

Capítulo 5

Camada de rede: Plano de controle

Uma observação sobre o uso desses slides do PowerPoint:

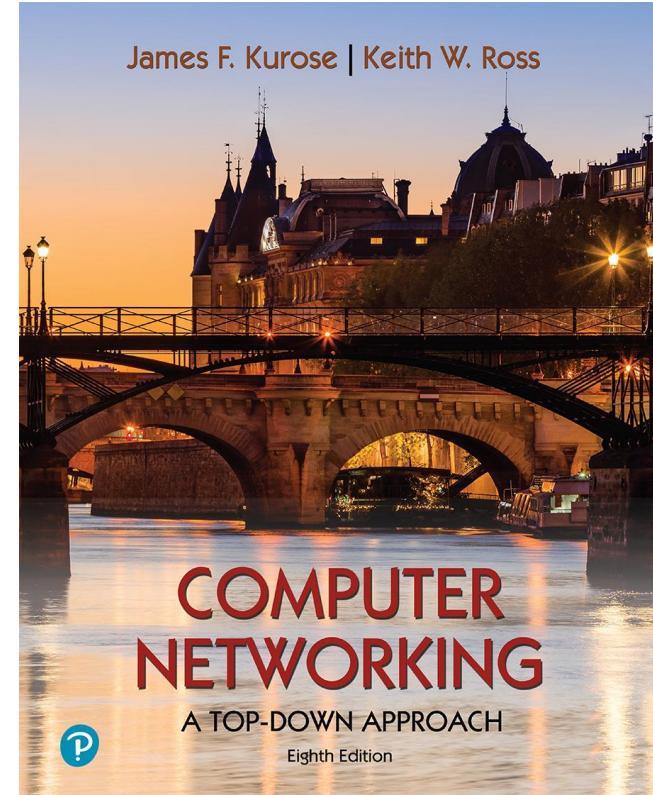
Estamos disponibilizando esses slides gratuitamente para todos (professores, alunos, leitores). Eles estão no formato PowerPoint para que você veja as animações e possa adicionar, modificar e excluir slides (inclusive este) e o conteúdo dos slides para atender às suas necessidades. Obviamente, eles representam *muito* trabalho de nossa parte. Em troca do uso, pedimos apenas o seguinte:

- Se você usar esses slides (por exemplo, em uma aula), mencione a fonte (afinal, gostaríamos que as pessoas usassem nosso livro!)
- Se você publicar algum slide em um site www, informe que ele foi adaptado de nossos slides (ou talvez idêntico a eles) e informe nossos direitos autorais sobre esse material.

Para obter um histórico de revisões, consulte a nota do slide desta página.

Obrigado e divirta-se! JFK/KWR

Todos os materiais têm direitos autorais de 1996 a 2023
J.F. Kurose e K.W. Ross, Todos os direitos reservados



*Redes de computadores: A
Top-Down Approach (Uma
abordagem de cima para
baixo)*

8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Plano de controle da camada de rede: nossos objetivos

- compreender os princípios por trás do plano de controle da rede:
 - algoritmos de roteamento tradicionais
 - Controladores SDN
 - gerenciamento de rede, configuração
- instanciação, implementação na Internet:
 - OSPF, BGP
 - Controladores OpenFlow, ODL e ONOS
 - Protocolo de Mensagens de Controle da Internet: ICMP
 - SNMP, YANG/NETCONF

Camada de rede: Roteiro do "plano de controle"

- **introdução**
- protocolos de roteamento
 - estado do link
 - vetor de distância
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- Plano de controle SDN
- Protocolo de Mensagens de Controle da Internet



- gerenciamento de rede, configuração
 - SNMP
 - NETCONF/YANG

Funções da camada de rede

- **encaminhamento:** mover os pacotes da entrada do roteador para a saída apropriada do roteador
 - **Roteamento:** determina a rota seguida pelos pacotes da origem ao destino

plano de dados

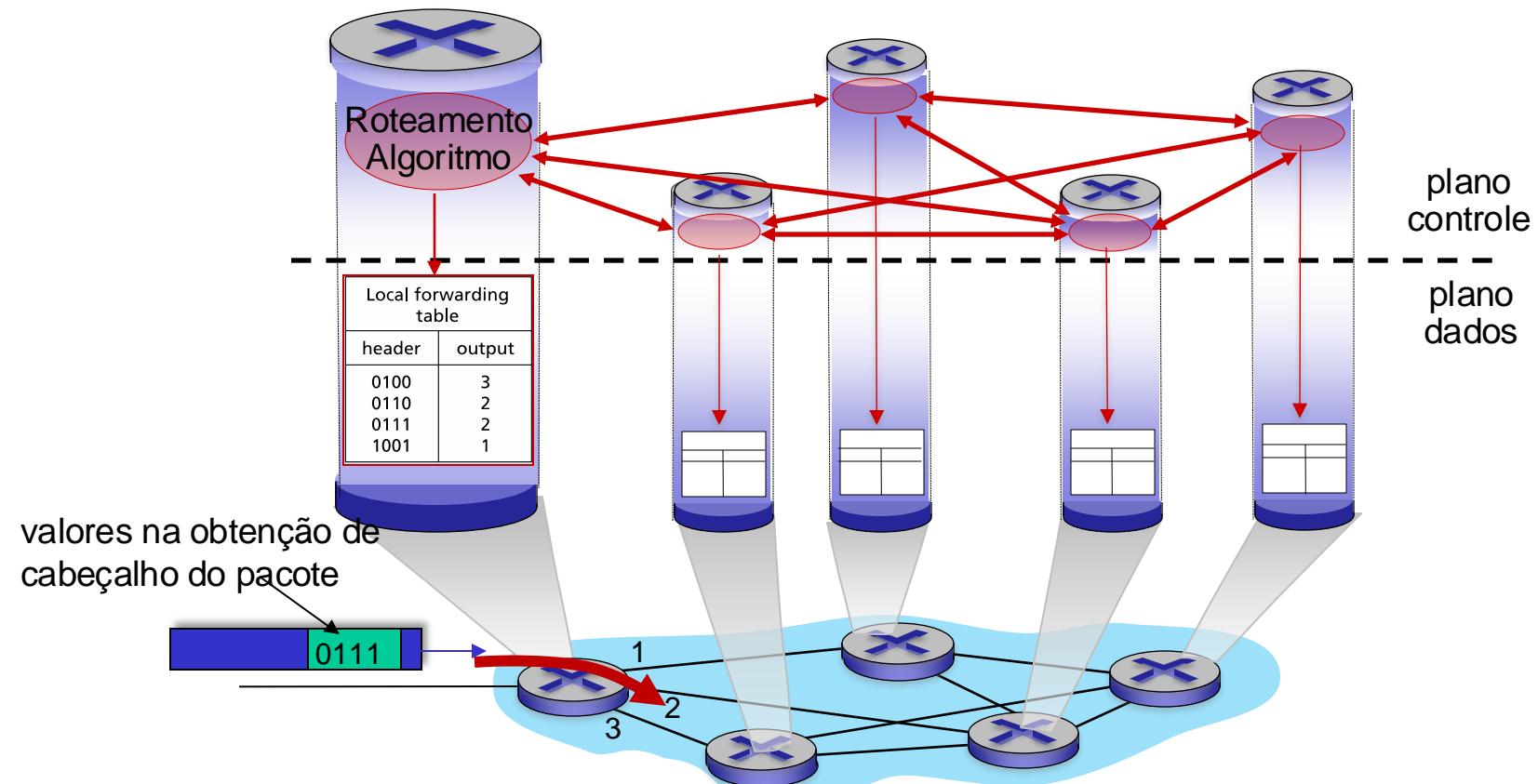
plano de controle

Duas abordagens para estruturar o plano de controle da rede:

- controle por roteador (tradicional)
- controle logicamente centralizado (rede definida por software)

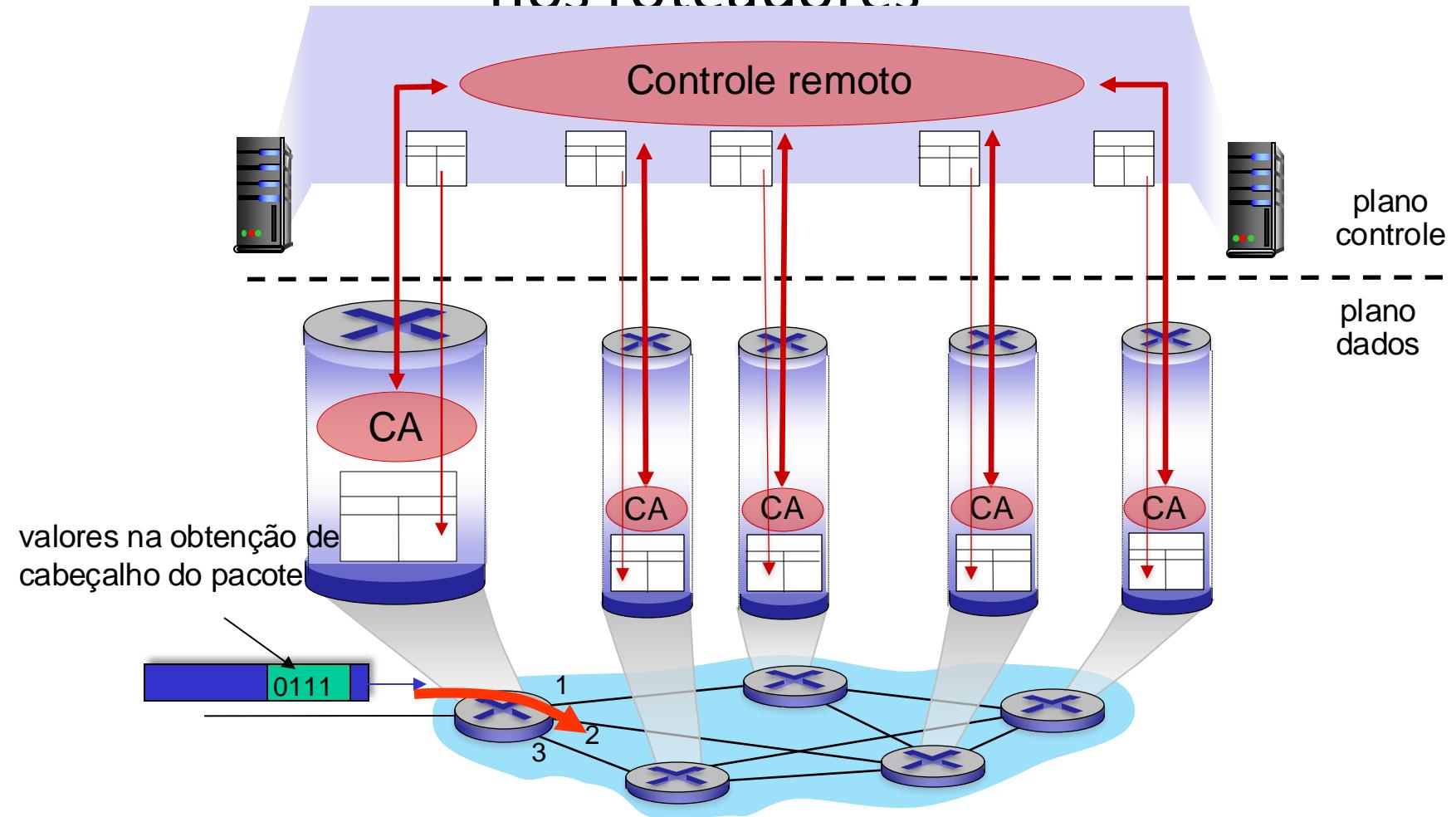
Plano de controle por roteador

Os componentes individuais do algoritmo de roteamento *em cada roteador* interagem no plano de controle

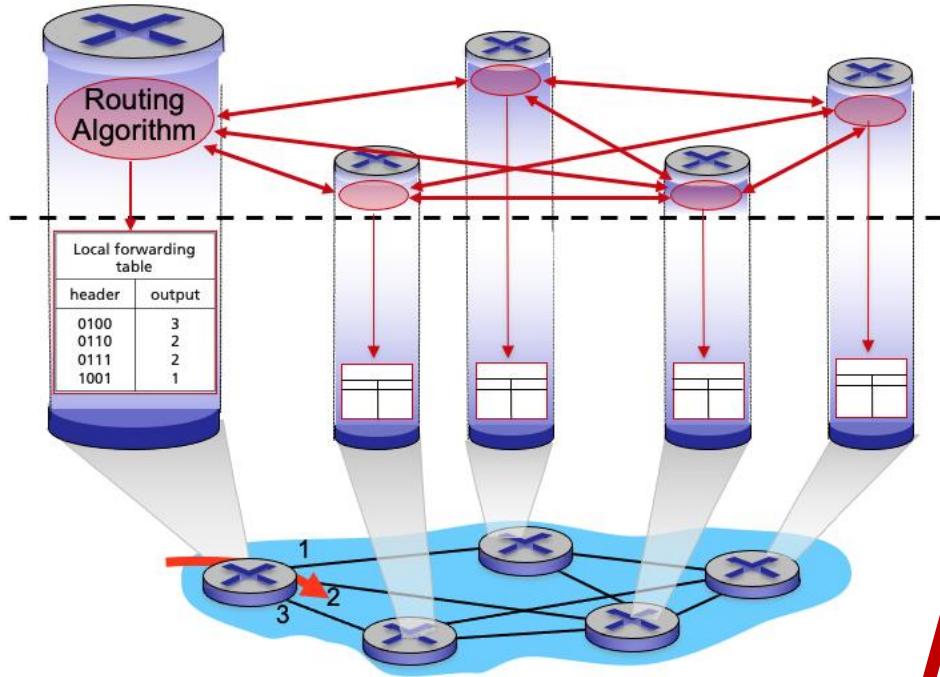


Plano de controle de SDN (Software-Defined Networking)

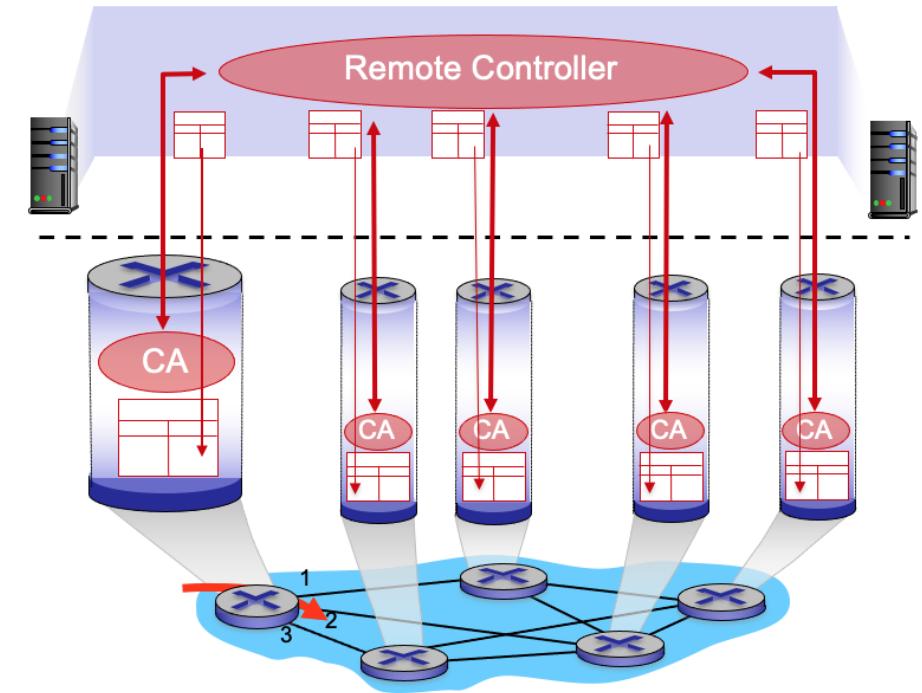
O controlador remoto calcula e instala tabelas de encaminhamento nos roteadores



Plano de controle por roteador



Plano de controle SDN



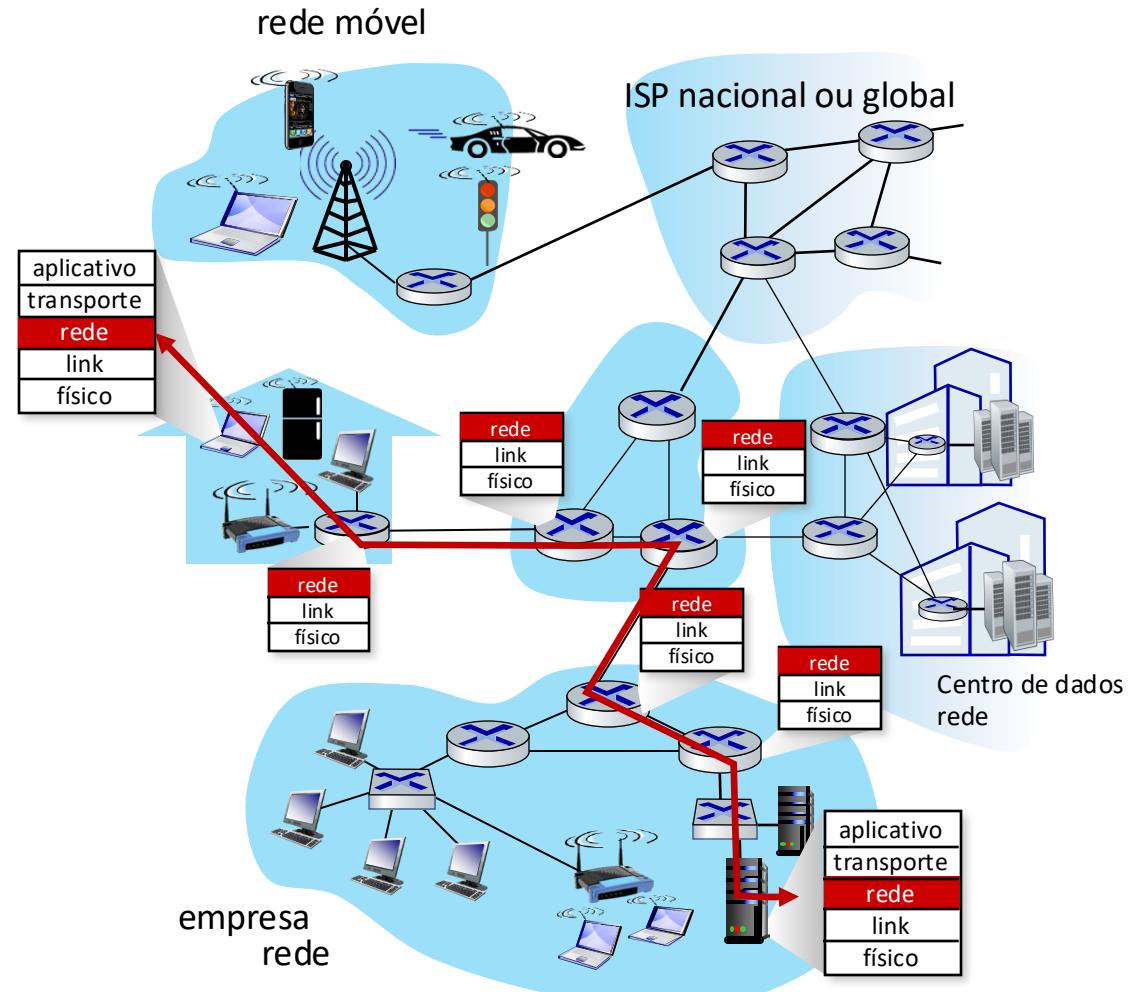
Camada de rede: Roteiro do "plano de controle"

- introdução
 - protocolos de roteamento
 - estado do link
 - vetor de distância
 - roteamento intra-ISP: OSPF
 - roteamento entre ISPs: BGP
 - Plano de controle SDN
 - Protocolo de Mensagens de Controle da Internet
- 
- gerenciamento de rede, configuração
 - SNMP
 - NETCONF/YANG

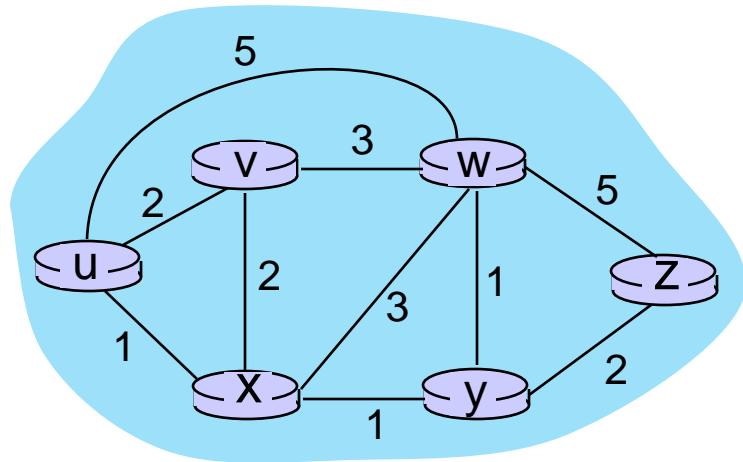
Protocolos de roteamento

Objetivo do protocolo de roteamento: determinar caminhos "bons" (ou seja, rotas), de hosts de envio a hosts de recebimento, por meio de uma rede de roteadores

- **caminho:** sequência de roteadores que os pacotes percorrem de um determinado host de origem inicial até o host de destino final
- **"bom":** menor "custo", "mais rápido", "menos congestionado"
- roteamento: um desafio de rede "top 10"!



Abstração de gráficos: custos de links



$c_{a,b}$: custo do link *direto* que conecta a e b

Por exemplo, $c_{w,z} = 5$, $c_{u,z} = \infty$

custo definido pela operadora de rede: pode ser sempre 1, ou inversamente relacionado à largura de banda, ou inversamente relacionado ao congestionamento

gráfico: $G = (N, E)$

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

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

Classificação do algoritmo de roteamento

Com que rapidez as rotas mudam?

estático: as rotas mudam lentamente ao longo do tempo

global: todos os roteadores têm topologia *completa*, informações de custo de link

- "Algoritmos de "estado do link

dinâmico: as rotas mudam mais rapidamente

- atualizações periódicas ou em resposta a alterações nos custos do link

descentralizado: processo iterativo de computação, troca de informações com os vizinhos

- os roteadores inicialmente só conhecem os custos de link dos vizinhos conectados
- "Algoritmos de "vetor de distância

informações globais ou descentralizadas?

Camada de rede: Roteiro do "plano de controle"

- introdução
- protocolos de roteamento
 - estado do link
 - vetor de distância
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- Plano de controle SDN
- Protocolo de Mensagens de Controle da Internet



- gerenciamento de rede, configuração
 - SNMP
 - NETCONF/YANG

Algoritmo de roteamento link-state de Dijkstra

- **centralizado:** topologia da rede, custos de link conhecidos por *todos* os nós
 - realizado por meio da "transmissão do estado do link"
 - todos os nós têm as mesmas informações
- calcula os caminhos de menor custo de um nó ("fonte") para todos os outros nós
 - fornece *a tabela de encaminhamento* para esse nó
- **iterativo:** após k iterações, conheça o caminho de menor custo para k destinos

notação

- $c_{x,y}$: custo do link direto do nó x para y ; $= \infty$ se não houver vizinhos diretos
- $D(v)$: estimativa *atual* do custo do caminho de menor custo da origem ao destino v
- $p(v)$: nó predecessor ao longo do caminho da fonte até v
- N' : conjunto de nós cujo caminho de menor custo é *definitivamente* conhecido

Algoritmo de roteamento link-state de Dijkstra

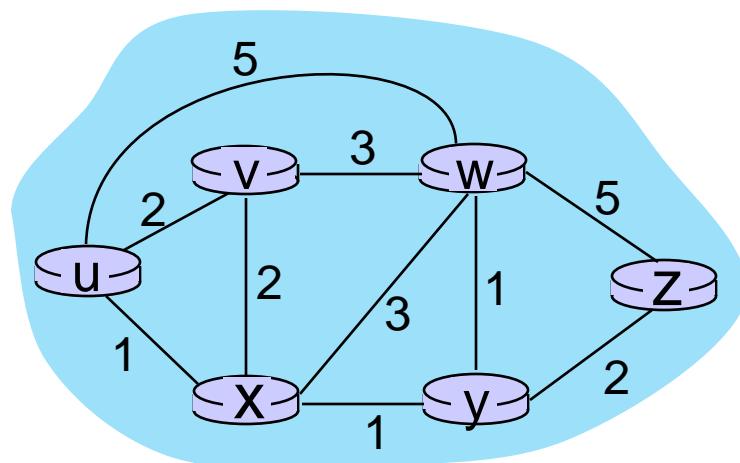
```
1 Inicialização:
2  $N' = \{u\}$  /* calcular o caminho de menor custo de  $u$  para todos os outros nós */
3 para todos os nós  $v$ 
4 se  $v$  for adjacente a  $u$  /*  $u$  inicialmente conhece o direct-path-cost apenas dos vizinhos diretos */
5 then  $D(v) = c_{u,v}$  /* mas pode não ser o custo mínimo! */ 
6 senão  $D(v) = \infty$ 
7
```

8 Loop

```
9     encontrar  $w$  que não esteja em  $N'$  de modo que  $D(w)$  seja um mínimo
10    adicionar  $w$  a  $N'$ 
11    atualizar  $D(v)$  para todos os  $v$  adjacentes a  $w$  e que não estejam em  $N'$  :
12         $D(v) = \min(D(v), D(w) + c_{w,v})$ 
13    /* o novo caminho de menor custo para  $v$  é o antigo caminho de menor custo para  $v$  ou é conhecido
14    caminho de menor custo para  $w$  mais custo direto de  $w$  para  $v$  */
15 até que todos os nós em  $N'$ 
```

Algoritmo de Dijkstra: um exemplo

Etapa	N'	V	W	X	Y	Z	
		D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)	
0	u	2,u	5,u	1,u	∞	∞	D(w),p(w)
1							5,u
2							4,x
3							3,y
4							3,y
5							



Inicialização (etapa 0):

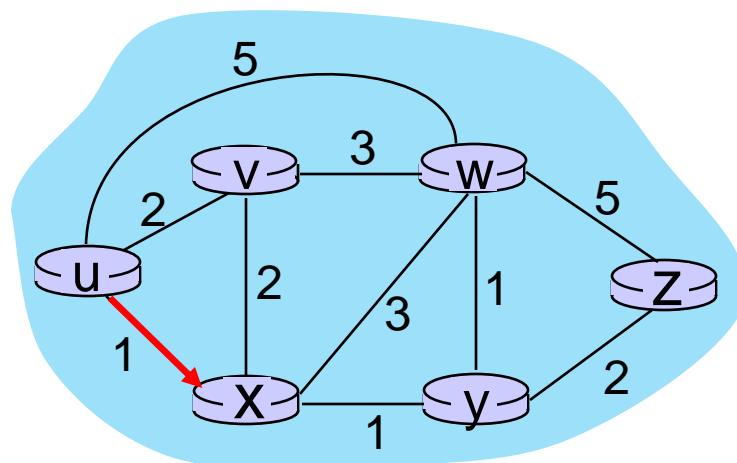
Para todo a : se a for adjacente a u , então
 $D(a) = c_{u,a}$

Algoritmo de Dijkstra: um exemplo

Etapa	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$	
0	u	2,u	5,u	1,u	∞	∞	$D(w), p(w)$
1	ux						5,u
2							4,x
3							3,y
4							3,y
5							

