplo: Ajustando a tabela de roteamento no roteador **1d**
  - Suponha que AS1 aprende pelo protocolo inter-AS protocol que a sub-rede **x** é alcancável através de AS3 (gateway **1c**) mas não através de AS2.
  - O protocolo inter-AS propaga informações de alcance para todos os roteadores internos.
  - Baseado nas informações de roteamento intra-AS, o roteador **1d** determina que sua interface I está no caminho de menor custo para **1c.**
  - Coloca na tabela de roteamento a entrada **(x,I)**.

Exemplo: Múltiplas ASs

- Exemplo: Escolhendo entre múltiplas ASs
  - AS1 aprende pelo protocolo inter-AS que a sub-rede x é alcançavel através de AS3 e através de AS2.
  - Para configurar a tabela de roteamento, o roteador 1d deve determinar por qual gateway ele deve encaminhar os pacotes para o destino x.
  - Isso também é tarefa para o protocolo de roteamento inter-AS.



Roteiro

- Algoritmos de Roteamento Dinâmico
  - Link State
  - Distance Vector
  - Roteamento Hierárquico
- Protocolos de Roteamento na Internet

Roteiro

- Algoritmos de Roteamento Dinâmico
  - Link State
  - Distance Vector
  - Roteamento Hierárquico
- Protocolos de Roteamento na Internet

Roteiro

- Algoritmos de Roteamento Dinâmico
  - Link State
  - Distance Vector
  - Roteamento Hierárquico
- Protocolos de Roteamento na Internet
  - **RIP**
  - OSPF
  - BGP

Roteiro

- Algoritmos de Roteamento Dinâmico
  - Link State
  - Distance Vector
  - Roteamento Hierárquico
- Protocolos de Roteamento na Internet
  - RIP
  - **OSPF**
  - BGP

Roteiro

- Algoritmos de Roteamento Dinâmico
  - Link State
  - Distance Vector
  - Roteamento Hierárquico
- Protocolos de Roteamento na Internet
  - RIP
  - OSPF
  - **BGP**

Resumo e Conceitos-Chave

Referências

- Iraj Sodagar, *"The MPEG-DASH Standard for Multimedia Streaming Over the Internet,"* **IEEE Multimedia**, vol. 18, no. 4, pp. 62-67, October-December, 2011.