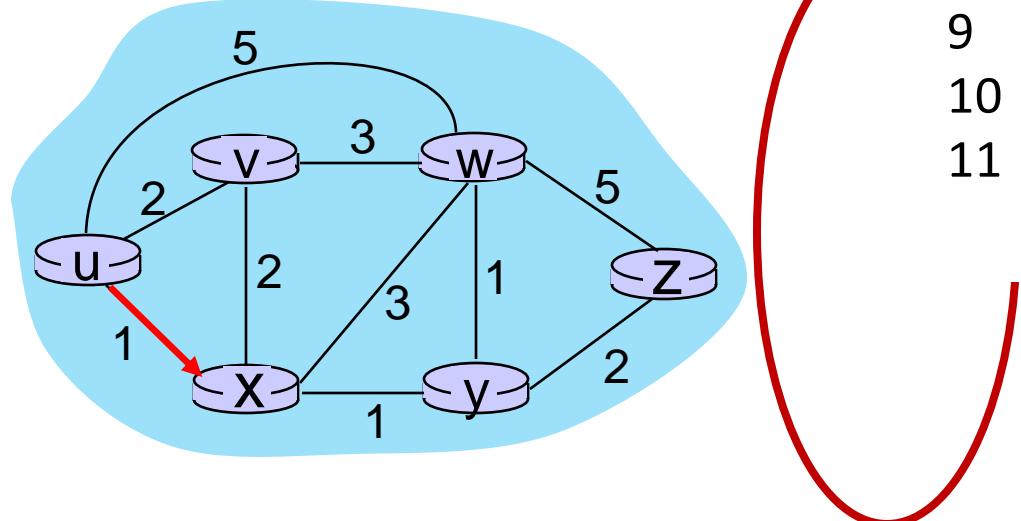
8 Loop

- 9 encontrar um não em N' de modo que $D(a)$ seja um
mínimo
10 adicionar a a N'



Algoritmo de Dijkstra: um exemplo

Etapa	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$	$D(w), p(w)$
0	u	2,u	5,u	1,u	∞	∞	5,u
1	ux	2,u	4,x		2,x	∞	4,x
2							3,y
3							3,y
4							
5							



8 Loop

encontrar a não em N' de modo que $D(a)$ seja um mínimo
adicionar a a N'
atualizar $D(b)$ para todos os b adjacentes a a e que não estejam em N'

$$D(b) = \min(D(b), D(a) + c)_{a,b}$$

$$D(v) = \min(D(v), D(x) + c_{x,v}) = \min(2, 1+2) = 2$$

$$D(w) = \min(D(w), D(x) + c_{x,w}) = \min(5, 1+3) = 4$$

$$D(y) = \min(D(y), D(x) + c_{x,y}) = \min(\infty, 1+1) = \infty$$

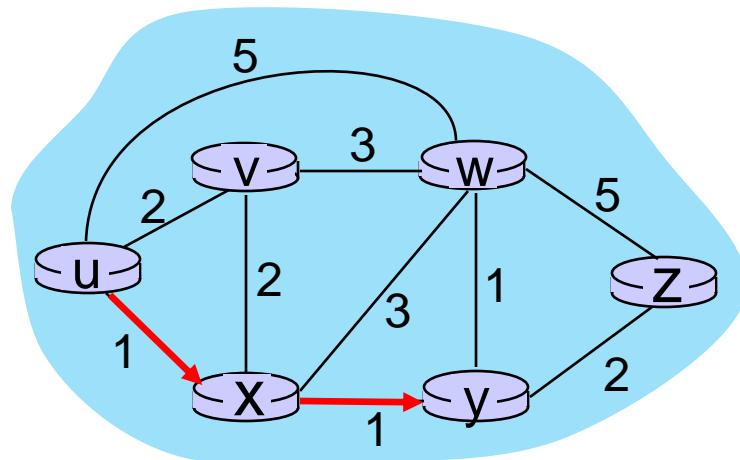


Algoritmo de Dijkstra: um exemplo

Etapa	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$	
0	u	2,u	5,u	1,u	∞	∞	$D(w), p(w)$
1	ux	2,u	4,x		2,x	∞	$5,u$
2	uxy						$4,x$
3							$3,y$
4							$3,y$
5							

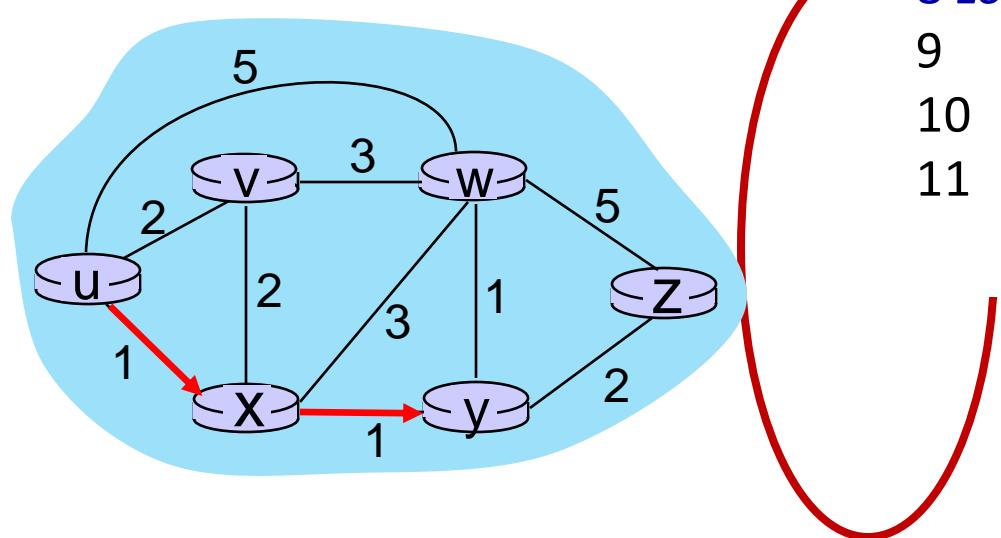
8 Loop

- 9 encontrar u não em N' de modo que $D(u)$ seja um mínimo
- 10 adicionar u a N'



Algoritmo de Dijkstra: um exemplo

Etapa	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3						
4						
5						



8 Loop

encontrar u não em N' de modo que $D(u)$ seja um mínimo
adicionar u a N'

atualizar $D(b)$ para todos os b adjacentes a u e que não estejam em N' :

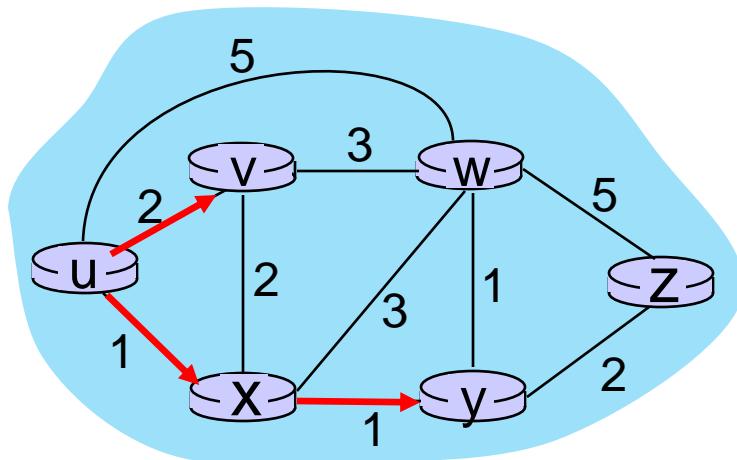
$$D(b) = \min(D(b), D(u) + c_{u,b})$$

$D(w) = \min(D(w), D(y) + c_{y,w}) = \min(4, 2+1) = 3$ NEW!

$D(z) = \min(D(z), D(y) + c_{y,z}) = \min(\infty, 2+2) = 4$ NEW!

Algoritmo de Dijkstra: um exemplo

Etapa	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv					
4						
5						

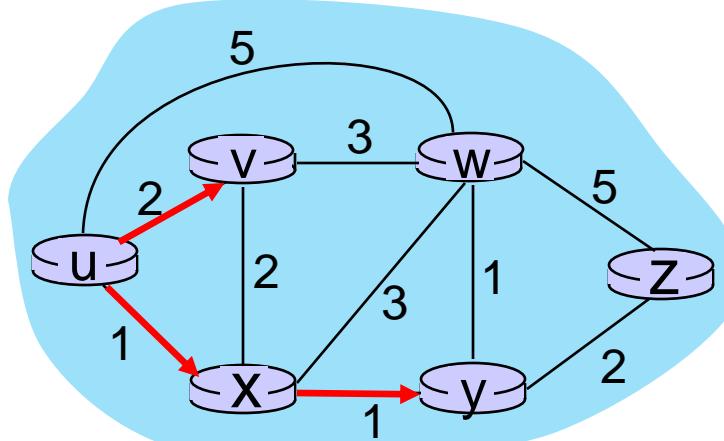


8 Loop

- 9 encontrar um não em N' de modo que $D(a)$ seja um mínimo
- 10 adicionar a a N'

Algoritmo de Dijkstra: um exemplo

Etapa	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyyv		3,y			4,y
4						
5						



8 Loop

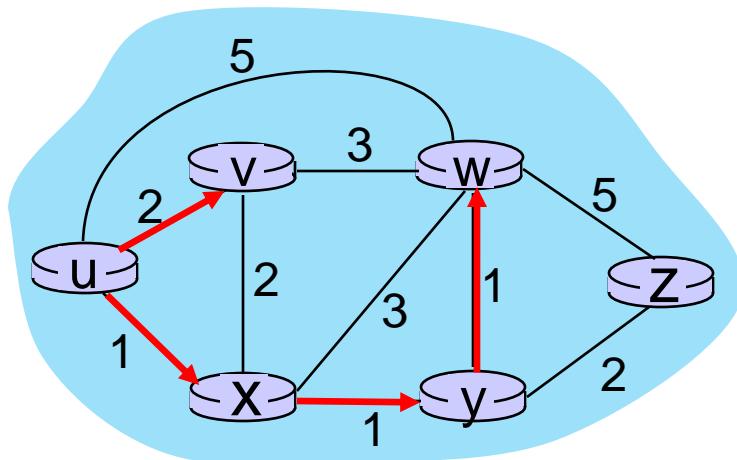
- encontrar u não em N' de modo que $D(a)$ seja um mínimo
adicionar a a N'
atualizar $D(b)$ para todos os b adjacentes a a e que não estejam em N' :

$$D(b) = \min (D(b), D(a) + c) ,$$

$$D(w) = \min (D(w), D(v) + c_{v,w}) = \min (3, 2+3) = 3$$

Algoritmo de Dijkstra: um exemplo

Etapa	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyyv		3,y			4,y
4	uxyvw					
5						

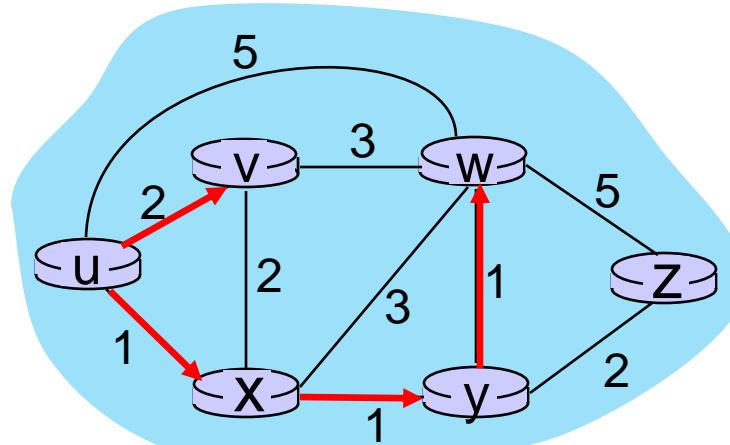


8 Loop

- 9 encontrar a em N' de modo que $D(a)$ seja um mínimo
- 10 adicionar a a N'

Algoritmo de Dijkstra: um exemplo

Etapa	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyyv		3,y			4,y
4	uxyvw					4,y
5						



8 Loop

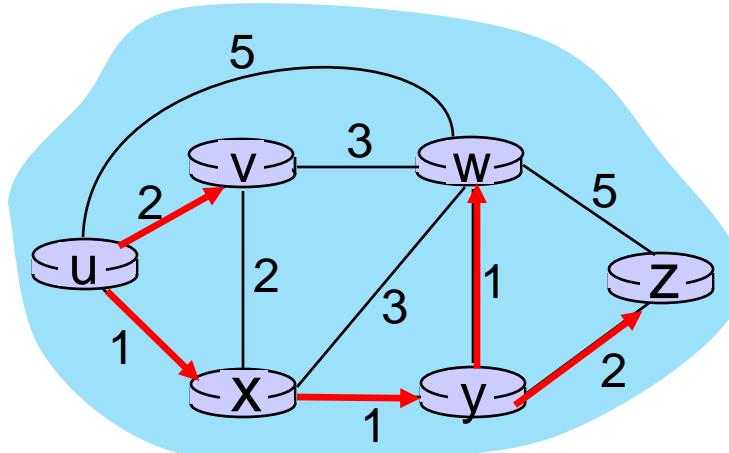
- 9 encontrar a não em N' de modo que $D(a)$ seja um mínimo
- 10 adicionar a a N'
- 11 atualizar $D(b)$ para todos os b adjacentes a a e que não estejam em N'
- :

$$D(b) = \min (D(b), D(a) + c)_{a \sim b}$$

$$D(z) = \min (D(z), D(w) + c_{w,z}) = \min (4, 3+5) = 4$$

Algoritmo de Dijkstra: um exemplo

Etapa	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

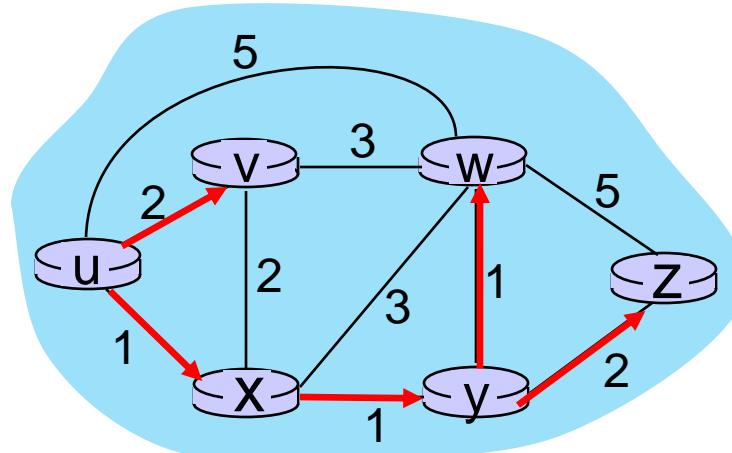


8 Loop

- 9 encontrar um não em N' de modo que $D(a)$ seja um mínimo
- 10 adicionar a a N'

Algoritmo de Dijkstra: um exemplo

Etapa	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

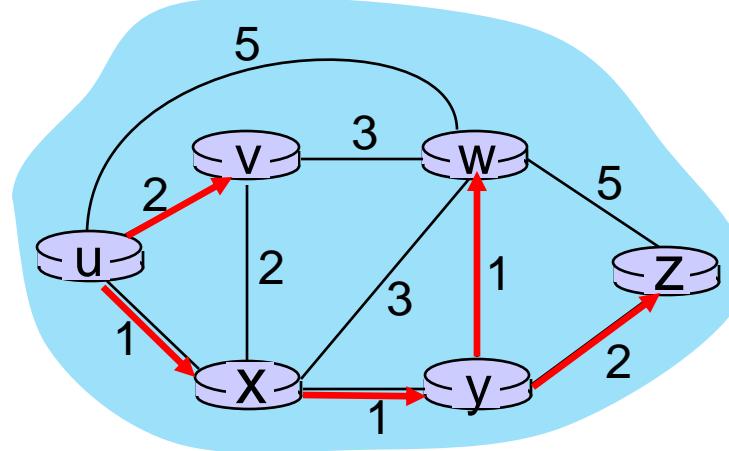


8 Loop

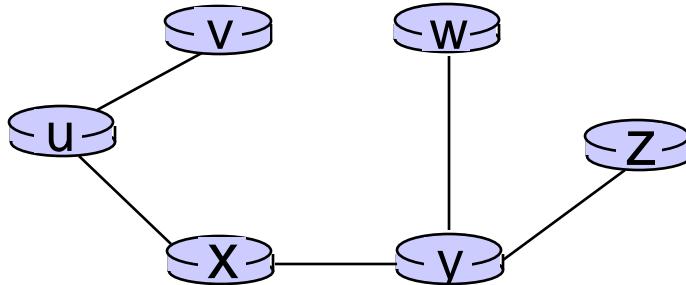
- 9 encontrar um não em N' de modo que $D(a)$ seja um mínimo
- 10 adicionar a a N'
- 11 atualizar $D(b)$ para todos os b adjacentes a a e que não estejam em N' :

$$D(b) = \min (D(b), D(a) + c)_{a,b}$$

Algoritmo de Dijkstra: um exemplo



árvore de caminho de menor custo resultante e a tabela de encaminhamento resultante em u:

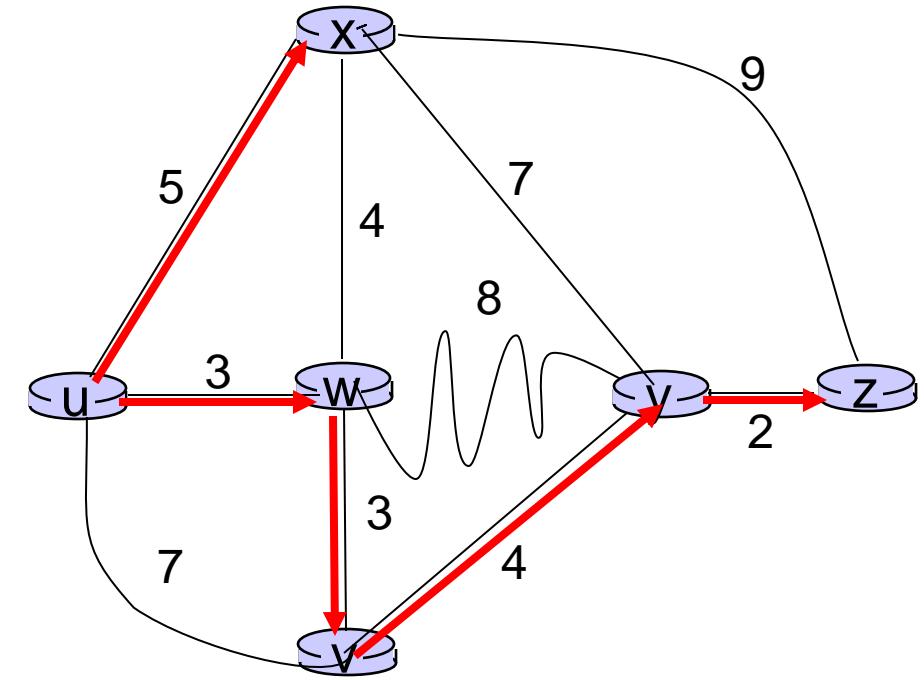


destino	link de saída
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
x	(u,x)

rota de u para v diretamente
rota de u para todos os outros destinos via x

Algoritmo de Dijkstra: outro exemplo

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



notas:

- construir uma árvore de caminho de menor custo rastreando os nós predecessores
- podem existir laços (podem ser quebrados arbitrariamente)

Algoritmo de Dijkstra: discussão

complexidade do algoritmo: n nós

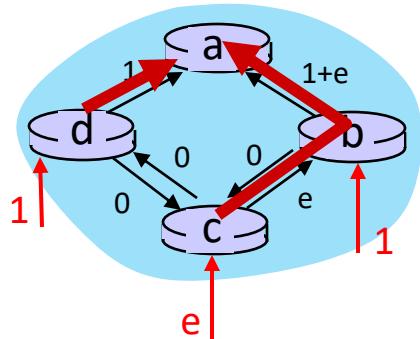
- cada uma das n iterações: é necessário verificar todos os nós, w , que não estão em N
- $n(n+1)/2$ comparações: Complexidade $O(n^2)$
- é possível fazer implementações mais eficientes: $O(n \log n)$

complexidade da mensagem:

- cada roteador deve *transmitir* suas informações de estado do link para outros n roteadores
- algoritmos de transmissão eficientes (e interessantes!): $O(n)$ cruzamentos de links para disseminar uma mensagem de transmissão de uma fonte
- a mensagem de cada roteador atravessa $O(n)$ links: complexidade geral da mensagem: $O(n^2)$

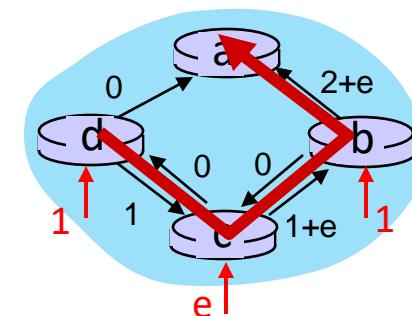
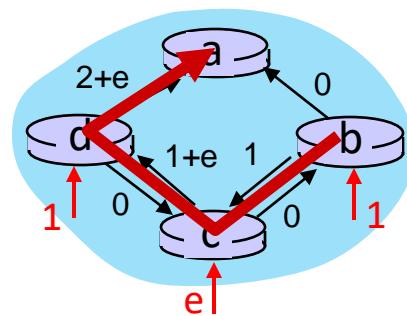
Algoritmo de Dijkstra: oscilações possíveis

- quando os custos do link dependem do volume de tráfego, é possível haver
- **oscilações de rota**
- cenário de exemplo:
 - roteamento para o destino a, tráfego entrando em d, c, e com taxas 1, e (<1), 1
 - os custos do link são direcionais e dependem do volume

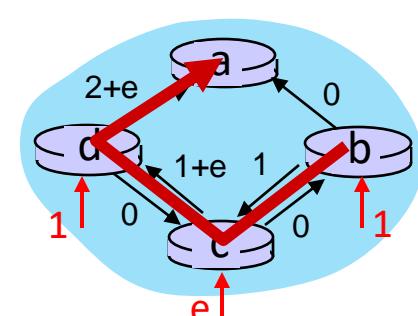


inicialmente

considerando esses custos,
encontrar novas rotas....
resultando em novos custos



considerando esses custos,
encontrar novas rotas....
resultando em novos custos



considerando esses custos,
encontrar novas rotas....
resultando em novos custos

Camada de rede: Roteiro do "plano de controle"

- introdução
- protocolos de roteamento
 - estado do link
 - **vetor de distância**
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- Plano de controle SDN
- Protocolo de Mensagens de Controle da Internet



- gerenciamento de rede, configuração
 - SNMP
 - NETCONF/YANG

Algoritmo de vetor de distância

Baseado na equação *de Bellman-Ford* (BF) (programação dinâmica):

Equação de Bellman-Ford

Seja $D_x(y)$: custo do caminho de menor custo de x para y .

Então:

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

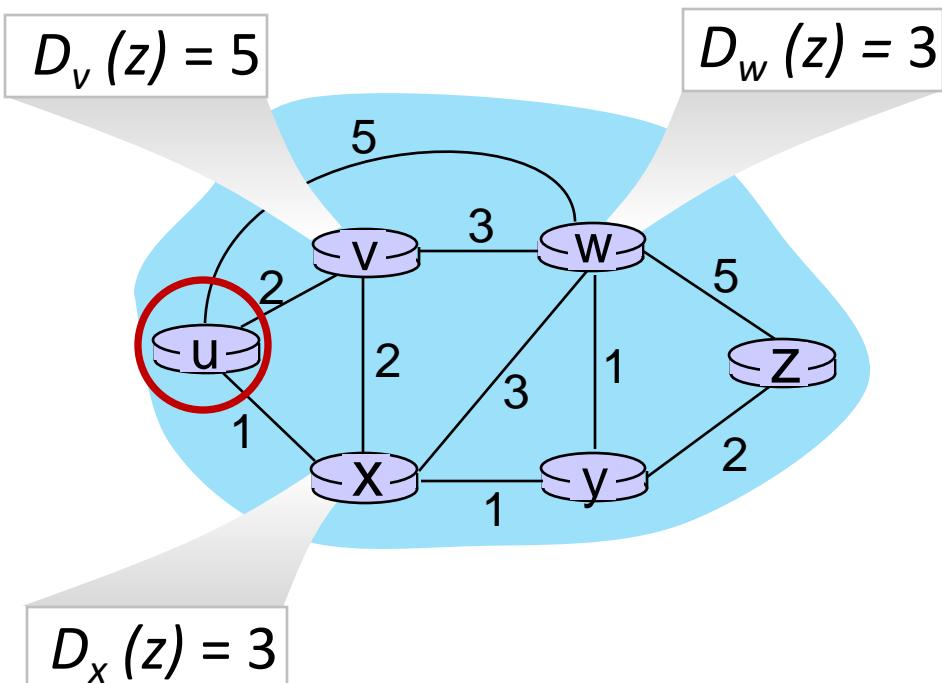
mínimo obtido em todos os vizinhos v de x

O custo estimado do caminho de menor custo de v para y

custo direto do link de x para v

Exemplo de Bellman-Ford

Suponha que os nós vizinhos de u, x, v, w , saibam que para o destino z :



A equação de Bellman-Ford diz:

$$\begin{aligned} D_u(z) &= \min \{ c_{u,v} + D_v(z), \\ &\quad c_{u,x} + D_x(z), \\ &\quad c_{u,w} + D_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

o nó que atinge o mínimo (x) é o próximo salto no caminho de menor custo estimado para o destino (z)

Algoritmo de vetor de distância

ideia-chave:

- De tempos em tempos, cada nó envia sua própria estimativa do vetor de distância para os vizinhos
- Quando x recebe uma nova estimativa de DV de qualquer vizinho, ele atualiza seu próprio DV usando a equação B-F:

$$D_x(y) \leftarrow \min \{c_{vx,v} + D_v(y)\} \text{ para cada nó } y \in N$$

- Em condições naturais menores, a estimativa $D_x(y)$ converge para o custo mínimo real $d_x(y)$

Algoritmo de vetor de distância:

cada nó:

aguardar (alteração no custo do link local ou mensagem do vizinho)

recompute as estimativas de DV usando o DV recebido do vizinho

se o DV para qualquer destino tiver mudado, *notifique* os vizinhos

iterativo, assíncrono: cada iteração local causada por:

- alteração do custo do link local
- Mensagem de atualização DV do vizinho

distribuído, com parada automática:

cada nó notifica os vizinhos *somente* quando seu DV é alterado

- os vizinhos notificam seus vizinhos - *somente se necessário*
- nenhuma notificação recebida, nenhuma ação tomada!

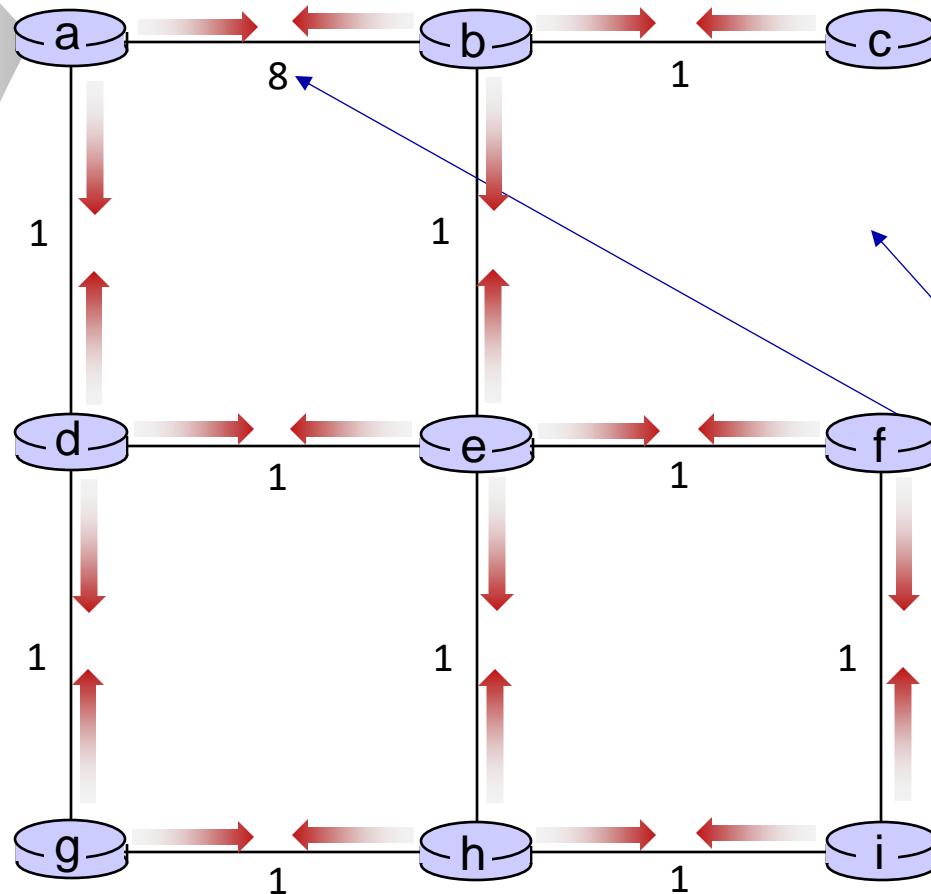
Vetor de distância: exemplo



$t=0$

- Todos os nós têm estimativas de distância para os vizinhos mais próximos (apenas)
- Todos os nós enviam seu vetor de distância local para seus vizinhos

DV em a:
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$



- Algumas assimetrias:
- elo perdido
 - custo maior

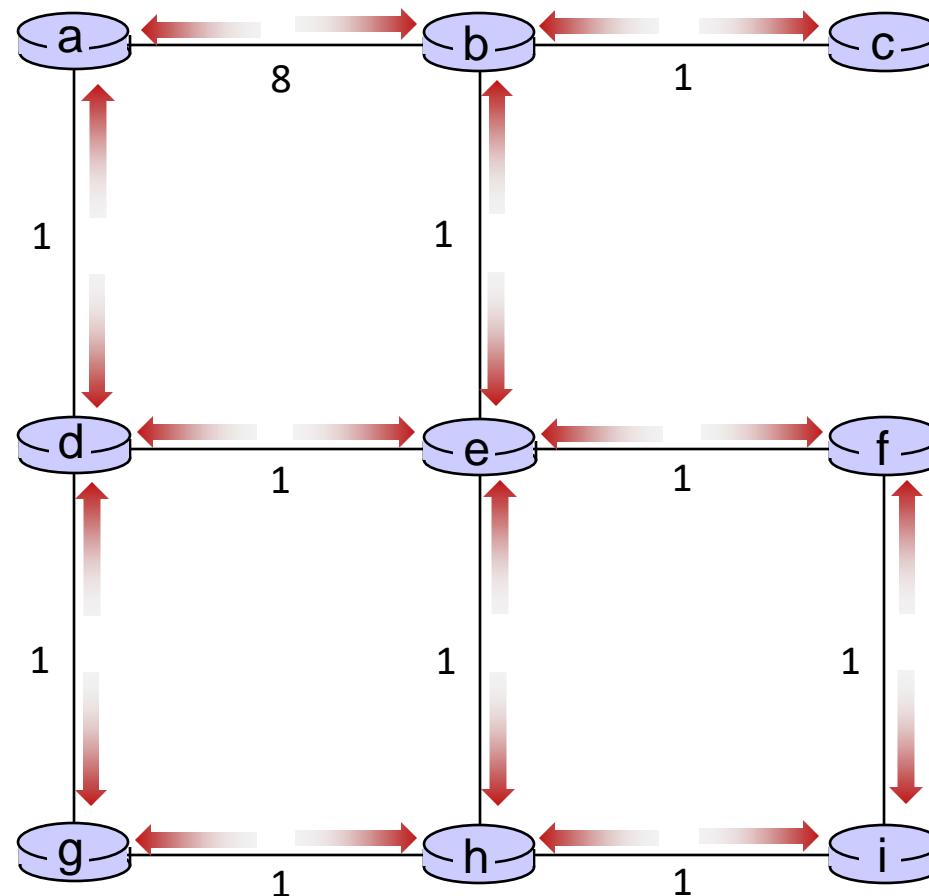
Exemplo de vetor de distância: iteração



$t=1$

Todos os nós:

- receber vetores de distância dos vizinhos
- calculam seu novo vetor de distância local
- enviam seu novo vetor de distância local para os vizinhos



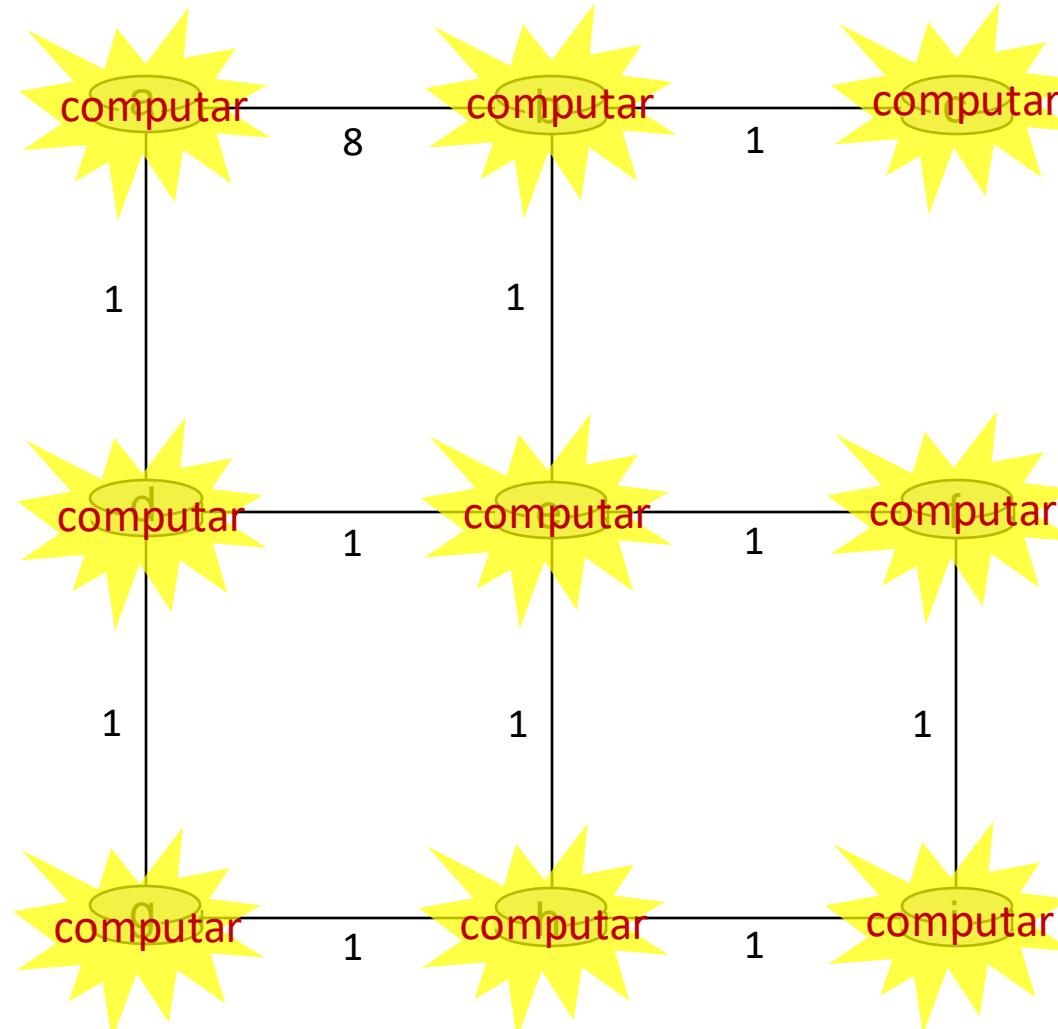
Exemplo de vetor de distância: iteração



t=1

Todos os nós:

- receber vetores de distância dos vizinhos
 - calculam seu novo vetor de distância local
 - enviam seu novo vetor de distância local para os vizinhos



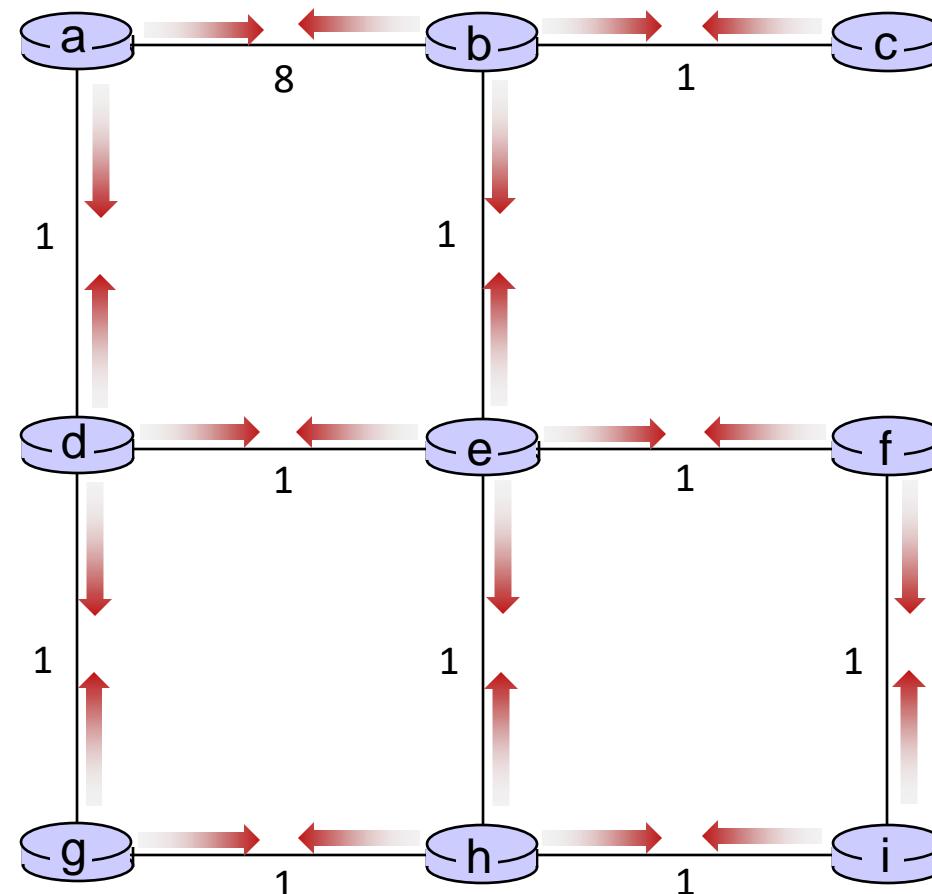
Exemplo de vetor de distância: iteração



t=1

Todos os nós:

- receber vetores de distância dos vizinhos
- calculam seu novo vetor de distância local
- enviam seu novo vetor de distância local para os vizinhos



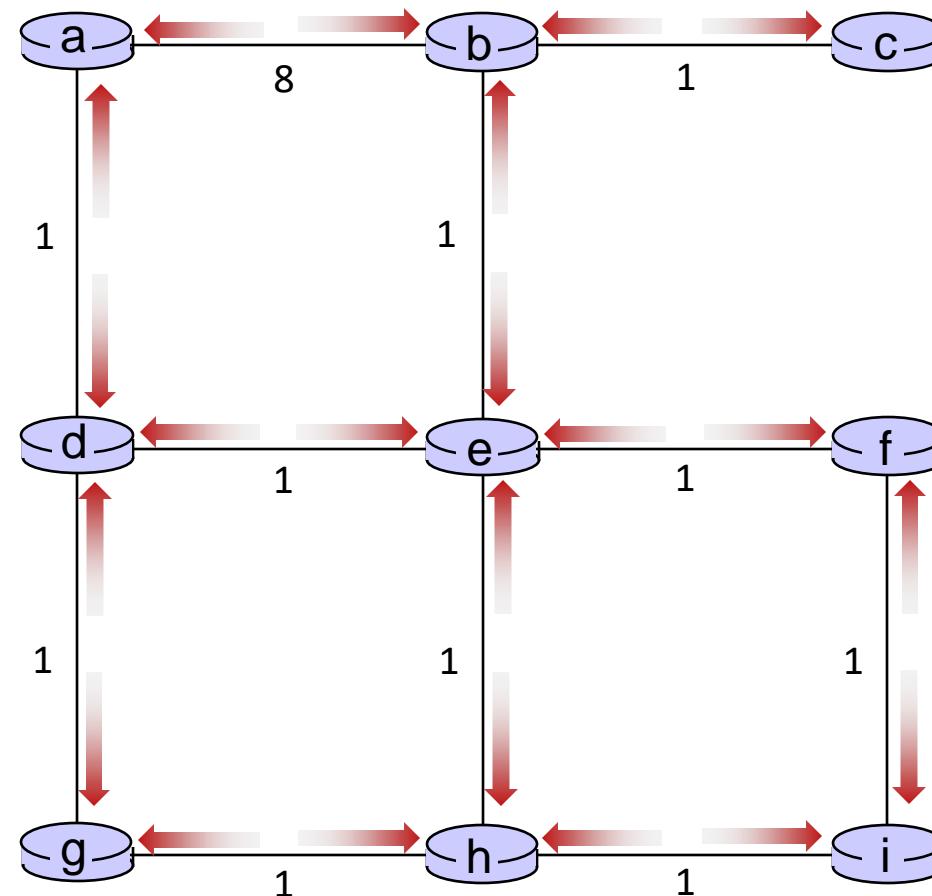
Exemplo de vetor de distância: iteração



$t=2$

Todos os nós:

- receber vetores de distância dos vizinhos
- calculam seu novo vetor de distância local
- enviam seu novo vetor de distância local para os vizinhos



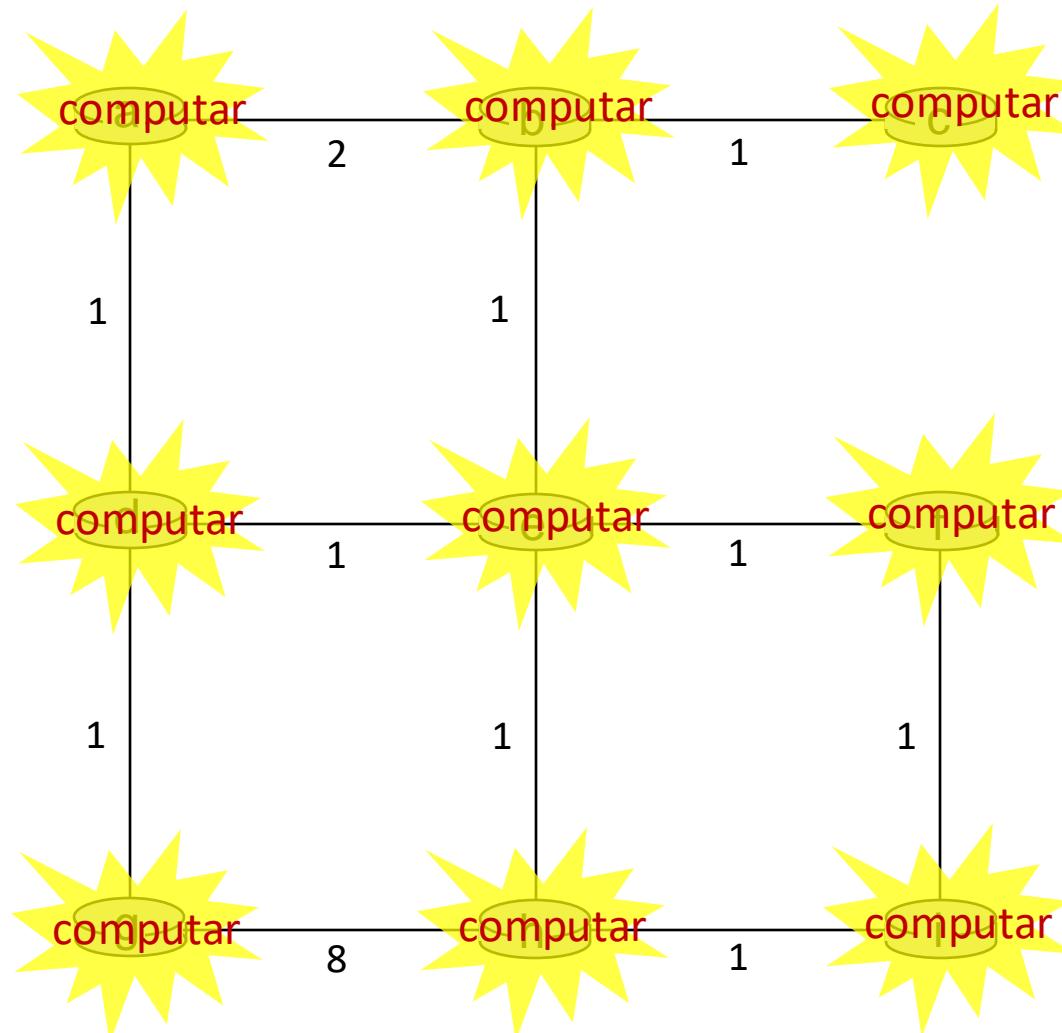
Exemplo de vetor de distância: iteração



$t=2$

Todos os nós:

- receber vetores de distância dos vizinhos
- calculam seu novo vetor de distância local
- enviam seu novo vetor de distância local para os vizinhos



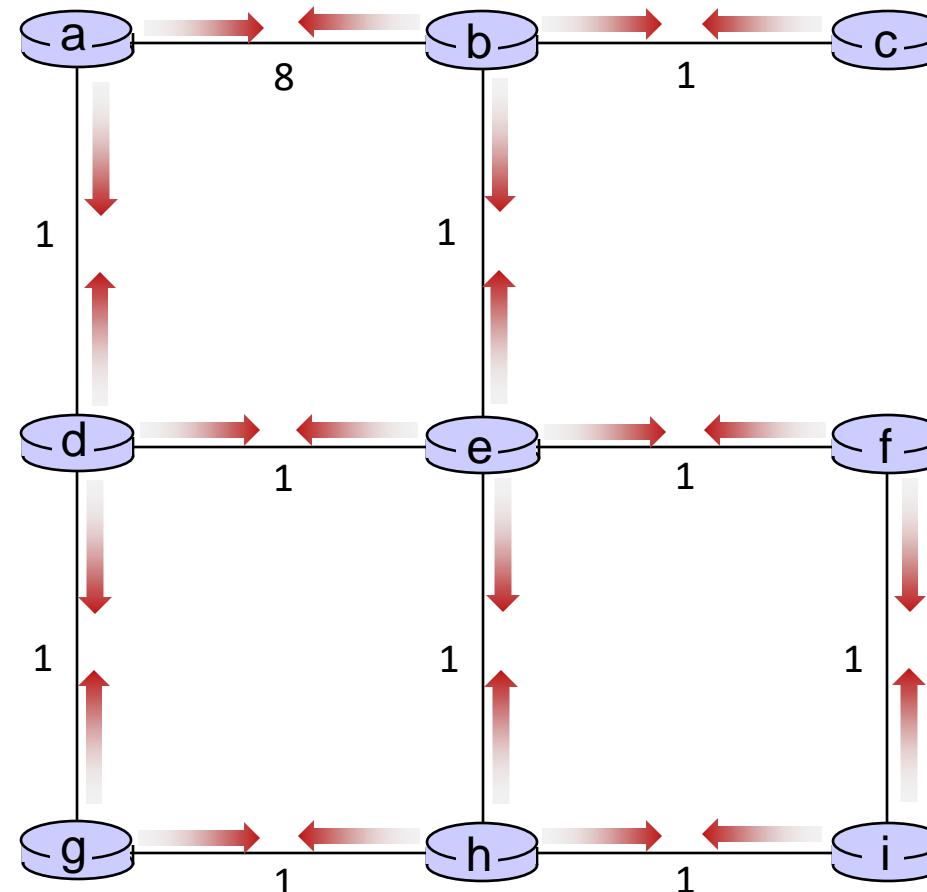
Exemplo de vetor de distância: iteração



$t=2$

Todos os nós:

- receber vetores de distância dos vizinhos
- calculam seu novo vetor de distância local
- enviam seu novo vetor de distância local para os vizinhos



Exemplo de vetor de distância: iteração

.... e assim por diante

A seguir, vamos dar uma olhada nos *cálculos* iterativos nos nós

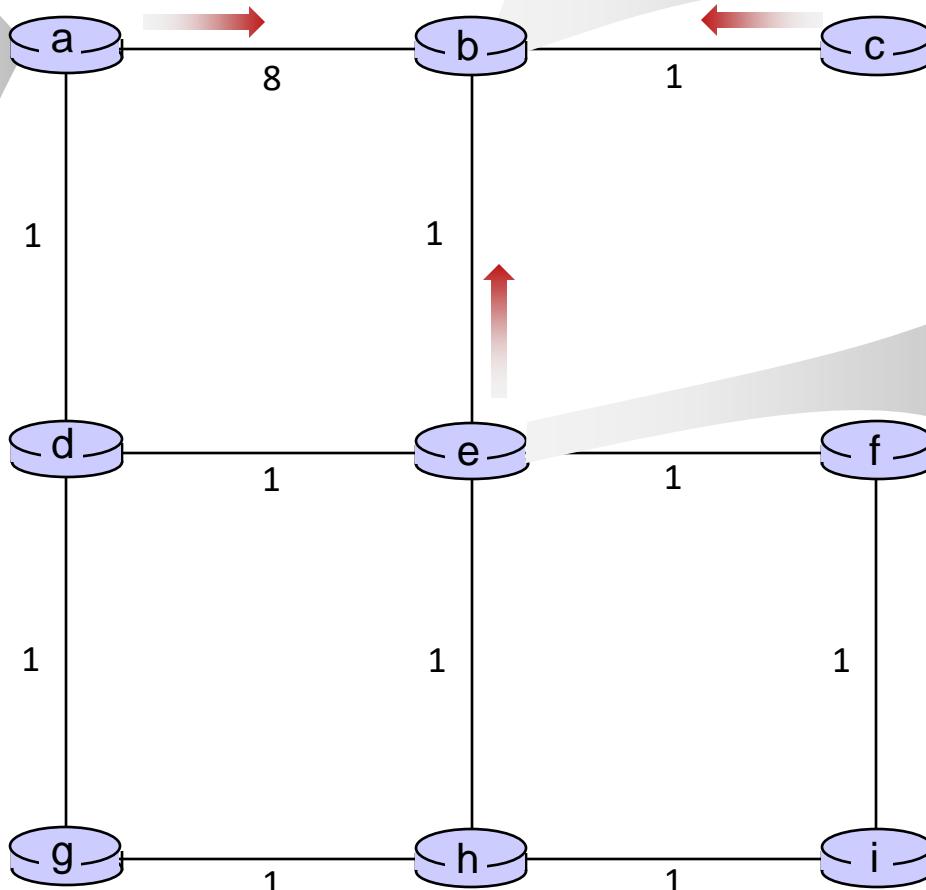
Exemplo de vetor de distância



$t=1$

- b recebe DVs de a, c, e

DV em a:
$D_a(a)=0$
$D_a(b)=8$
$D_a(c)=\infty$
$D_a(d)=1$
$D_a(e)=\infty$
$D_a(f)=\infty$
$D_a(g)=\infty$
$D_a(h)=\infty$
$D_a(i)=\infty$



DV em b:
$D_b(a)=8$
$D_b(c)=1$
$D_b(d)=\infty$
$D_b(e)=1$
$D_b(f)=\infty$
$D_b(g)=\infty$
$D_b(h)=\infty$
$D_b(i)=\infty$

DV em c:
$D_c(a)=\infty$
$D_c(b)=1$
$D_c(c)=0$
$D_c(d)=\infty$
$D_c(e)=\infty$
$D_c(f)=\infty$
$D_c(g)=\infty$
$D_c(h)=\infty$
$D_c(i)=\infty$

DV em e:
$D_e(a)=\infty$
$D_e(b)=1$
$D_e(c)=\infty$
$D_e(d)=1$
$D_e(e)=0$
$D_e(f)=1$
$D_e(g)=\infty$
$D_e(h)=1$
$D_e(i)=\infty$

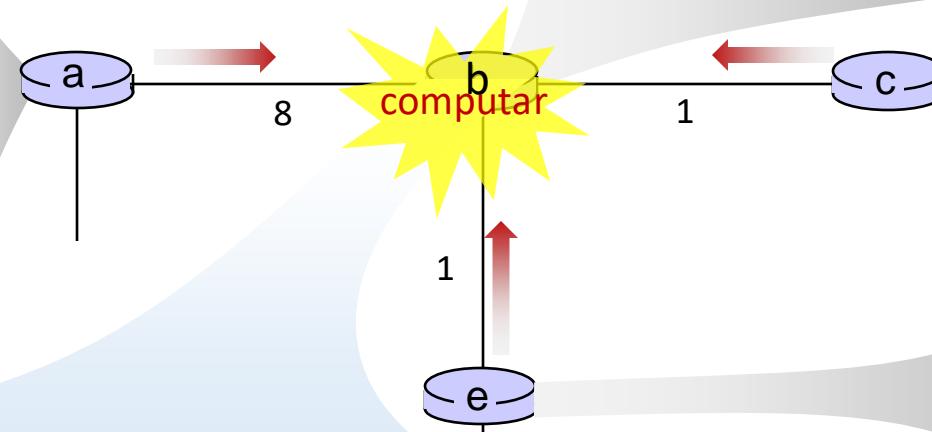
Exemplo de vetor de distância



$t=1$

- b recebe DVs de a, c, e, calcula:

DV em a:
$D_a(a)=0$
$D_a(b)=8$
$D_a(c)=\infty$
$D_a(d)=1$
$D_a(e)=\infty$
$D_a(f)=\infty$
$D_a(g)=\infty$
$D_a(h)=\infty$
$D_a(i)=\infty$



$$D_b(a) = \min\{c + D_{b,aa}(a), c + D_{b,cc}(a), c + D_{b,ee}(a)\} = \min\{8, \infty, \infty\} = 8$$

$$D_b(c) = \min\{c + D_{b,aa}(c), c + D_{b,cc}(c), c + D_{b,ee}(c)\} = \min\{\infty, 1, \infty\} = 1$$

$$D_b(d) = \min\{c + D_{b,aa}(d), c + D_{b,cc}(d), c + D_{b,ee}(d)\} = \min\{9, 2, \infty\} = 2$$

$$D_b(e) = \min\{c + D_{b,aa}(e), c + D_{b,cc}(e), c + D_{b,ee}(e)\} = \min\{\infty, \infty, 1\} = 1$$

$$D_b(f) = \min\{c + D_{b,aa}(f), c + D_{b,cc}(f), c + D_{b,ee}(f)\} = \min\{\infty, \infty, 2\} = 2$$

$$D_b(g) = \min\{c + D_{b,aa}(g), c + D_{b,cc}(g), c + D_{b,ee}(g)\} = \min\{\infty, \infty, \infty\} = \infty$$

$$D_b(h) = \min\{c + D_{b,aa}(h), c + D_{b,cc}(h), c + D_{b,ee}(h)\} = \min\{\infty, \infty, 2\} = 2$$

$$D_b(i) = \min\{c + D_{b,aa}(i), c + D_{b,cc}(i), c + D_{b,ee}(i)\} = \min\{\infty, \infty, \infty\} = \infty$$

DV em b:	
$D_b(a)=8$	$D_b(f)=\infty$
$D_b(c)=1$	$D_b(g)=\infty$
$D_b(d)=\infty$	$D_b(h)=\infty$
$D_b(e)=1$	$D_b(i)=\infty$

DV em c:
$D_c(a)=\infty$
$D_c(b)=1$
$D_c(c)=0$
$D_c(d)=\infty$
$D_c(e)=\infty$
$D_c(f)=\infty$
$D_c(g)=\infty$
$D_c(h)=\infty$
$D_c(i)=\infty$

DV em e:	
$D_e(a)=\infty$	
$D_e(b)=1$	
$D_e(c)=\infty$	
$D_e(d)=1$	
$D_e(e)=0$	
$D_e(f)=1$	
$D_e(g)=\infty$	
$D_e(h)=1$	
$D_e(i)=\infty$	

DV em b:	
$D_b(a)=8$	$D_b(f)=2$
$D_b(c)=1$	$D_b(g)=\infty$
$D_b(d)=2$	$D_b(h)=2$
$D_b(e)=1$	$D_b(i)=\infty$

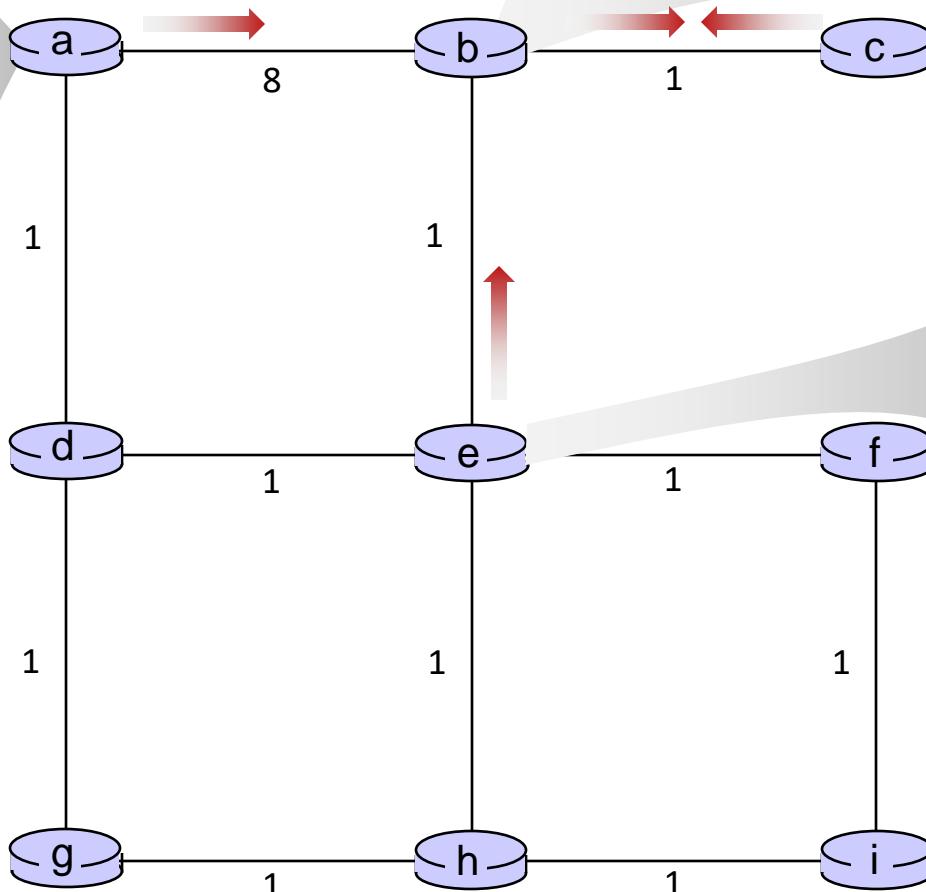
Exemplo de vetor de distância



$t=1$

- c recebe DVs de b

DV em a:
$D_a(a)=0$
$D_a(b)=8$
$D_a(c)=\infty$
$D_a(d)=1$
$D_a(e)=\infty$
$D_a(f)=\infty$
$D_a(g)=\infty$
$D_a(h)=\infty$
$D_a(i)=\infty$



DV em b:
$D_b(a)=8$
$D_b(c)=1$
$D_b(d)=\infty$
$D_b(e)=1$
$D_b(f)=\infty$
$D_b(g)=\infty$
$D_b(h)=\infty$
$D_b(i)=\infty$

DV em c:
$D_c(a)=\infty$
$D_c(b)=1$
$D_c(c)=0$
$D_c(d)=\infty$
$D_c(e)=\infty$
$D_c(f)=\infty$
$D_c(g)=\infty$
$D_c(h)=\infty$
$D_c(i)=\infty$

DV em e:
$D_e(a)=\infty$
$D_e(b)=1$
$D_e(c)=\infty$
$D_e(d)=1$
$D_e(e)=0$
$D_e(f)=1$
$D_e(g)=\infty$
$D_e(h)=1$
$D_e(i)=\infty$

Exemplo de vetor de distância



$t=1$

- c recebe DVs de b
calcula:

$$D_c(a) = \min\{c + D_{c,bb}(a)\} = 1 + 8 = 9$$

$$D_c(b) = \min\{c + D_{c,bb}(b)\} = 1 + 0 = 1$$

$$D_c(d) = \min\{c + D_{c,bb}(d)\} = 1 + \infty = \infty$$

$$D_c(e) = \min\{c + D_{c,bb}(e)\} = 1 + 1 = 2$$

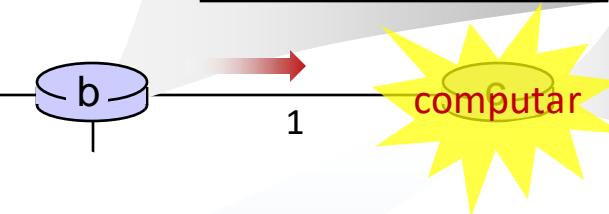
$$D_c(f) = \min\{c + D_{c,bb}(f)\} = 1 + \infty = \infty$$

$$D_c(g) = \min\{c + D_{c,bb}(g)\} = 1 + \infty = \infty$$

$$D_c(h) = \min\{c + D_{c,bb}(h)\} = 1 + \infty = \infty$$

$$D_c(i) = \min\{c + D_{c,bb}(i)\} = 1 + \infty = \infty$$

DV em b:	
$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$



DV em c:	
$D_c(a) = \infty$	
$D_c(b) = 1$	
$D_c(c) = 0$	
$D_c(d) = \infty$	
$D_c(e) = \infty$	
$D_c(f) = \infty$	
$D_c(g) = \infty$	
$D_c(h) = \infty$	
$D_c(i) = \infty$	

DV em c:

$$D_c(a) = 9$$

$$D_c(b) = 1$$

$$D_c(c) = 0$$

$$D_c(d) = 2$$

$$D_c(e) = \infty$$

$$D_c(f) = \infty$$

$$D_c(g) = \infty$$

$$D_c(h) = \infty$$

$$D_c(i) = \infty$$

* Confira os exercícios interativos on-line para obter mais exemplos:
http://gaia.cs.umass.edu/kurose_ross/interactive/

Exemplo de vetor de distância

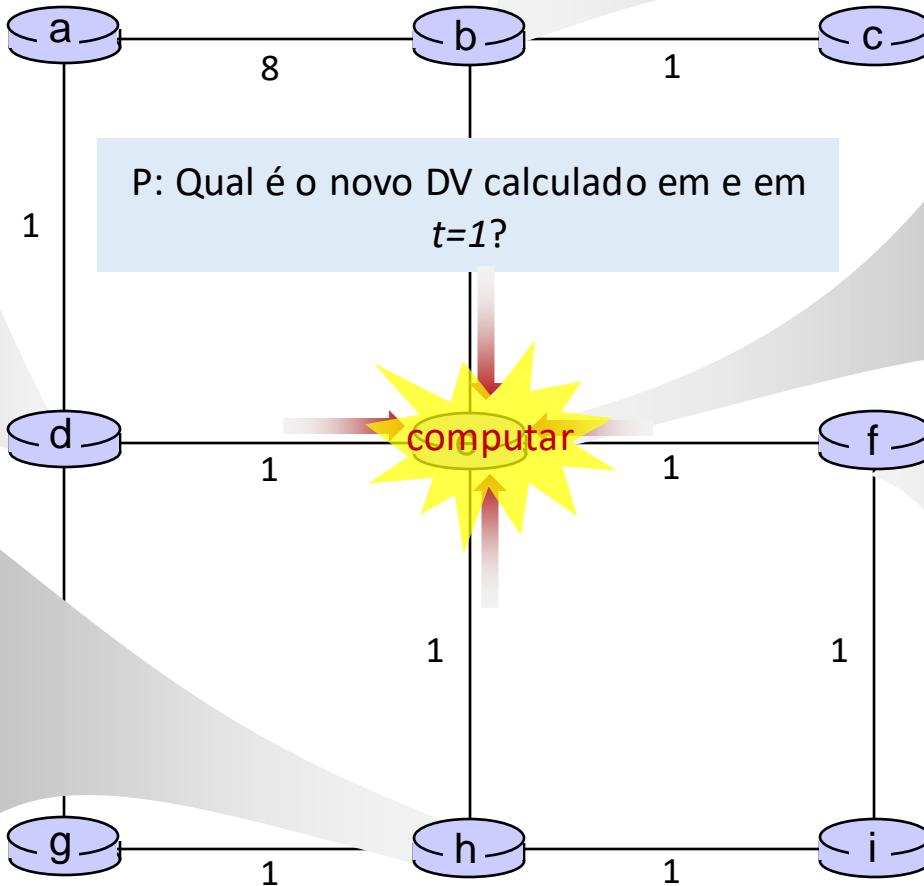


$t=1$

- e recebe DVs de b, d, f, h

DV em d:
$D_c(a) = 1$
$D_c(b) = \infty$
$D_c(c) = \infty$
$D_c(d) = 0$
$D_c(e) = 1$
$D_c(f) = \infty$
$D_c(g) = 1$
$D_c(h) = \infty$
$D_c(i) = \infty$

DV em h:
$D_c(a) = \infty$
$D_c(b) = \infty$
$D_c(c) = \infty$
$D_c(d) = \infty$
$D_c(e) = 1$
$D_c(f) = \infty$
$D_c(g) = 1$
$D_c(h) = 0$
$D_c(i) = 1$



DV em b:
$D_b(a) = 8$
$D_b(c) = 1$
$D_b(d) = \infty$
$D_b(e) = 1$
$D_b(f) = \infty$
$D_b(g) = \infty$
$D_b(h) = \infty$
$D_b(i) = \infty$

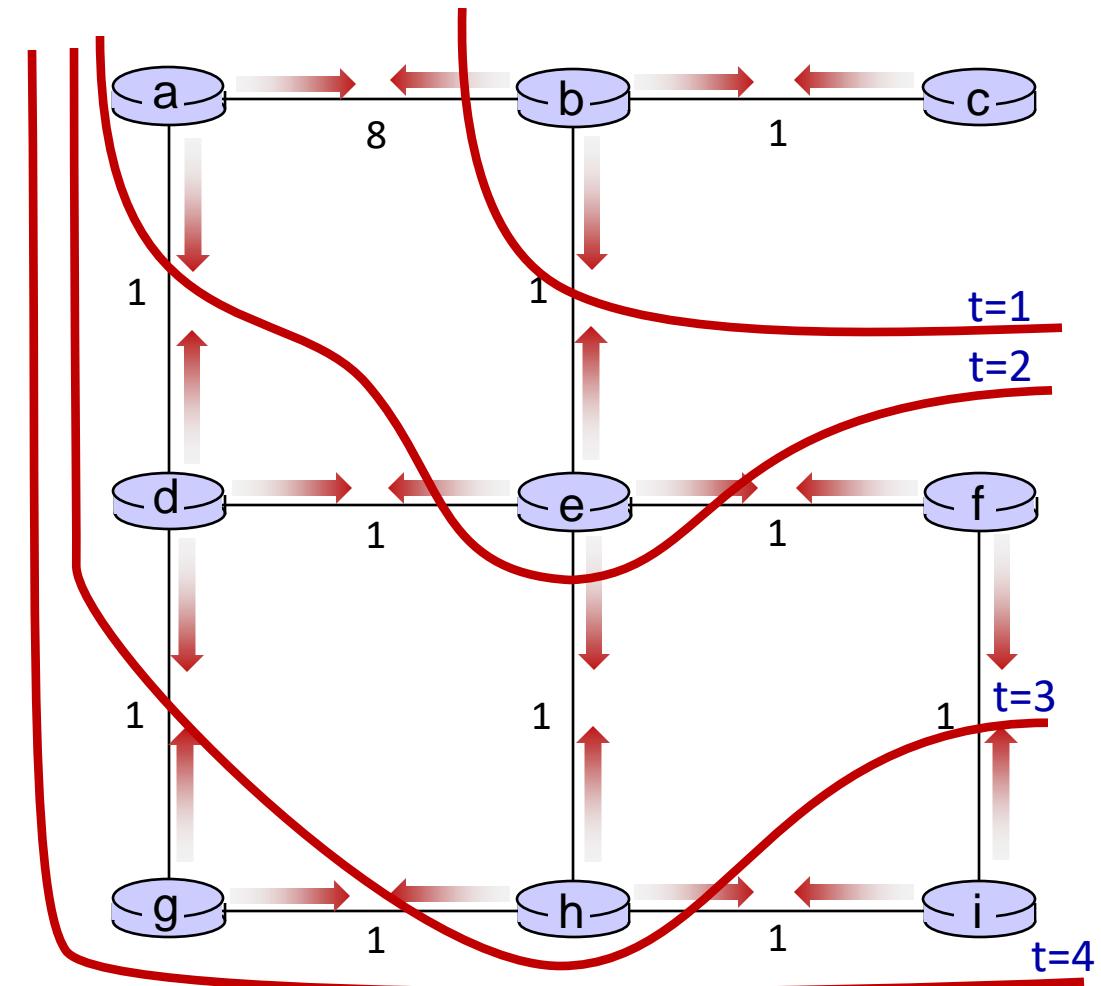
DV em e:
$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

DV em f:
$D_c(a) = \infty$
$D_c(b) = \infty$
$D_c(c) = \infty$
$D_c(d) = \infty$
$D_c(e) = 1$
$D_c(f) = 0$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = 1$

vetor de distância: difusão de informações de estado

Comunicação iterativa, as etapas de computação difundem informações pela rede:

-  t=0 O estado de c em t=0 é apenas em c
-  t=1 O estado de c em t=0 se propagou para b e pode influenciar os cálculos do vetor de distância até **1** salto de distância, ou seja, em b
-  t=2 O estado de c em t=0 agora pode influenciar os cálculos do vetor de distância até **2** hops de distância, ou seja, em b e agora também em a, e
-  t=3 O estado de c em t=0 pode influenciar os cálculos do vetor de distância até **3** hops de distância, ou seja, em d, f, h
-  t=4 O estado de c em t=0 pode influenciar os cálculos do vetor de distância até **4** hops de distância, ou seja, em g, i

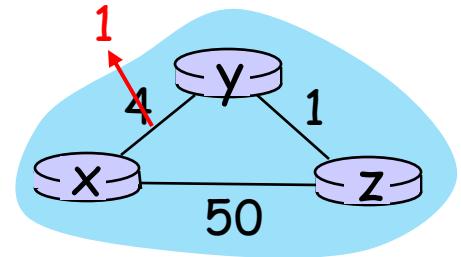


Vetor de distância: alterações no custo do link

alterações no custo do link:

- O nó detecta a mudança no custo do link local
- atualiza as informações de roteamento, recalcula o DV local
- Se o DV mudar, notifique os vizinhos
 - t_0 : y detecta a mudança no custo do link, atualiza seu DV e informa seus vizinhos.
 - t_1 : z recebe a atualização de y , atualiza seu DV, calcula o novo menor custo para x e envia seu DV aos vizinhos.
 - t_2 : y recebe a atualização de z e atualiza seu DV. Os custos mínimos de y não mudam, portanto y não envia uma mensagem para z .

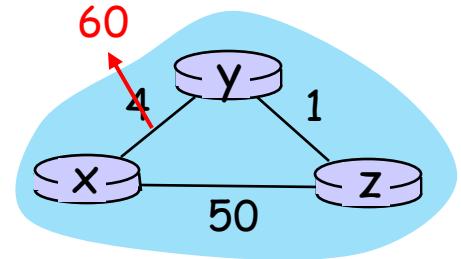
"boas
notícias
viaja rápido"



Vetor de distância: alterações no custo do link

alterações no custo do link:

- O nó detecta a mudança no custo do link local
- "As más notícias correm devagar" - problema de contagem até o infinito:
 - y vê que o link direto para x tem um novo custo de 60, mas z disse que tem um caminho com custo de 5. Portanto, y calcula "meu novo custo para x será 6, via z); notifica z sobre o novo custo de 6 para x.
 - z descobre que o caminho para x via y tem um novo custo de 6, então z calcula "meu novo custo para x será de 7 via y), notifica y sobre o novo custo de 7 para x.
 - y descobre que o caminho para x via z tem um novo custo 7, portanto y calcula "meu novo custo para x será 8 via y), notifica z sobre o novo custo de 8 para x.
 - z descobre que o caminho para x via y tem um novo custo 8, portanto z calcula "meu novo custo para x será 9 via y), notifica y sobre o novo custo de 9 para x.
 - ...
- consulte o texto para obter soluções. *Os algoritmos distribuídos são complicados!*



Comparação dos algoritmos LS e DV

complexidade da mensagem

LS: n roteadores, $O(n^2)$ mensagens enviadas

DV: troca entre vizinhos; o tempo de convergência varia

velocidade de convergência

LS: Algoritmo $O(n^2)$, mensagens $O(n)^2$

- pode ter oscilações

DV: o tempo de convergência varia

- pode ter loops de roteamento
- problema de contagem até o infinito

Robustez: o que acontece se o roteador apresentar mau funcionamento ou for comprometido?

LS:

- o roteador pode anunciar um custo de *link* incorreto
- cada roteador calcula apenas sua *própria* tabela

DV:

- O roteador DV pode anunciar um custo de *caminho* incorreto ("Tenho um caminho de custo *muito* baixo para todos os lugares"): *black-holing*
- o DV de cada roteador é usado por outros: o erro se propaga pela rede

Camada de rede: Roteiro do "plano de controle"

- introdução
- protocolos de roteamento
- **roteamento intra-ISP: OSPF**
- roteamento entre ISPs: BGP
- Plano de controle SDN
- Protocolo de Mensagens de Controle da Internet
- gerenciamento de rede, configuração
 - SNMP
 - NETCONF/YANG



Tornar o roteamento escalonável

nosso estudo de roteamento até o momento - idealizado

- todos os roteadores idênticos
- rede "plana"

... não é verdade na prática

escala: bilhões de destinos:

- não é possível armazenar todos os destinos nas tabelas de roteamento!
- a troca de tabelas de roteamento sobrecarregaria os links!

autonomia administrativa:

- Internet: uma rede de redes
- cada administrador de rede pode querer controlar o roteamento em sua própria rede

Abordagem da Internet para roteamento dimensionável

agregar roteadores em regiões conhecidas como "sistemas autônomos" (AS) (também conhecidos como "domínios")

intra-AS (também conhecido como "intra-domínio"): roteamento entre roteadores *dentro do mesmo AS ("rede")*

- todos os roteadores no AS devem executar o mesmo protocolo intra-domínio
- roteadores em diferentes AS podem executar diferentes protocolos de roteamento intra-domínio
- **roteador de gateway:** na "borda" de seu próprio AS, tem links para roteadores em outros ASs

inter-AS (também conhecido como "inter-domínio"): roteamento *entre ASs*

- os gateways executam o roteamento entre domínios (bem como o roteamento intra-domínio)

ASes interconectados

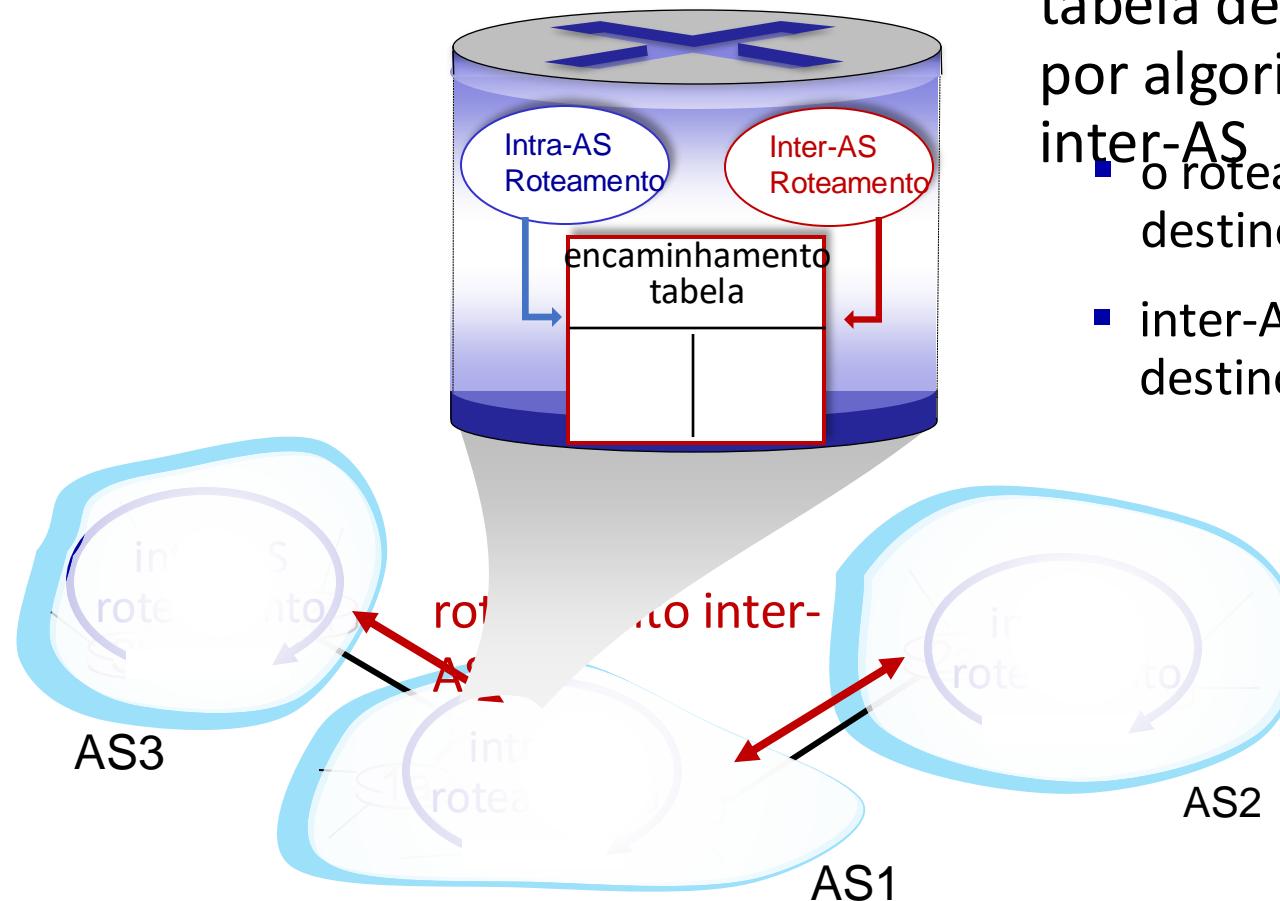
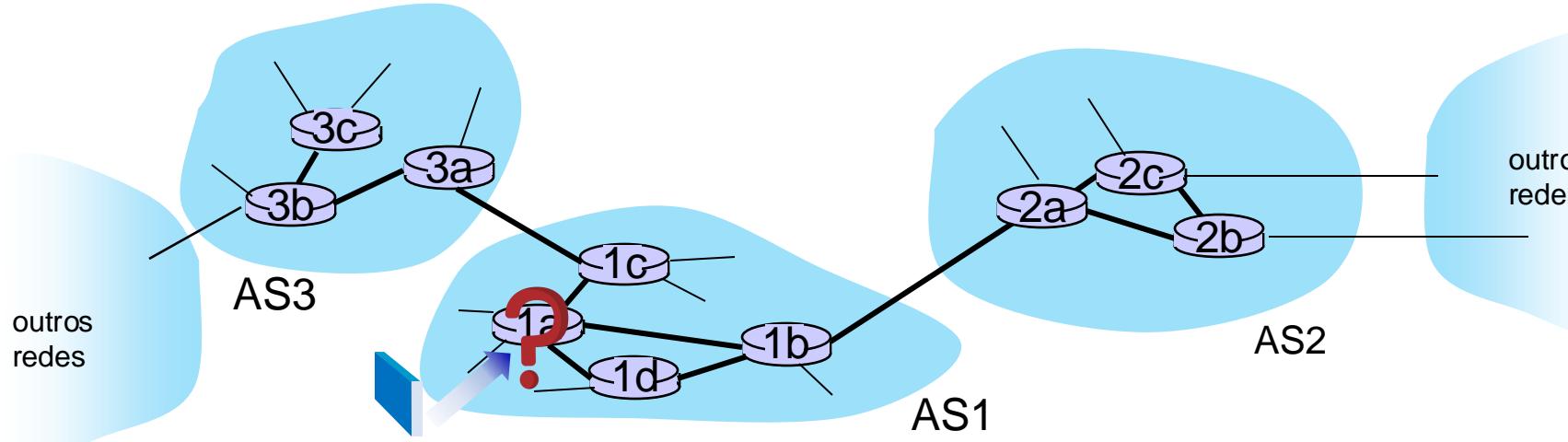


tabela de encaminhamento configurada por algoritmos de roteamento intra-AS e inter-AS

- o roteamento intra-AS determina entradas para destinos dentro do AS
- inter-AS e intra-AS determinam entradas para destinos externos

Roteamento inter-AS: uma função no encaminhamento intradomínio

- Suponha que o roteador em AS1 receba um datagrama destinado a fora de AS1:
 - deve encaminhar o pacote para o roteador de gateway em AS1, mas qual deles?



O roteamento entre domínios AS1 deve:

1. saber quais destinos podem ser alcançados pelo AS2 e quais pelo AS3
2. propagar essas informações de acessibilidade para todos os roteadores em AS1

Roteamento intra-AS: roteamento dentro de um AS

protocolos de roteamento intra-AS mais comuns:

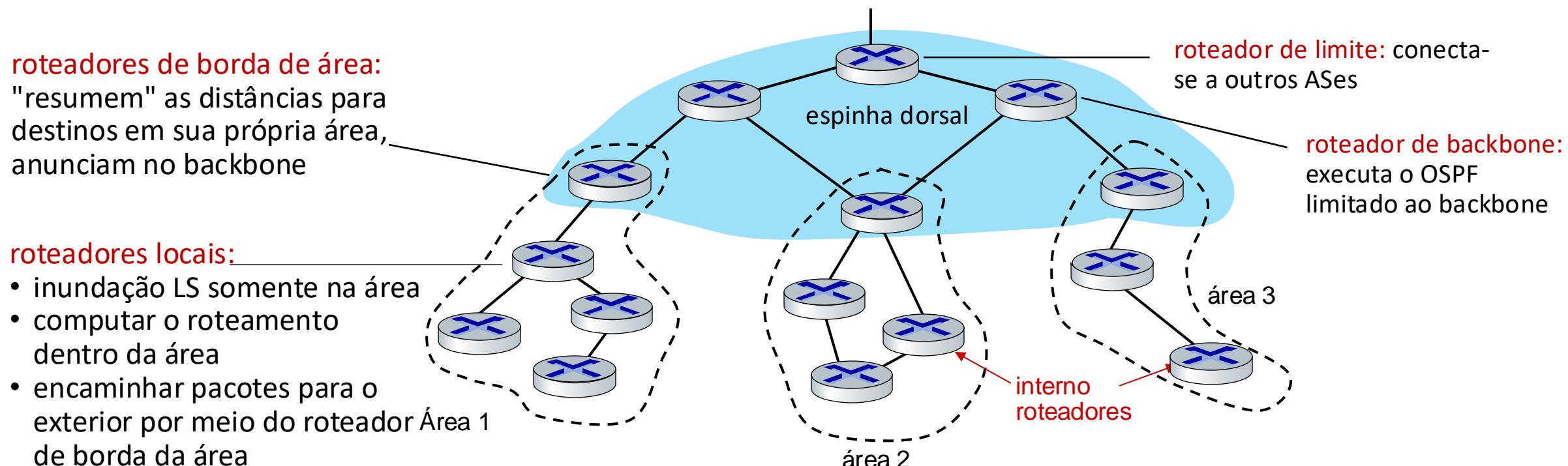
- **RIP: Protocolo de Informações de Roteamento [RFC 1723]**
 - DV clássico: DVs trocados a cada 30 segundos
 - não é mais amplamente utilizado
- **EIGRP: protocolo aprimorado de roteamento de gateway interior**
 - Baseado em DV
 - Anteriormente proprietário da Cisco por décadas (tornou-se aberto em 2013 [RFC 7868])
- **OSPF: caminho mais curto aberto primeiro [RFC 2328]**
 - roteamento link-state
 - Protocolo IS-IS (padrão ISO, não padrão RFC) essencialmente igual ao OSPF

Roteamento OSPF (Open Shortest Path First)

- "open": disponível publicamente
- estado do link clássico
 - Cada roteador inunda anúncios de estado de link OSPF (diretamente sobre IP em vez de usar TCP/UDP) para todos os outros roteadores em todo o AS
 - Possibilidade de várias métricas de custos de link: largura de banda, atraso
 - cada roteador tem uma topologia completa, usa o algoritmo de Dijkstra para calcular a tabela de encaminhamento
- *segurança*: todas as mensagens OSPF são autenticadas (para evitar invasões maliciosas)

OSPF hierárquico

- hierarquia de dois níveis: área local, backbone.
 - anúncios de estado de link inundados somente na área ou no backbone
 - cada nó tem uma topologia de área detalhada; sabe apenas a direção para chegar a outros destinos

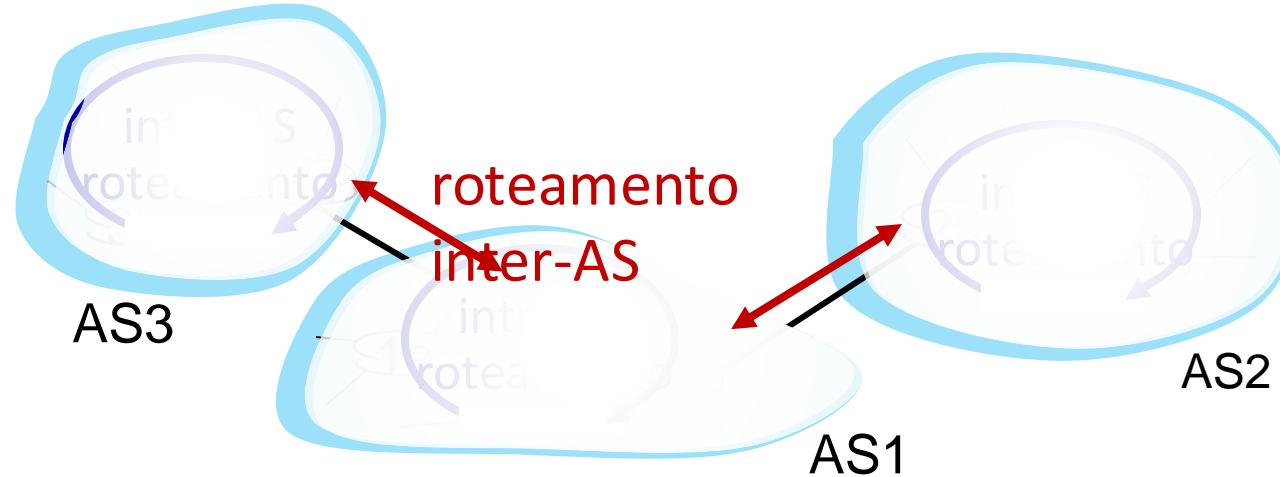


Camada de rede: Roteiro do "plano de controle"

- introdução
- protocolos de roteamento
- roteamento intra-ISP: OSPF
- **roteamento entre ISPs:
BGP**
- Plano de controle SDN
- Protocolo de Mensagens de Controle da Internet
- gerenciamento de rede,
configuração
 - SNMP
 - NETCONF/YANG



ASes interconectados

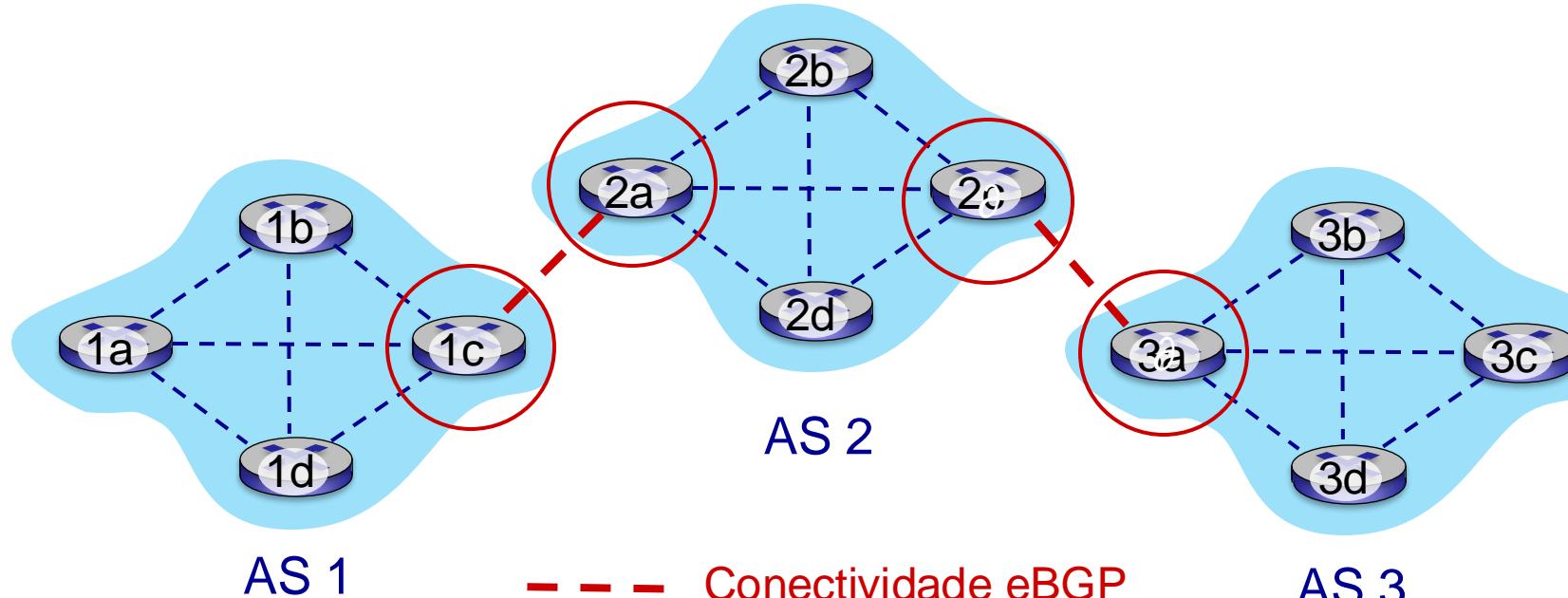


- ✓ intra-AS (também conhecido como "intra-domínio"): roteamento entre roteadores *dentro do mesmo AS ("rede")*
- inter-AS (também conhecido como "inter-domínio"): roteamento *entre ASs*

Roteamento inter-AS da Internet: BGP

- **BGP (Border Gateway Protocol)**: o protocolo de roteamento interdomínios de fato
 - "cola que mantém a Internet unida"
- permite que a sub-rede anuncie sua existência e os destinos que pode alcançar para o restante da Internet: "*Estou aqui, aqui está quem eu posso alcançar e como*"
- O BGP fornece a cada AS um meio de:
 - obter informações de acessibilidade da rede de destino dos ASes vizinhos (**eBGP**)
 - determinar rotas para outras redes com base em informações de acessibilidade e *políticas*
 - propagar informações de acessibilidade para todos os roteadores internos do AS (**iBGP**)
 - **anunciar** (para redes vizinhas) informações de acessibilidade do destino

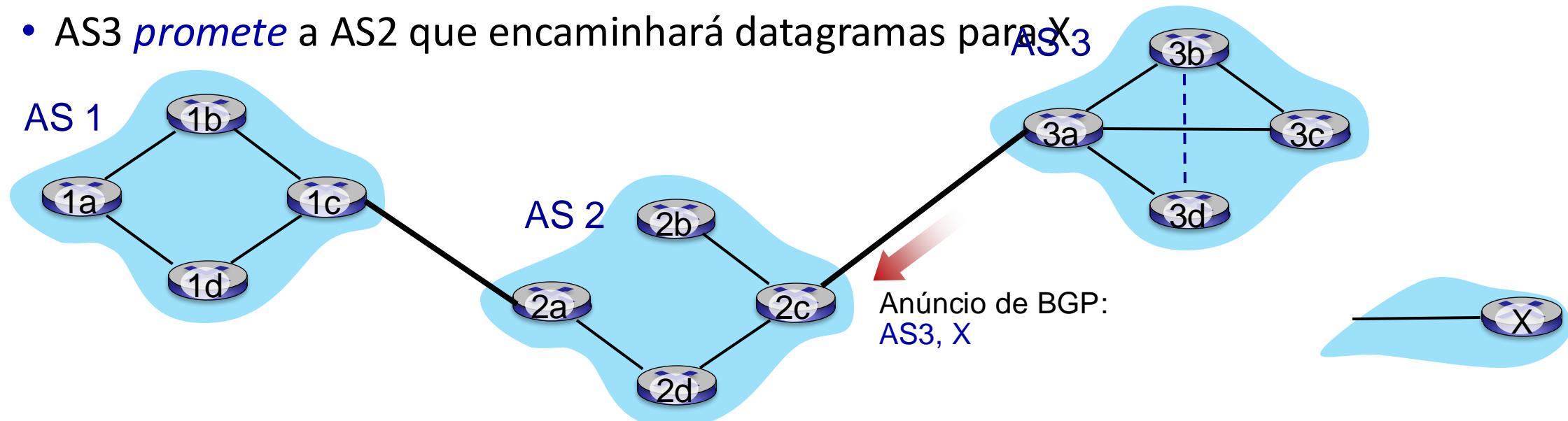
Conexões eBGP, iBGP



os roteadores de gateway executam os protocolos eBGP e iBGP

Noções básicas de BGP

- **Sessão BGP:** dois roteadores BGP ("pares") trocam mensagens BGP por meio de uma conexão TCP semipermanente:
 - anunciar *caminhos* para diferentes prefixos de rede de destino (o BGP é um protocolo de "vetor de caminho")
- quando o gateway AS3 3a anuncia **o caminho AS3,X** para o gateway AS2 2c:
 - AS3 *promete* a AS2 que encaminhará datagramas para X



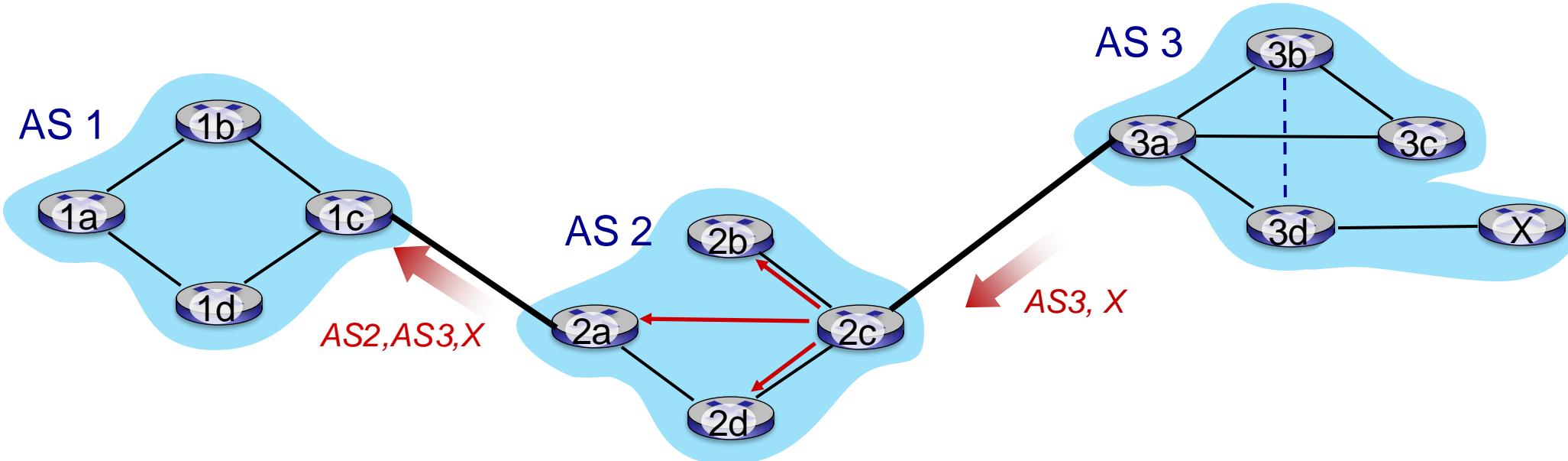
Mensagens do protocolo BGP

- Mensagens BGP trocadas entre pares por meio de conexão TCP
- Mensagens BGP [RFC 4371]:
 - **OPEN**: abre a conexão TCP com o par BGP remoto e autentica o par BGP remetente
 - **UPDATE**: anuncia um novo caminho (ou retira o antigo)
 - **KEEPALIVE**: mantém a conexão ativa na ausência de ATUALIZAÇÕES; também ACKs solicitação OPEN
 - **NOTIFICATION**: relata erros na mensagem anterior; também é usado para fechar a conexão

Atributos de caminho e rotas BGP

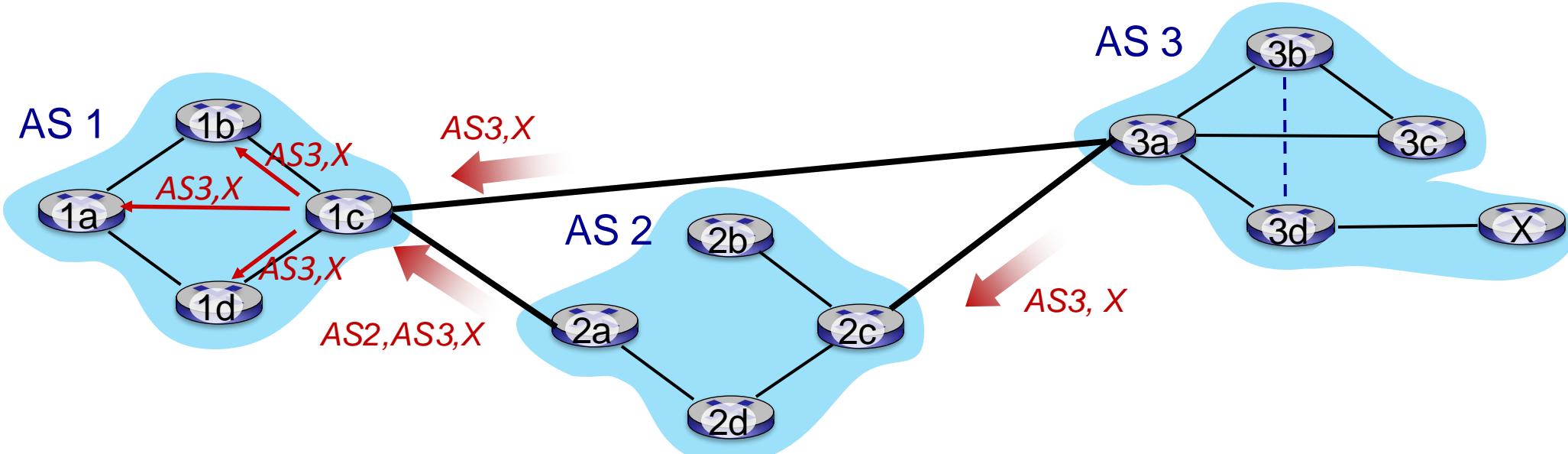
- Rota anunciada pelo BGP: prefixo + atributos
 - prefixo: destino que está sendo anunciado
 - dois atributos importantes:
 - AS-PATH: lista de ASes pelos quais o anúncio de prefixo passou
 - NEXT-HOP: indica o roteador específico do AS interno para o AS do próximo salto
- **roteamento baseado em políticas:**
 - O gateway que recebe o anúncio de rota usa *a política de importação* para aceitar/recusar o caminho (por exemplo, nunca rotear pelo AS Y).
 - A política de AS também determina se o caminho deve *ser anunciado* para outros ASes vizinhos

Anúncio de caminho BGP



- O roteador AS2 2c recebe o anúncio de caminho **AS3,X** (via eBGP) do roteador AS3 3a
- Com base na política AS2, o roteador AS2 2c aceita o caminho AS3,X, propaga (via iBGP) para todos os roteadores AS2
- Com base na política AS2, o roteador AS2 2a anuncia (via eBGP) o caminho **AS2, AS3, X** para o roteador AS1 1c

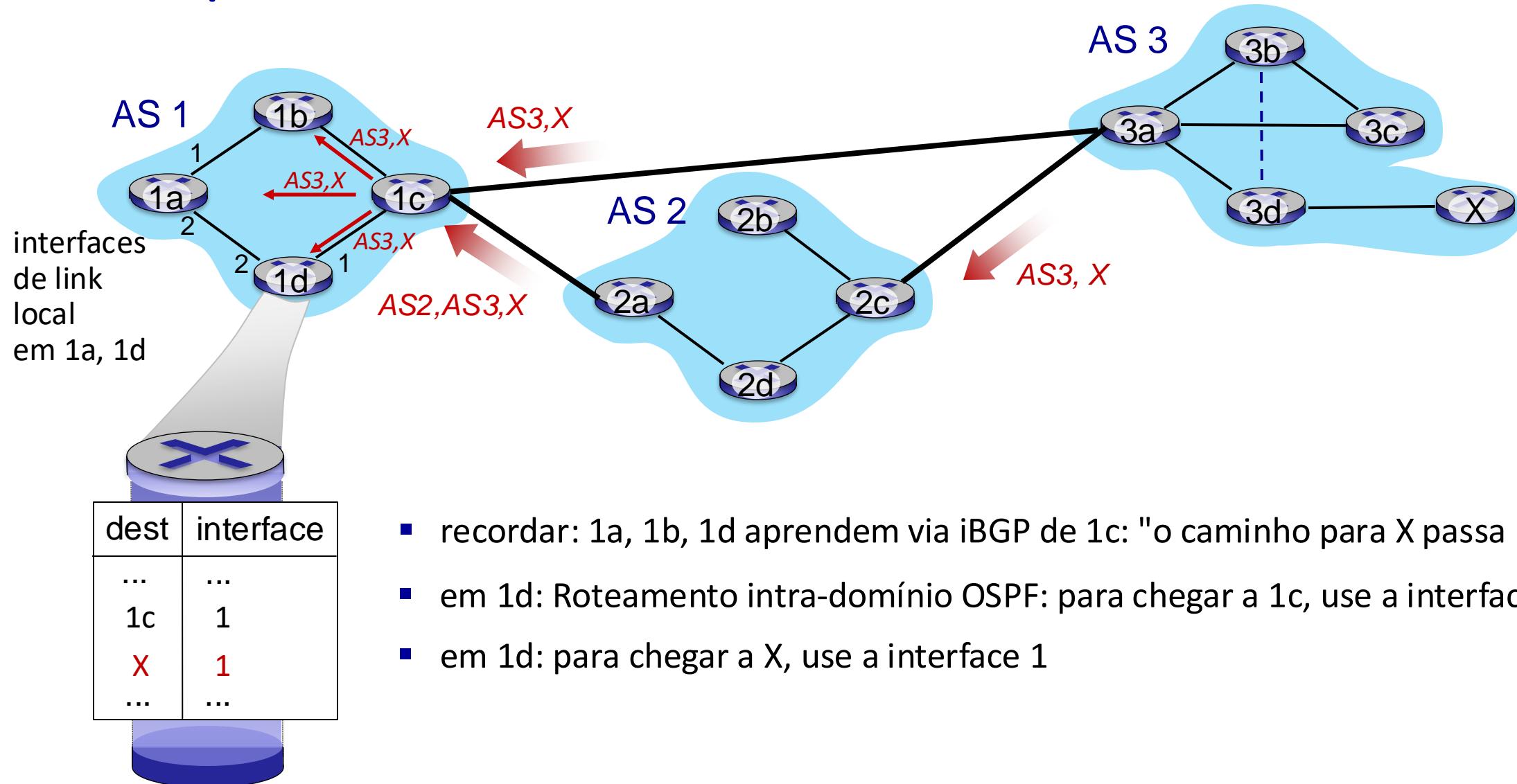
Anúncio de caminho BGP: vários caminhos



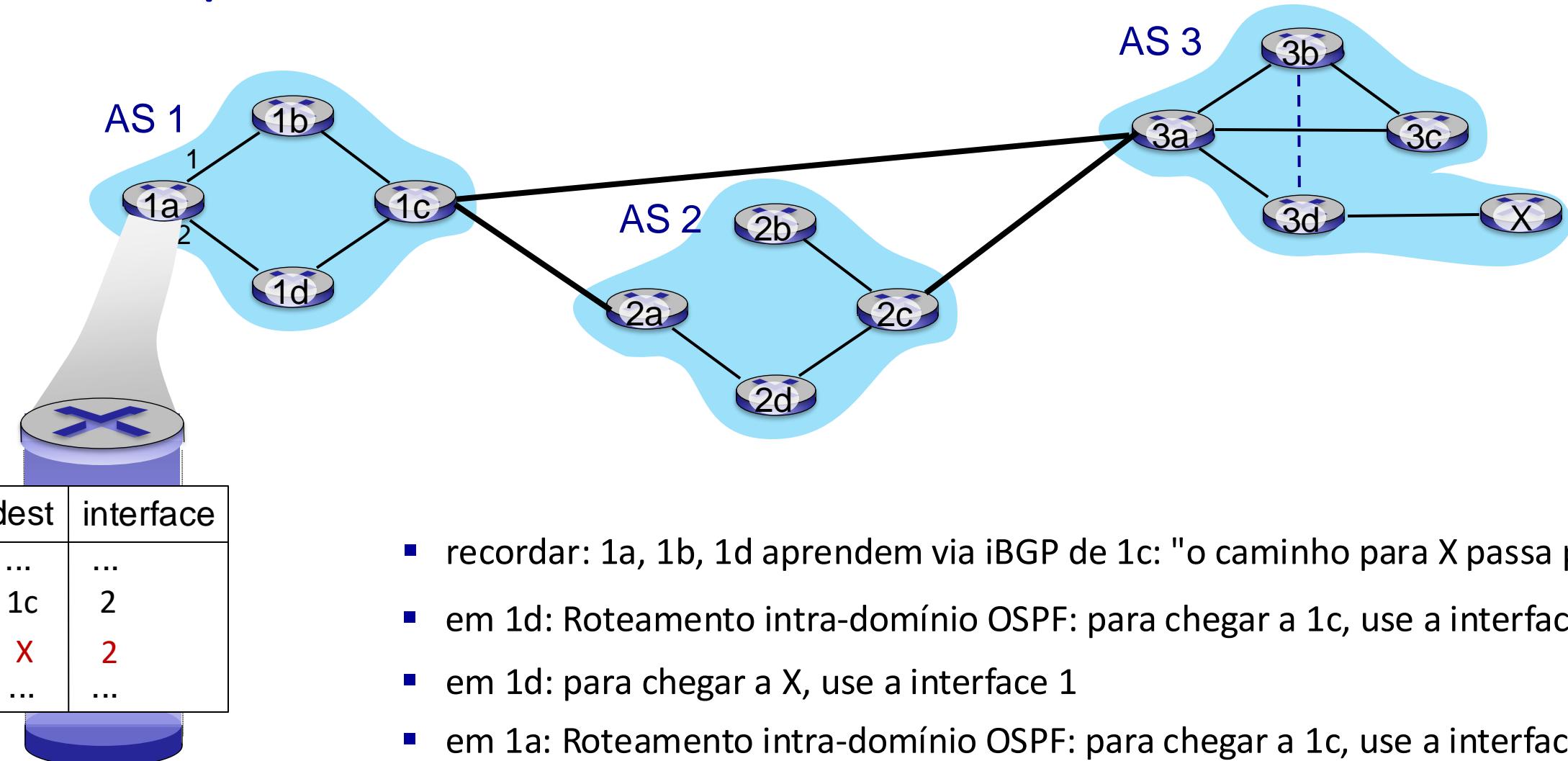
O roteador gateway pode conhecer **vários** caminhos até o destino:

- O roteador de gateway AS1 1c aprende o caminho **AS2,AS3,X** de 2a
- O roteador de gateway AS1 1c aprende o caminho **AS3,X** de 3a
- Com base na *política*, o roteador de gateway AS1 1c escolhe o caminho **AS3,X** e anuncia o caminho dentro do AS1 via iBGP

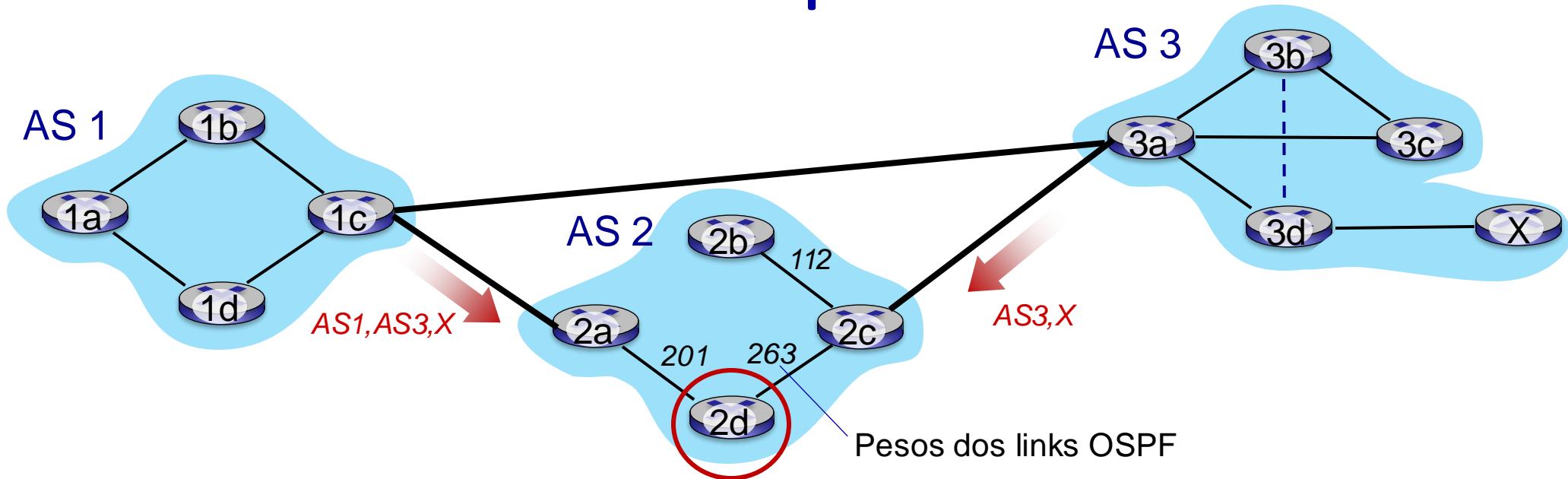
BGP: preenchimento de tabelas de encaminhamento



BGP: preenchimento de tabelas de encaminhamento

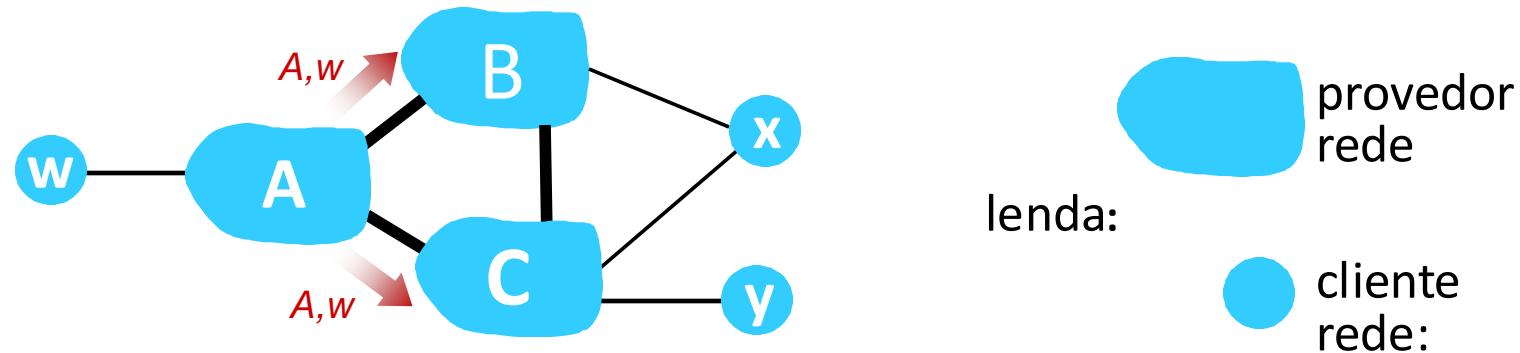


Roteamento de batata quente



- 2d descobre (via iBGP) que pode rotear para X via 2a ou 2c
- **roteamento hot potato:** escolha o gateway local que tenha o menor custo *intra-domínio* (por exemplo, 2d escolhe 2a, mesmo que haja mais saltos de AS para X): não se preocupe com o custo inter-domínio!

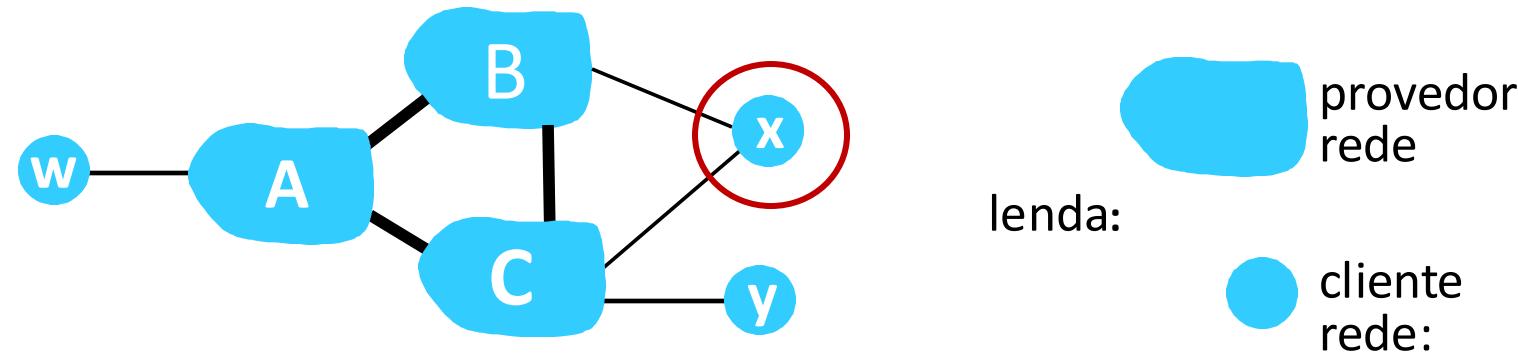
BGP: alcançar a política por meio de anúncios



O ISP só quer rotear o tráfego de/para suas redes de clientes (não quer transportar o tráfego de trânsito entre outros ISPs - uma política típica do "mundo real")

- A anuncia o caminho Aw para B e para C
- B *opta por não anunciar* BAw para C!
 - B não obtém nenhuma "receita" pelo roteamento de $CBAw$, pois nenhum de C, A, w é cliente de B
 - C *não aprende* sobre o caminho do $CBAw$
- C roteará CAw (sem usar B) para chegar a w

BGP: alcançando a política por meio de anúncios (mais)



O ISP só quer rotear o tráfego de/para suas redes de clientes (não quer transportar o tráfego de trânsito entre outros ISPs - uma política típica do "mundo real")

- A,B,C são **redes de provedores**
- x,w,y são **clientes** (de redes de provedores)
- x é **dual-homed**: conectado a duas redes
- **política a ser aplicada**: x não deseja rotear de B para C via x
 - ... portanto, x não anunciará para B uma rota para C

Seleção de rotas BGP

- O roteador pode conhecer mais de uma rota para o AS de destino e seleciona a rota com base nela:
 1. atributo de valor de preferência local: decisão de política
 2. AS-PATH mais curto
 3. roteador NEXT-HOP mais próximo: roteamento hot potato
 4. critérios adicionais

Por que o roteamento intra-AS e inter-AS é diferente?

política:

- inter-AS: o administrador deseja ter controle sobre como seu tráfego é roteado, quem roteia por sua rede
- intra-AS: administrador único, portanto a política é menos problemática

escala:

- o roteamento hierárquico economiza o tamanho da tabela e reduz o tráfego de atualização

desempenho:

- intra-AS: pode se concentrar no desempenho
- inter-AS: a política predomina sobre o desempenho

Camada de rede: Roteiro do "plano de controle"

- introdução
- protocolos de roteamento
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- **Plano de controle SDN**

- Protocolo de Mensagens de Controle da Internet



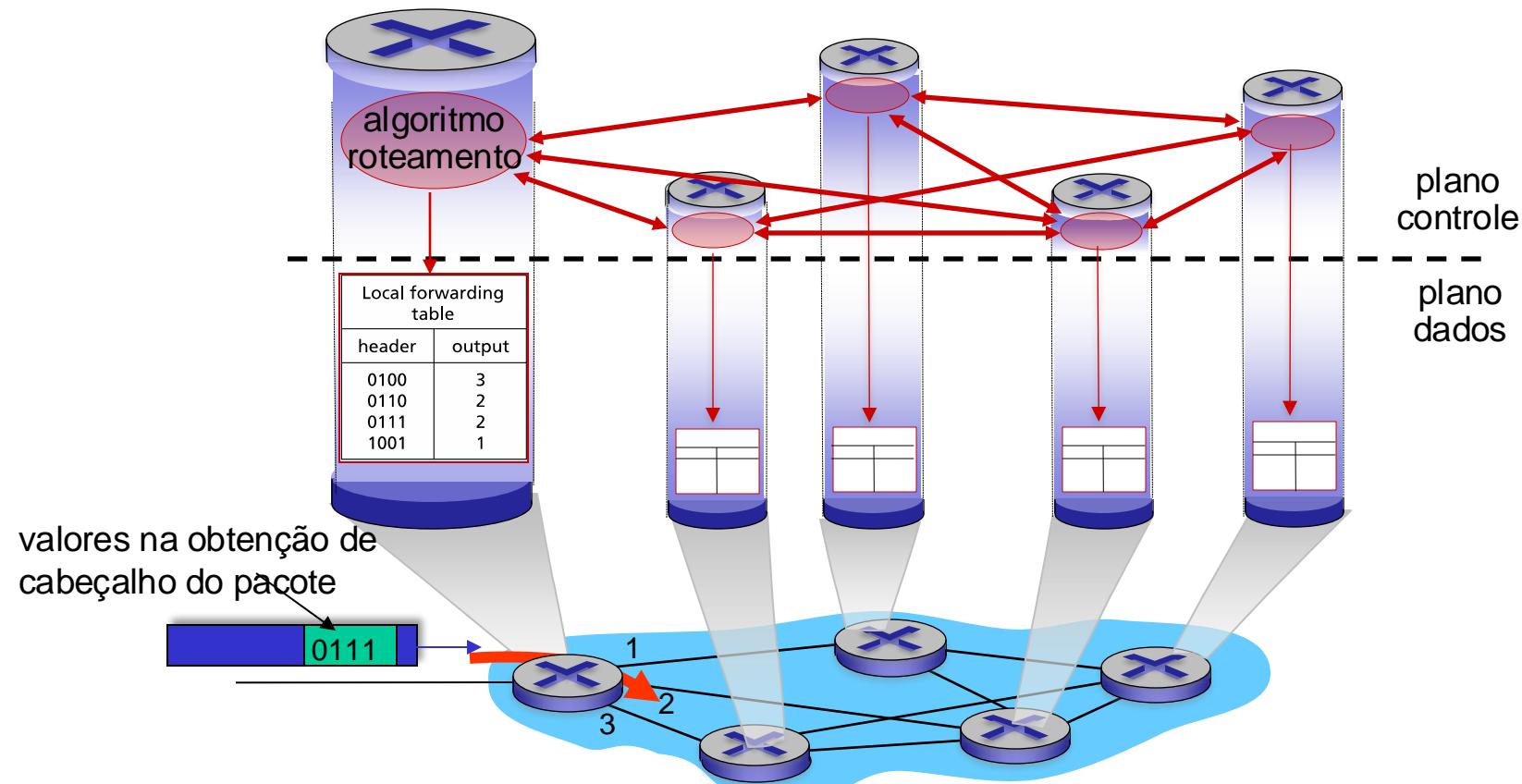
- gerenciamento de rede, configuração
 - SNMP
 - NETCONF/YANG

Rede definida por software (SDN)

- Camada de rede da Internet: historicamente implementada por meio de uma abordagem de controle distribuída e por roteador:
 - O roteador *monolítico* contém hardware de comutação, executa a implementação proprietária dos protocolos padrão da Internet (IP, RIP, IS-IS, OSPF, BGP) no sistema operacional proprietário do roteador (por exemplo, Cisco IOS)
 - diferentes "middleboxes" para diferentes funções da camada de rede: firewalls,平衡adores de carga, caixas NAT, ...
- ~2005: interesse renovado em repensar o plano de controle da rede

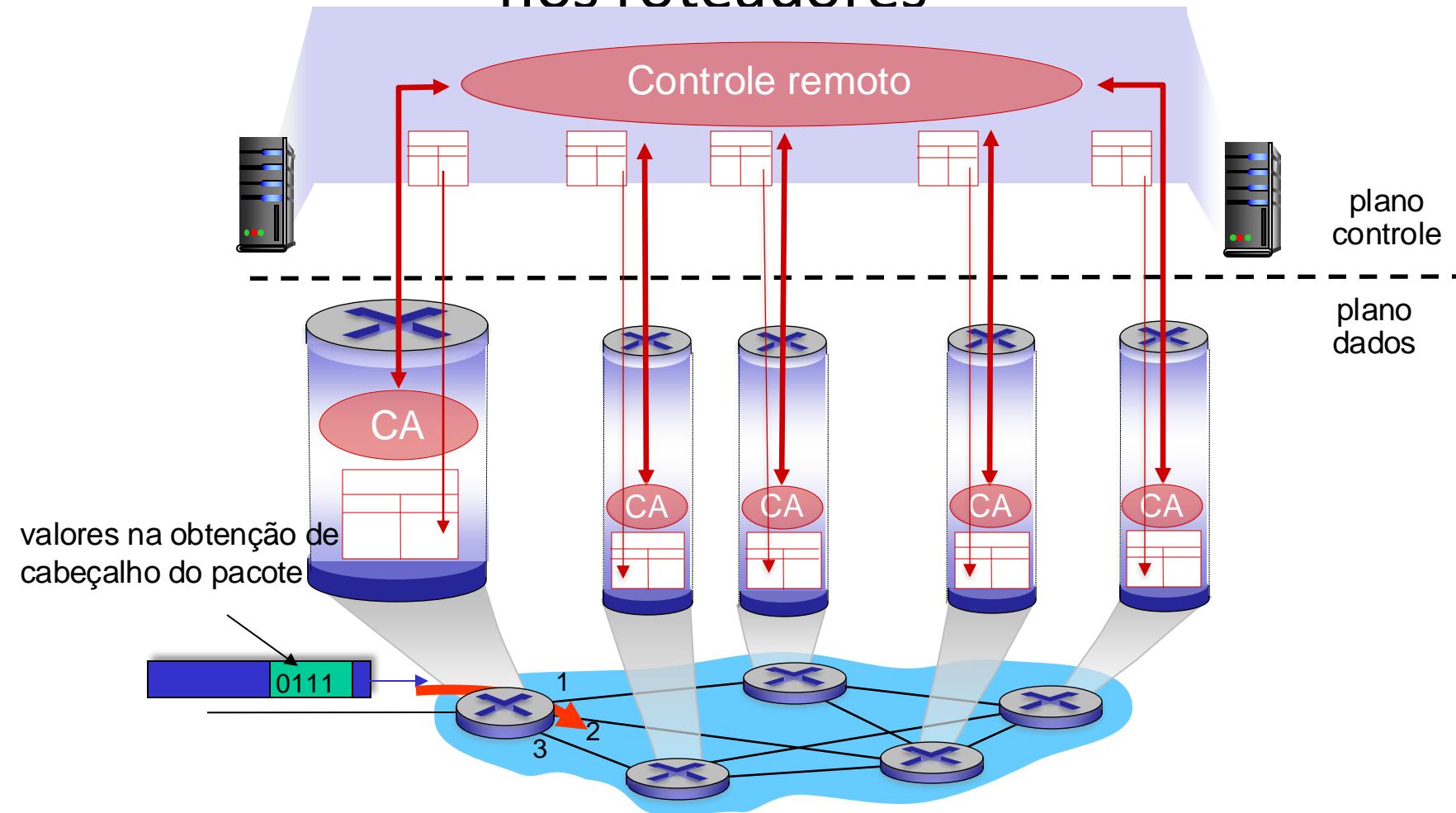
Plano de controle por roteador

Os componentes individuais do algoritmo de roteamento *em cada roteador* interagem no plano de controle para calcular as tabelas de encaminhamento



Plano de controle de SDN (Software-Defined Networking)

O controlador remoto calcula e instala tabelas de encaminhamento nos roteadores



Rede definida por software (SDN)

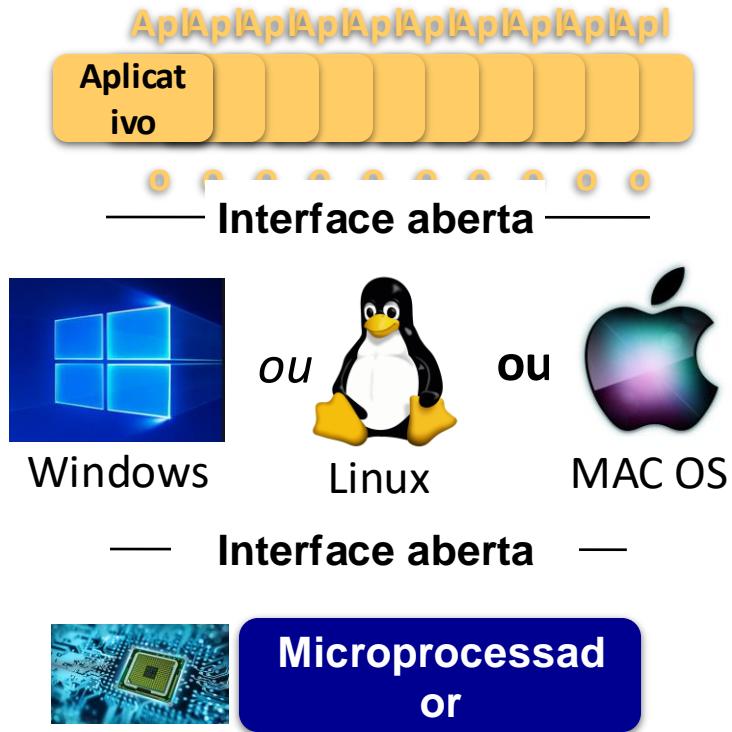
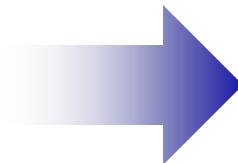
Por que um plano de controle *logicamente centralizado*?

- Gerenciamento de rede mais fácil: evita configurações incorretas do roteador, maior flexibilidade dos fluxos de tráfego
- O encaminhamento baseado em tabela (lembre-se da API OpenFlow) permite "programar" roteadores
 - "programação" centralizada mais fácil: calcular tabelas de forma centralizada e distribuir
 - "Programação" distribuída mais difícil: computar tabelas como resultado do algoritmo distribuído (protocolo) implementado em cada roteador
- implementação aberta (não proprietária) do plano de controle
 - fomente a inovação: deixe 1000 flores desabrocharem

Analogia da SDN: revolução do mainframe ao PC

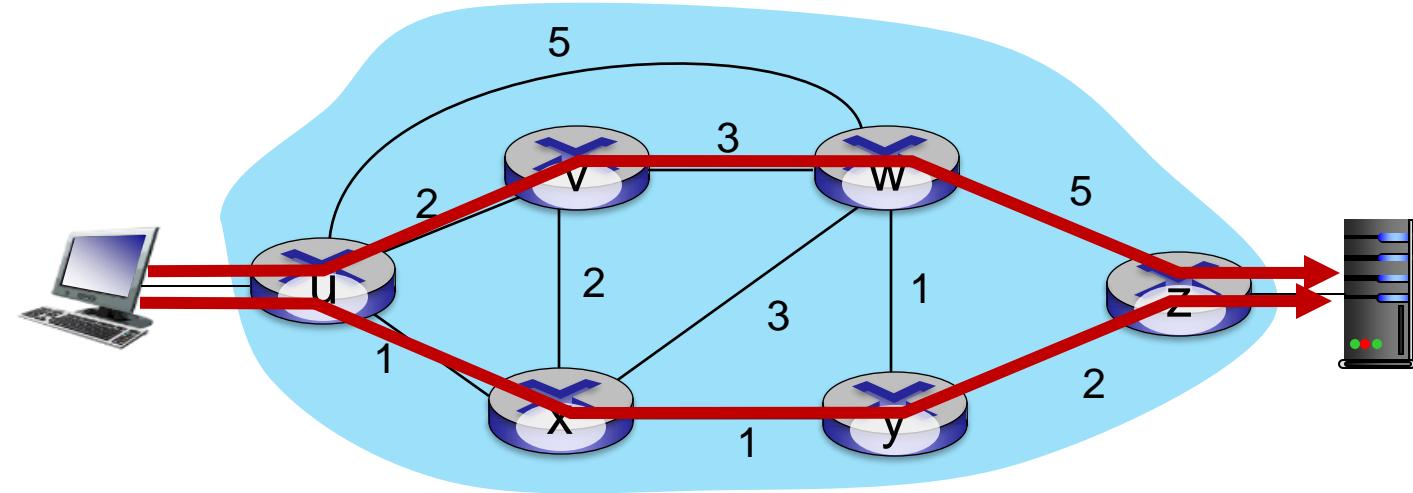


Integrado verticalmente
Fechado, proprietário
Inovação lenta
Pequeno setor



Horizontal
Interfaces abertas
Inovação rápida
Grande setor

Engenharia de tráfego: difícil com o roteamento tradicional

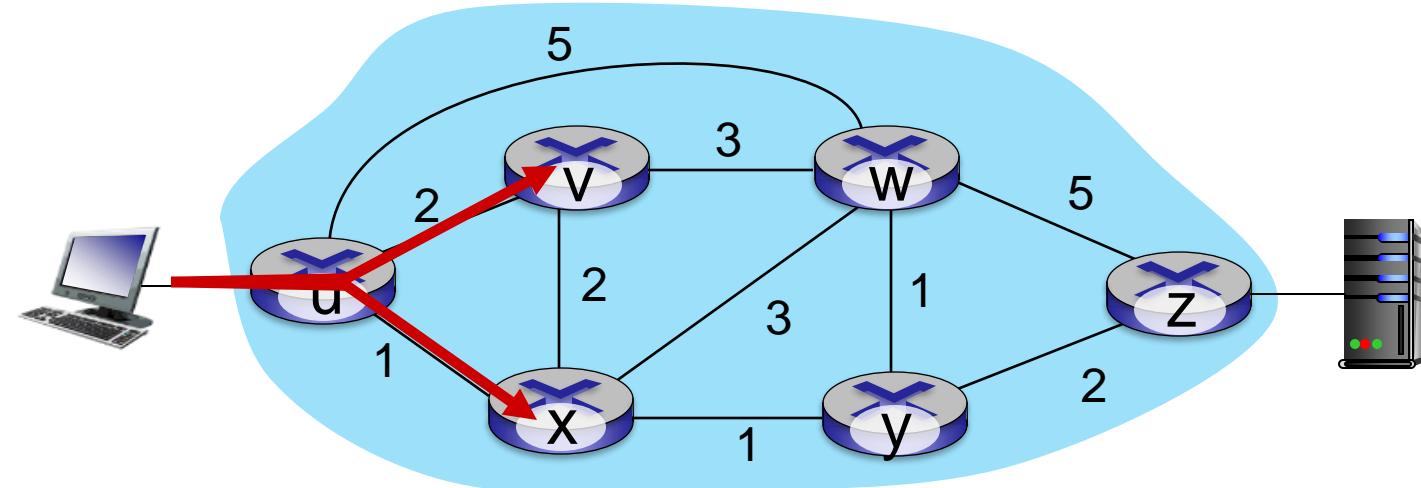


P: E se a operadora de rede quiser que o tráfego u-to-z flua ao longo de $uvwz$, em vez de $uxyz$?

R: precisa redefinir os pesos dos links para que o algoritmo de roteamento de tráfego calcule as rotas de acordo (ou precisa de um novo algoritmo de roteamento)!

Os pesos dos links são apenas "botões" de controle: não há muito controle!

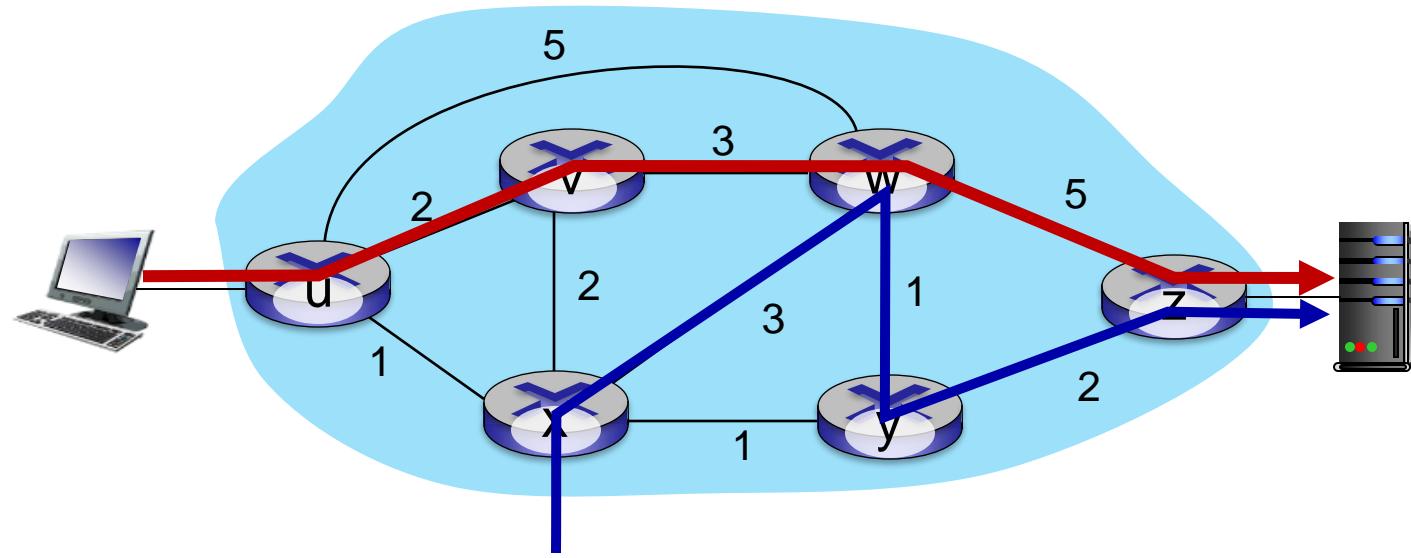
Engenharia de tráfego: difícil com o roteamento tradicional



P: E se a operadora de rede quiser dividir o tráfego u-to-z em uvwz **e** uxyz (balanceamento de carga)?

A: não pode fazer isso (ou precisa de um novo algoritmo de roteamento)

Engenharia de tráfego: difícil com o roteamento tradicional



P: E se w quiser rotear o tráfego azul e vermelho de forma diferente de w para z?

R: não é possível fazer isso (com encaminhamento baseado em destino e roteamento LS, DV)

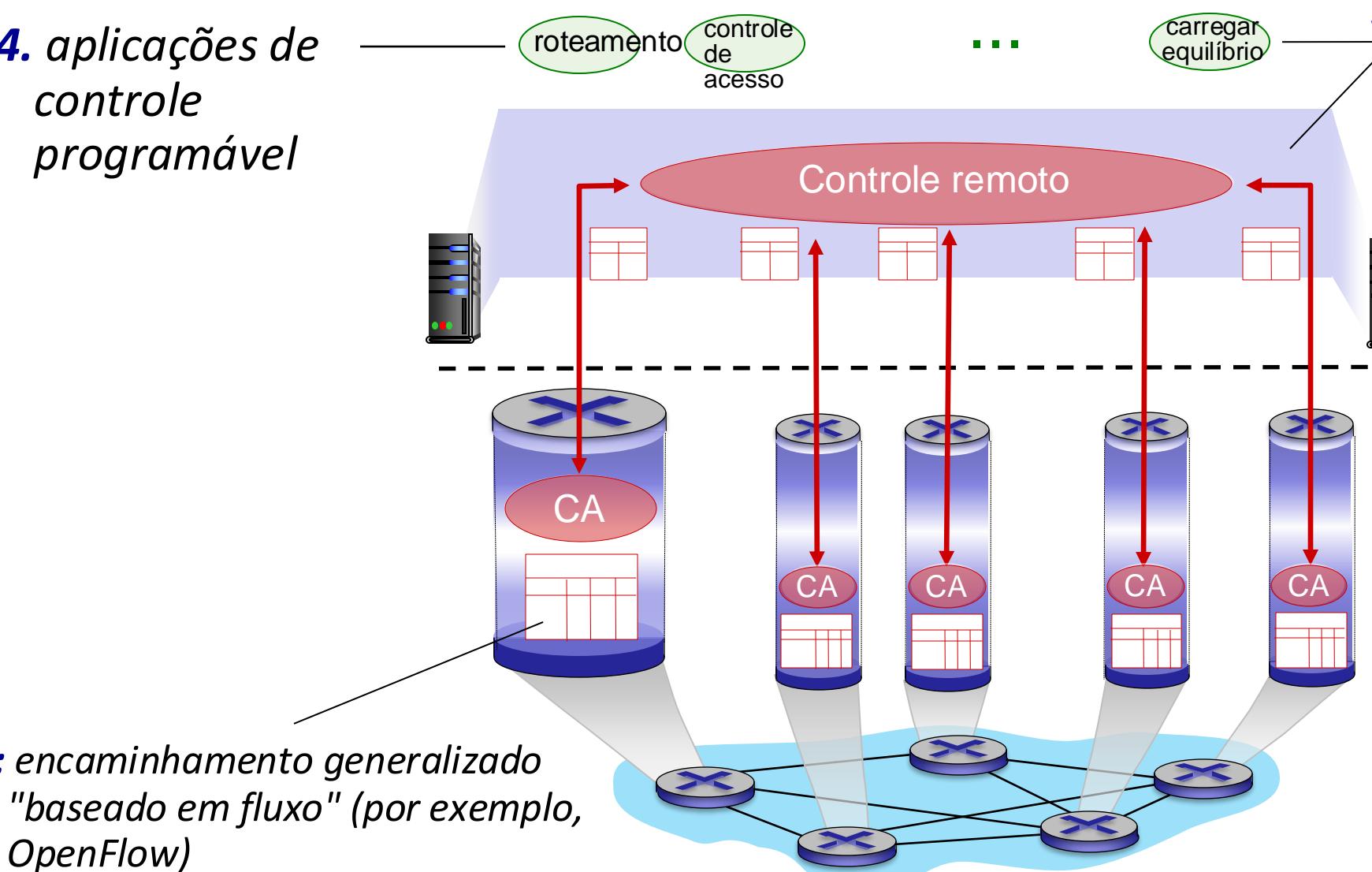
Aprendemos no Capítulo 4 que o encaminhamento generalizado e a SDN podem ser usados para obter *qualquer* roteamento desejado

Rede definida por software (SDN)

4. aplicações de controle programável

roteamento
controle de acesso

3. funções do plano de controle externas aos switches do plano de dados



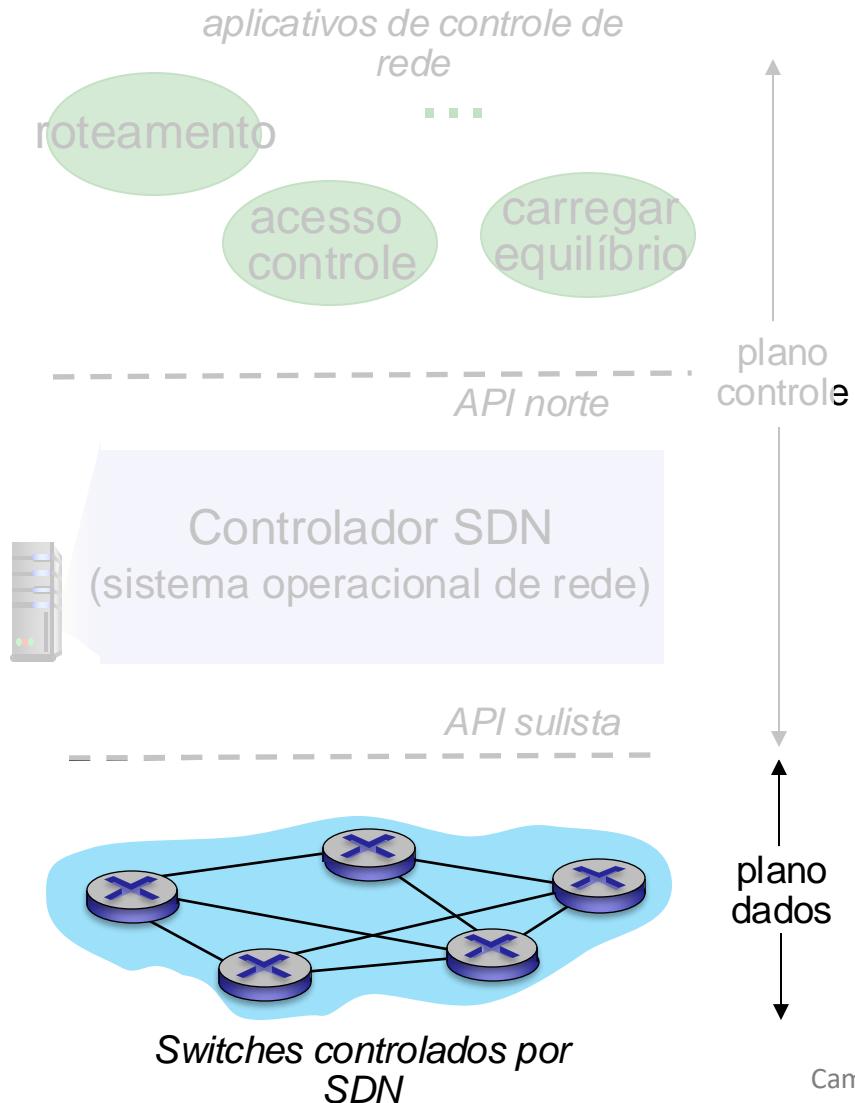
1: encaminhamento generalizado
"baseado em fluxo" (por exemplo,
OpenFlow)

2. controle,
separação do
plano de dados

Rede definida por software (SDN)

Switches de plano de dados:

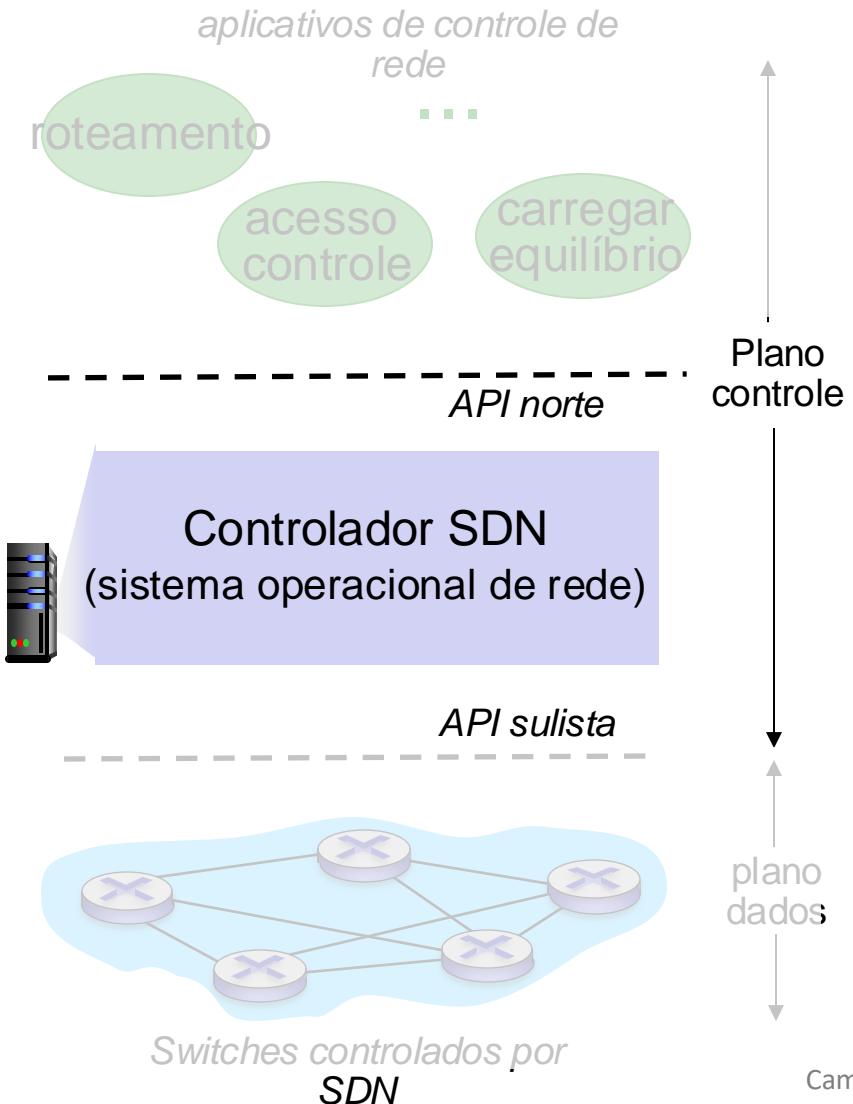
- switches rápidos, simples e de commodity que implementam o encaminhamento generalizado do plano de dados (Seção 4.4) em hardware
- tabela de fluxo (encaminhamento) computada, instalada sob a supervisão do controlador
- API para controle de switch baseado em tabela (por exemplo, OpenFlow)
 - define o que é controlável e o que não é
- protocolo para comunicação com o controlador (por exemplo, OpenFlow)



Rede definida por software (SDN)

Controlador SDN (sistema operacional de rede):

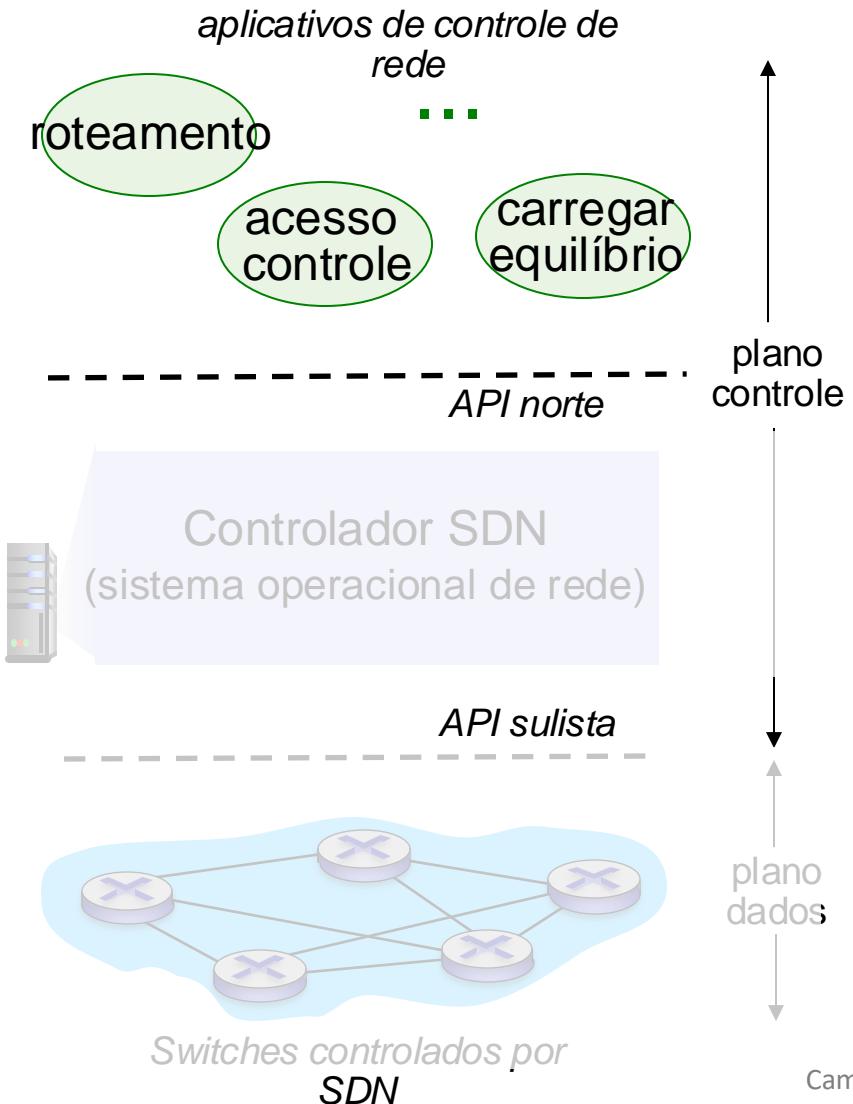
- manter informações sobre o estado da rede
- interage com os aplicativos de controle de rede "acima" por meio da API northbound
- interage com os comutadores de rede "abaixo" por meio da API sulista
- implementado como um sistema distribuído para desempenho, escalabilidade, tolerância a falhas, robustez



Rede definida por software (SDN)

aplicativos de controle de rede:

- "Cérebros" de controle: implementar funções de controle usando serviços de nível inferior, API fornecida pelo controlador SDN
- *desagregado*: pode ser fornecido por 3rd parte: diferente do fornecedor de roteamento ou do controlador SDN

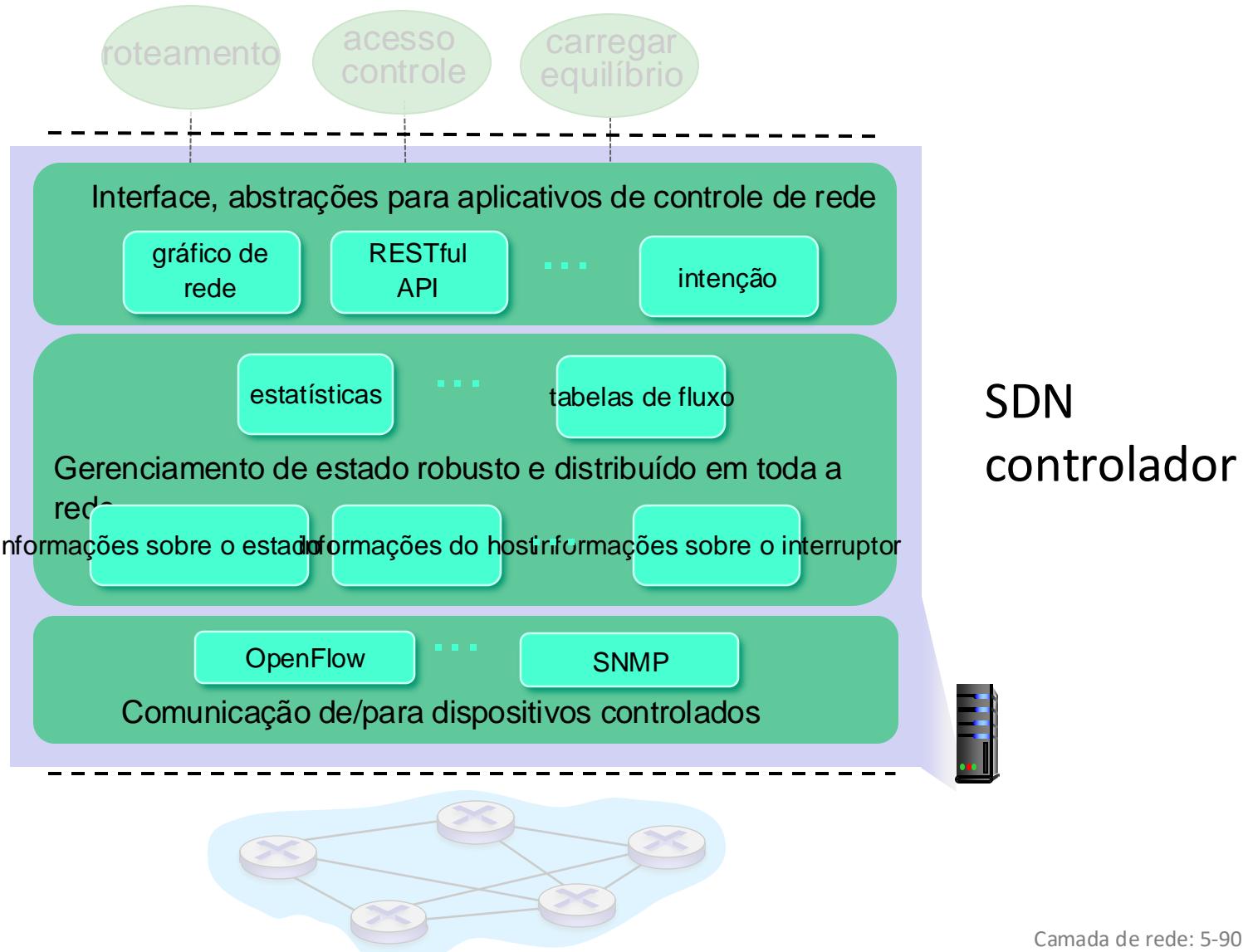


Componentes do controlador SDN

camada de interface para aplicativos de controle de rede: API de abstrações

Gerenciamento do estado de toda a rede: estado dos links, switches e serviços da rede: um *banco de dados distribuído*

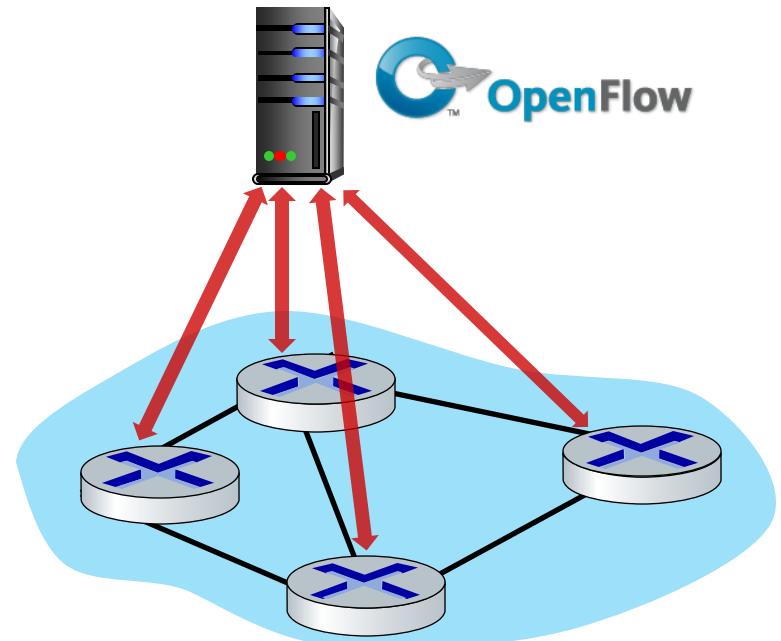
comunicação: comunicação entre o controlador SDN e os switches controlados



Protocolo OpenFlow

- opera entre o controlador, o interruptor
- TCP usado para trocar mensagens
 - criptografia opcional
- três classes de mensagens OpenFlow:
 - controlador para comutador
 - assíncrono (alternar para o controlador)
 - simétrico (diversos)
- diferente da API OpenFlow
 - API usada para especificar ações de encaminhamento generalizadas

Controlador OpenFlow

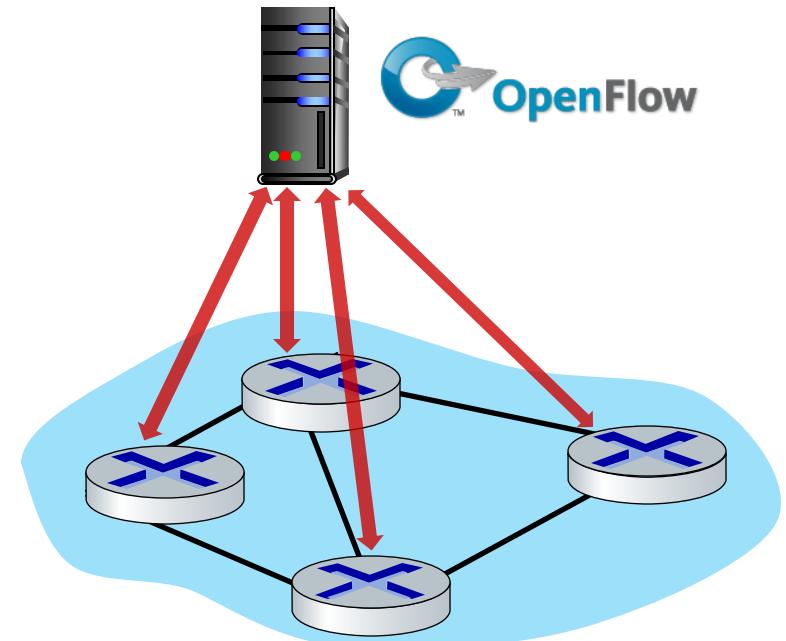


OpenFlow: mensagens de controlador para comutador

Principais mensagens do controlador para o comutador

- *recursos*: o controlador consulta os recursos do switch, as respostas do switch
- *configure*: o controlador consulta/define os parâmetros de configuração do switch
- *modify-state*: adiciona, exclui e modifica entradas de fluxo nas tabelas OpenFlow
- *packet-out*: o controlador pode enviar esse pacote para fora de uma porta de switch específica

Controlador OpenFlow



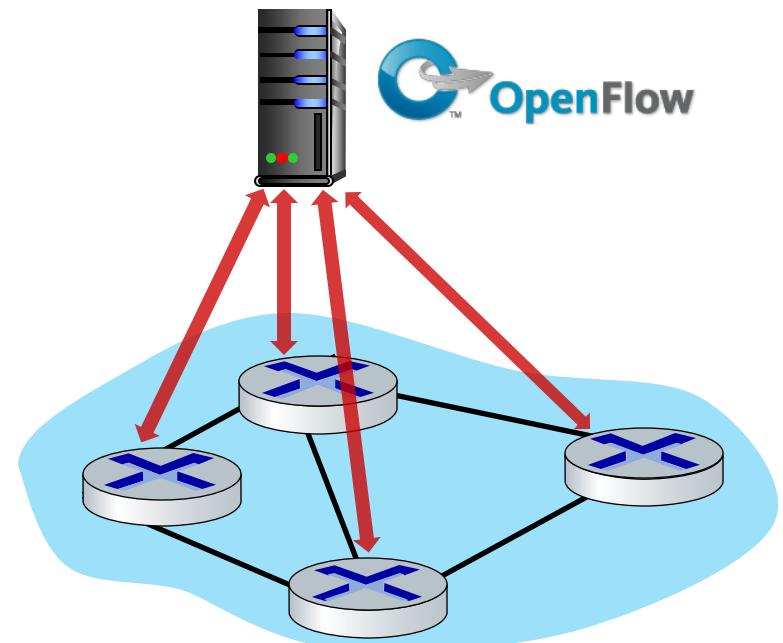
OpenFlow: mensagens do switch para o controlador

Mensagens-chave do interruptor para o controlador

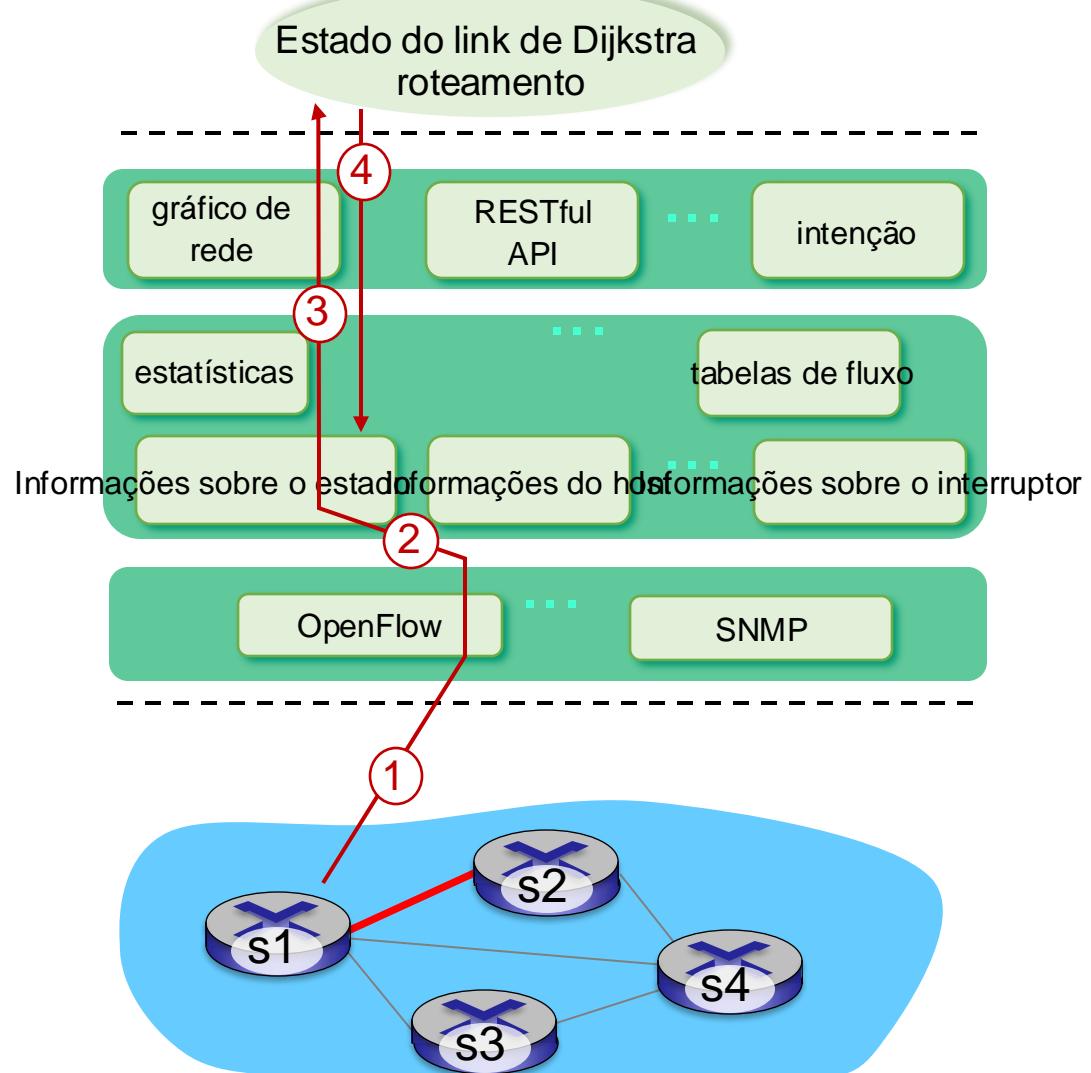
- *packet-in*: transfere o pacote (e seu controle) para o controlador. Veja a mensagem packet-out do controlador
- *flow-removed*: entrada da tabela de fluxo excluída no switch
- *status da porta*: informa o controlador sobre uma alteração em uma porta.

Felizmente, os operadores de rede não "programam" switches criando/enviando mensagens OpenFlow diretamente. Em vez disso, usam uma abstração de nível superior no controlador

Controlador OpenFlow

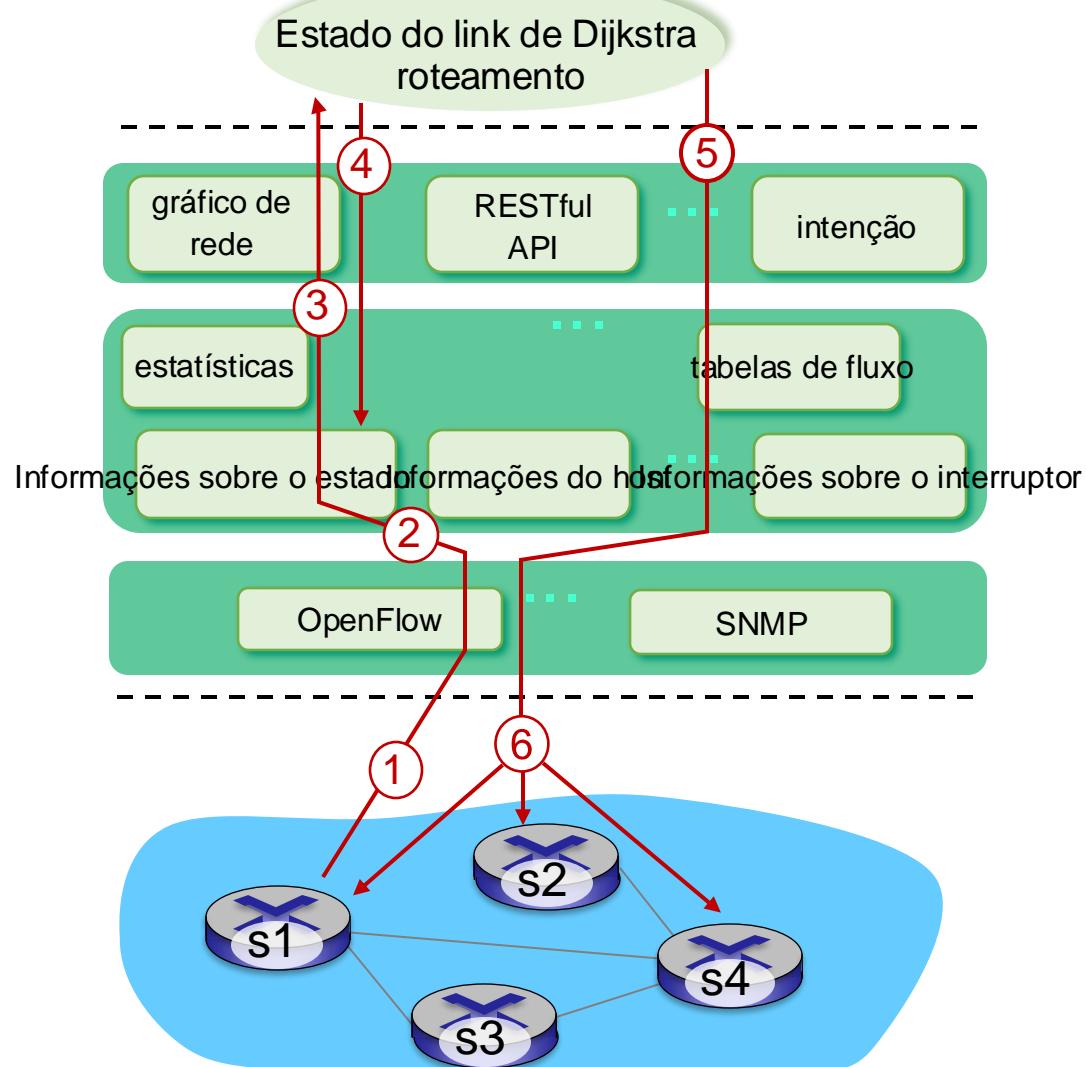


SDN: exemplo de interação do plano de controle/dados



- ① S1, com falha de link, usa a mensagem de status da porta OpenFlow para notificar o controlador
- ② O controlador SDN recebe a mensagem OpenFlow e atualiza as informações de status do link
- ③ O aplicativo do algoritmo de roteamento de Dijkstra foi registrado anteriormente para ser chamado sempre que o status do link for alterado. Ele é chamado.
- ④ O algoritmo de roteamento de Dijkstra acessa informações do gráfico da rede, informações do estado do link no controlador e calcula novas rotas

SDN: exemplo de interação do plano de controle/dados

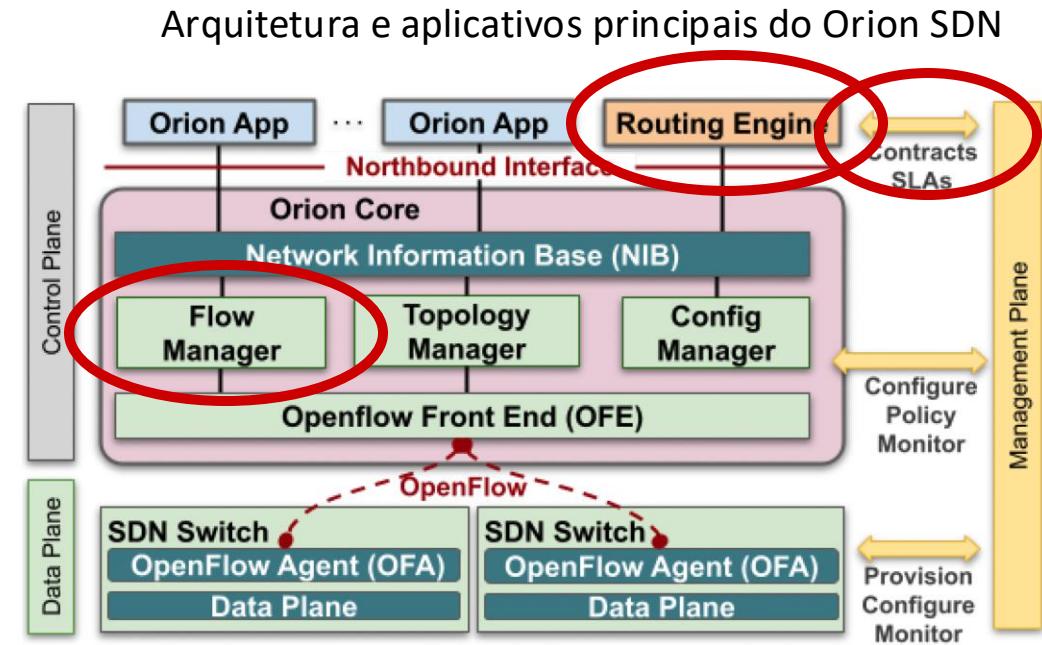


- ⑤ O aplicativo de roteamento de estado de link interage com o componente de computação de tabela de fluxo no controlador SDN, que calcula as novas tabelas de fluxo necessárias
- ⑥ O controlador usa o OpenFlow para instalar novas tabelas nos switches que precisam ser atualizados

Plano de controle SDN do Google ORION

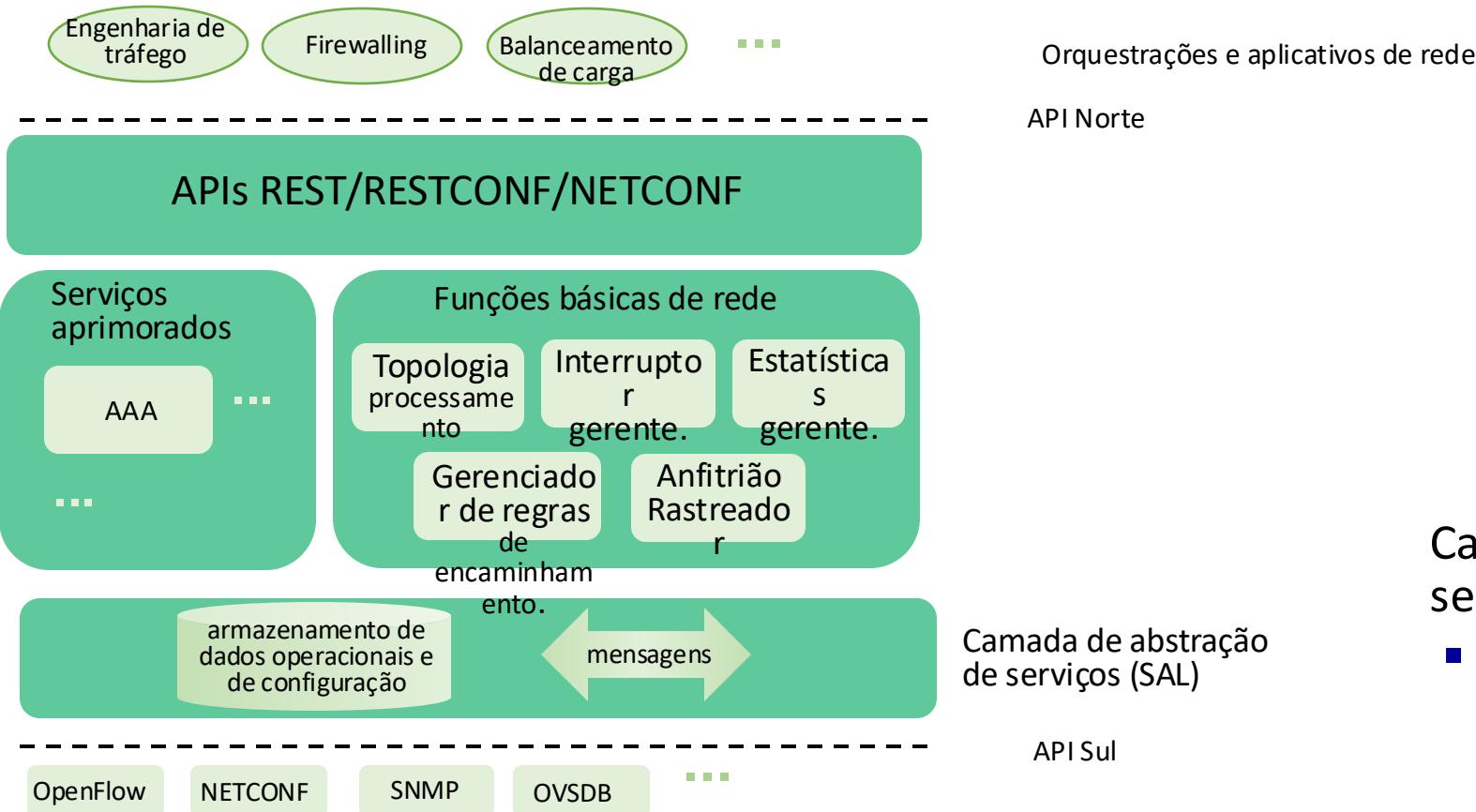
ORION: plano de controle SDN do Google (*NSDI'21*): plano de controle para as redes de data center (Jupiter) e de área ampla (B4) do Google

- **Roteamento** (intradomínio, iBGP), engenharia de tráfego: implementado em *aplicativos* sobre o núcleo do ORION
- controles **baseados em fluxo de borda a borda** (por exemplo, agendamento CoFlow) para atender aos SLAs de contrato
- **gerenciamento:** microsserviços distribuídos pub-sub no núcleo do Orion, OpenFlow para sinalização/monitoramento de switches



Observação: a ORION fornece serviços *intradomínio* dentro da rede do Google

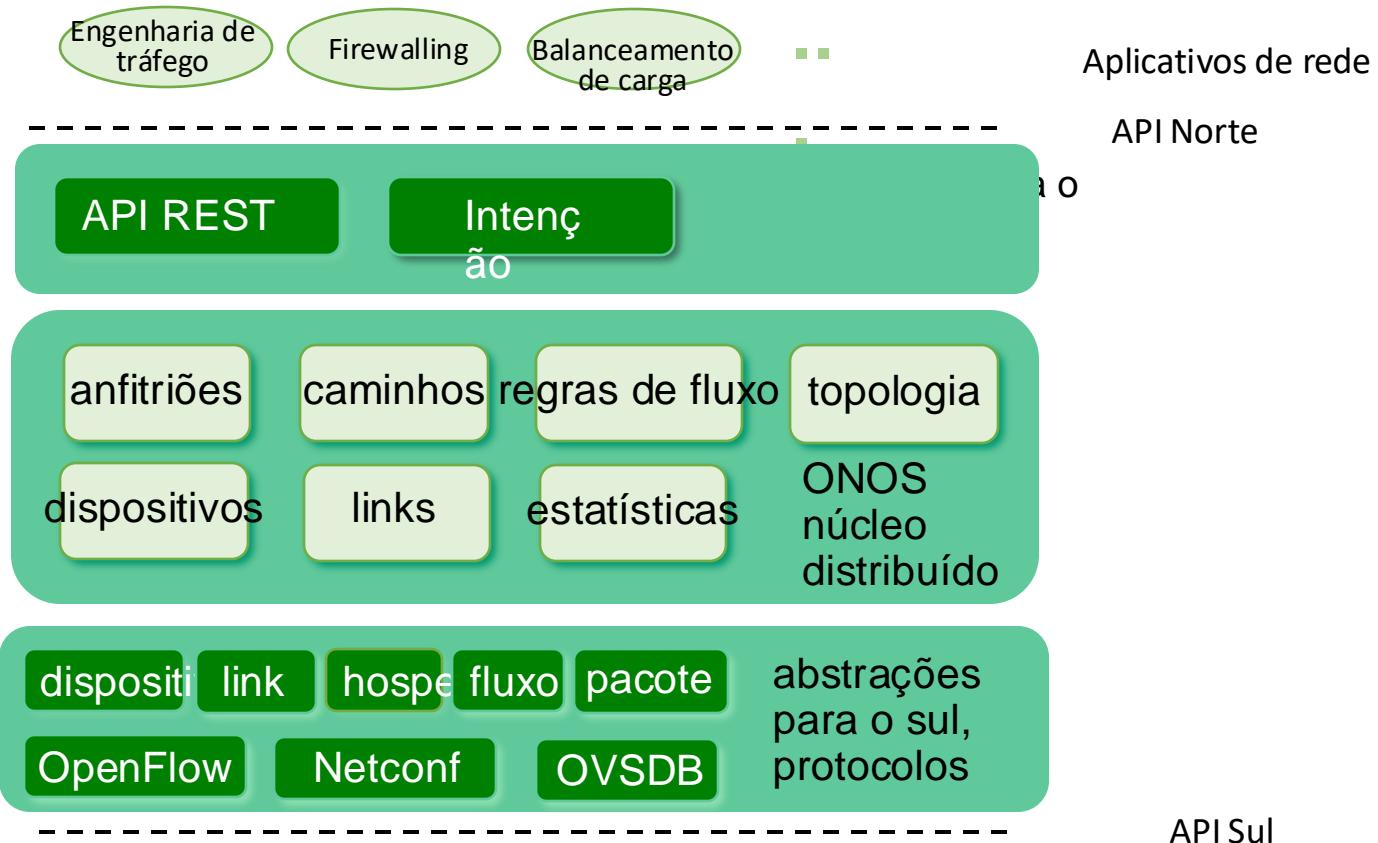
Controlador OpenDaylight (ODL)



Camada de abstração de serviço:

- interconecta aplicativos e serviços internos e externos

Controlador ONOS



- aplicativos de controle separados do controlador
- Estrutura de intenção: especificação de alto nível do serviço: o que em vez de como
- ênfase considerável no núcleo distribuído: confiabilidade do serviço, dimensionamento do desempenho da replicação

SDN: desafios selecionados

- Fortalecimento do plano de controle: sistema distribuído seguro, confiável, escalável em termos de desempenho
 - robustez a falhas: aproveite a teoria forte do sistema distribuído confiável para o plano de controle
 - confiabilidade, segurança: "incorporados" desde o primeiro dia?
- redes, protocolos que atendem aos requisitos específicos da missão
 - Por exemplo, em tempo real, ultraconfiável, ultra-seguro
- Dimensionamento da Internet: além de um único AS
- SDN é fundamental em redes celulares 5G

SDN e o futuro dos protocolos de rede tradicionais

- Tabelas de encaminhamento computadas por SDN versus tabelas de encaminhamento de roteador-computador:
 - apenas um exemplo de computação logicamente centralizada versus computação de protocolo
- pode-se imaginar um controle de congestionamento computado por SDN:
 - o controlador define as taxas do remetente com base nos níveis de congestionamento informados pelo roteador (ao controlador)



Como a implementação da funcionalidade de rede (SDN versus protocolos) evoluirá?



Camada de rede: Roteiro do "plano de controle"

- introdução
- protocolos de roteamento
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- Plano de controle SDN
- **Protocolo de Mensagens de Controle da Internet**



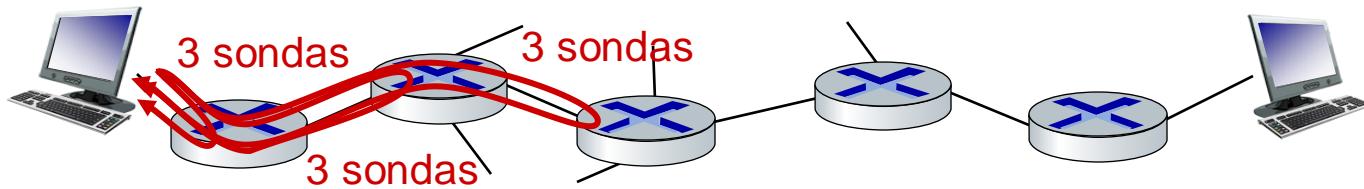
- gerenciamento de rede, configuração
 - SNMP
 - NETCONF/YANG

ICMP: protocolo de mensagens de controle da Internet

- usado por hosts e roteadores para comunicar informações em nível de rede
 - relatório de erros: host, rede, porta e protocolo inacessíveis
 - solicitação/resposta de eco (usado pelo ping)
- camada de rede "acima" do IP:
 - Mensagens ICMP transportadas em datagramas IP
- *Mensagem ICMP:* tipo, código mais os primeiros 8 bytes do datagrama IP que está causando o erro

Tipo	Descrição do código
0	0 resposta de eco (ping)
3	0 rede de destino inacessível
3	1 dest host inacessível
3	2 dest protocol unreachable
3	3 porta de destino inacessível
3	6 rede de destino desconhecida
3	7 host de destino desconhecido
4	0 resfriamento da fonte (congestionamento controle - não utilizado)
8	0 Solicitação de eco (ping)
9	0 anúncio de rota
10	0 descoberta de roteador
11	0 TTL expirado
12	0 cabeçalho IP ruim

Traceroute e ICMP



- a origem envia conjuntos de segmentos UDP para o destino
 - O conjunto 1st tem TTL =1, o conjunto 2nd tem TTL=2, etc.
- O datagrama do *enésimo* conjunto chega ao enésimo roteador:
 - O roteador descarta o datagrama e envia uma mensagem ICMP de origem (tipo 11, código 0)
 - A mensagem ICMP possivelmente inclui o nome do roteador e o endereço IP
- quando a mensagem ICMP chega à origem: registrar RTTs

critérios de interrupção:

- O segmento UDP eventualmente chega ao host de destino
- O destino retorna a mensagem ICMP "port unreachable" (porta inacessível) (tipo 3, código 3)
- paradas de origem

Camada de rede: Roteiro do "plano de controle"

- introdução
- protocolos de roteamento
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- Plano de controle SDN
- Protocolo de Mensagens de Controle da Internet



- gerenciamento de rede, configuração
 - SNMP
 - NETCONF/YANG

O que é gerenciamento de rede?

- sistemas autônomos (também conhecidos como "rede"): milhares de componentes de hardware/software que interagem
- outros sistemas complexos que exigem monitoramento, configuração e controle:
 - avião a jato, usina nuclear, outros?

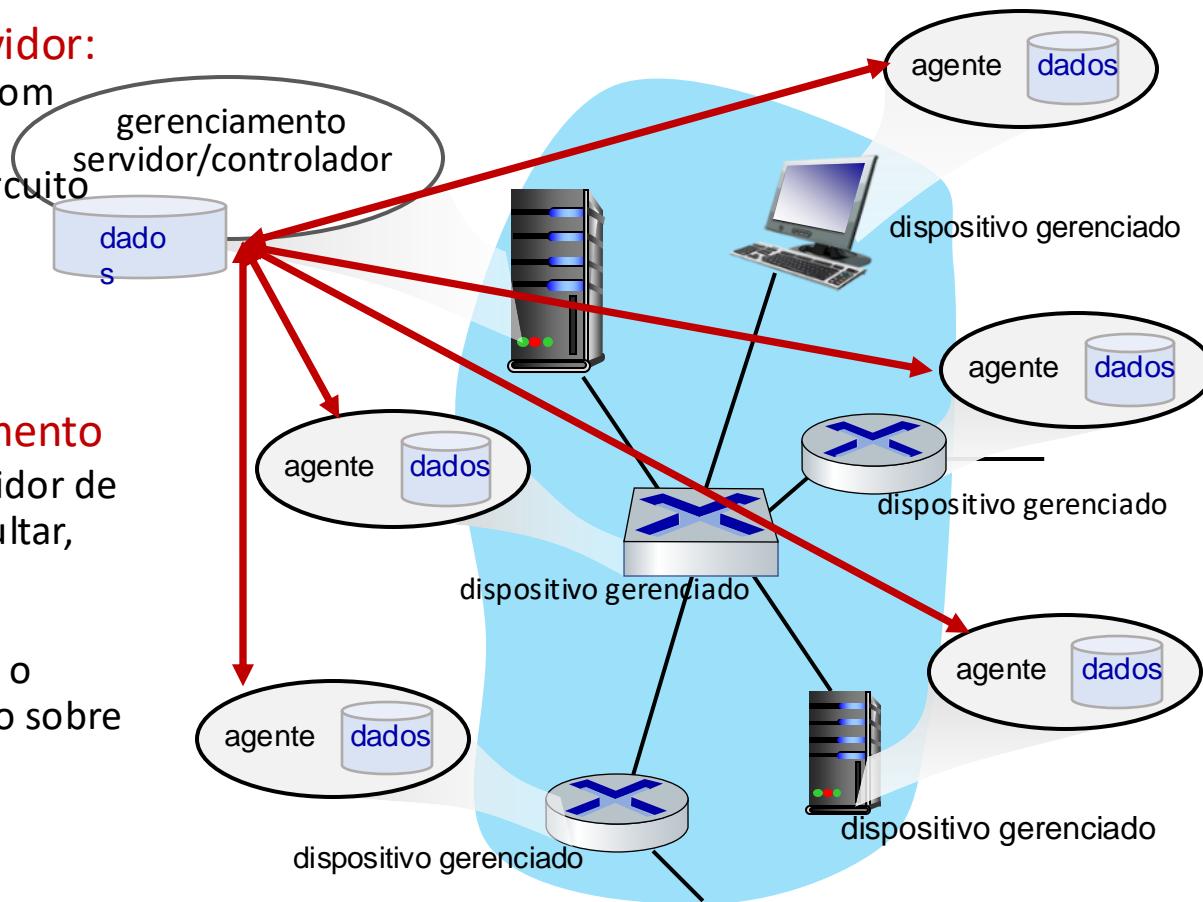


"O gerenciamento de rede inclui a implantação, a integração e coordenação do hardware, software e recursos humanos elementos para monitorar, testar, pesquisar, configurar, analisar e avaliar, e controlar os recursos da rede e dos elementos para atender às necessidades em tempo real, desempenho operacional e qualidade de serviço requisitos a um custo razoável".

Componentes do gerenciamento de rede

Gerenciamento de servidor:
aplicativo, normalmente com
rede
gerentes (humanos) no circuito

Protocolo de gerenciamento de rede: usado pelo servidor de gerenciamento para consultar, configurar e gerenciar o dispositivo; usado pelos dispositivos para informar o servidor de gerenciamento sobre dados e eventos.



Dispositivo gerenciado:
equipamento com componentes de hardware e software gerenciáveis e configuráveis

Dados: dados de configuração do "estado" do dispositivo, dados operacionais, estatísticas do dispositivo

Abordagens de gerenciamento das operadoras de rede

CLI (Command Line Interface)

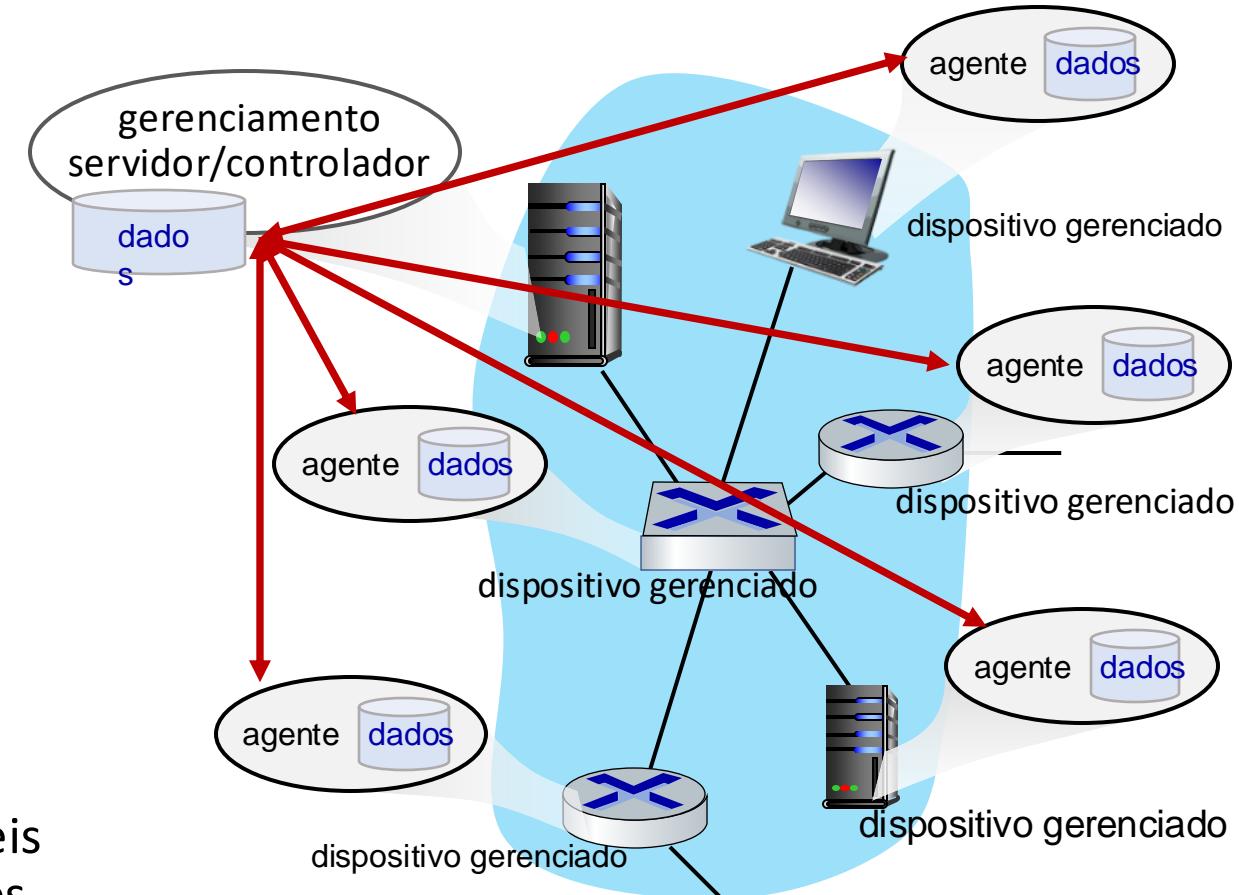
- problemas do operador (tipos, scripts) diretamente para dispositivos individuais (por exemplo, vis ssh)

SNMP/MIB

- O operador consulta/configura os dados dos dispositivos (MIB) usando o SNMP (Simple Network Management Protocol)

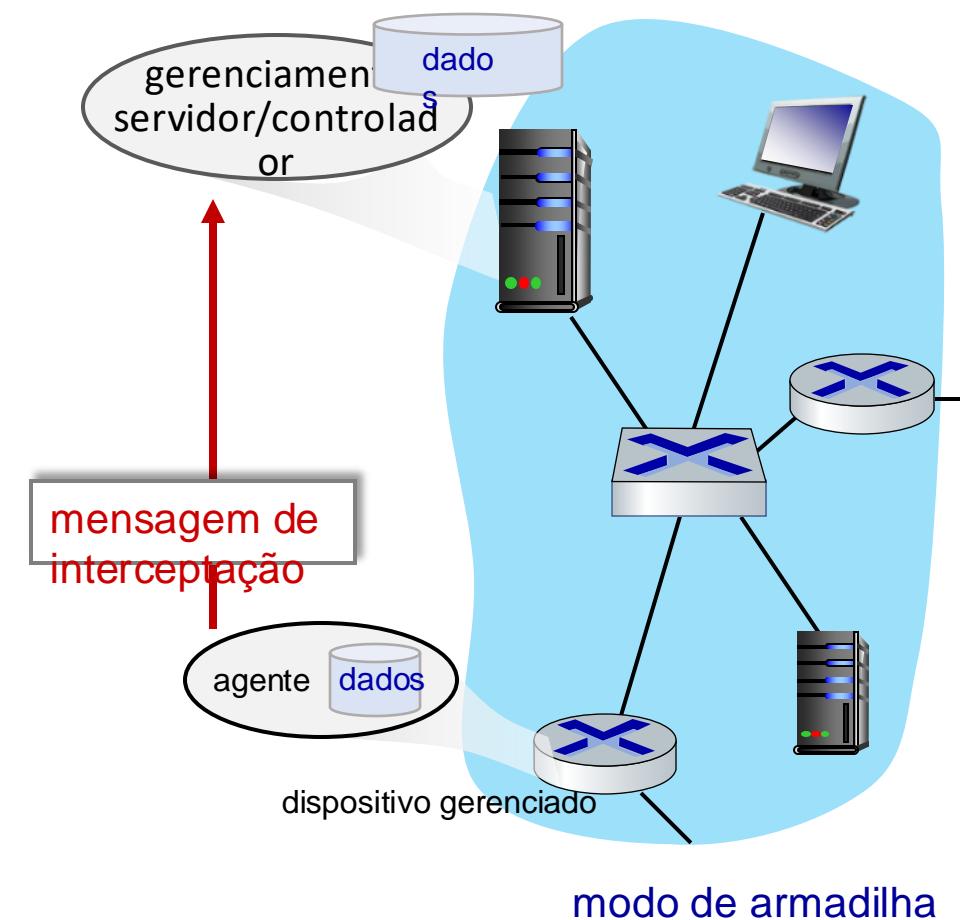
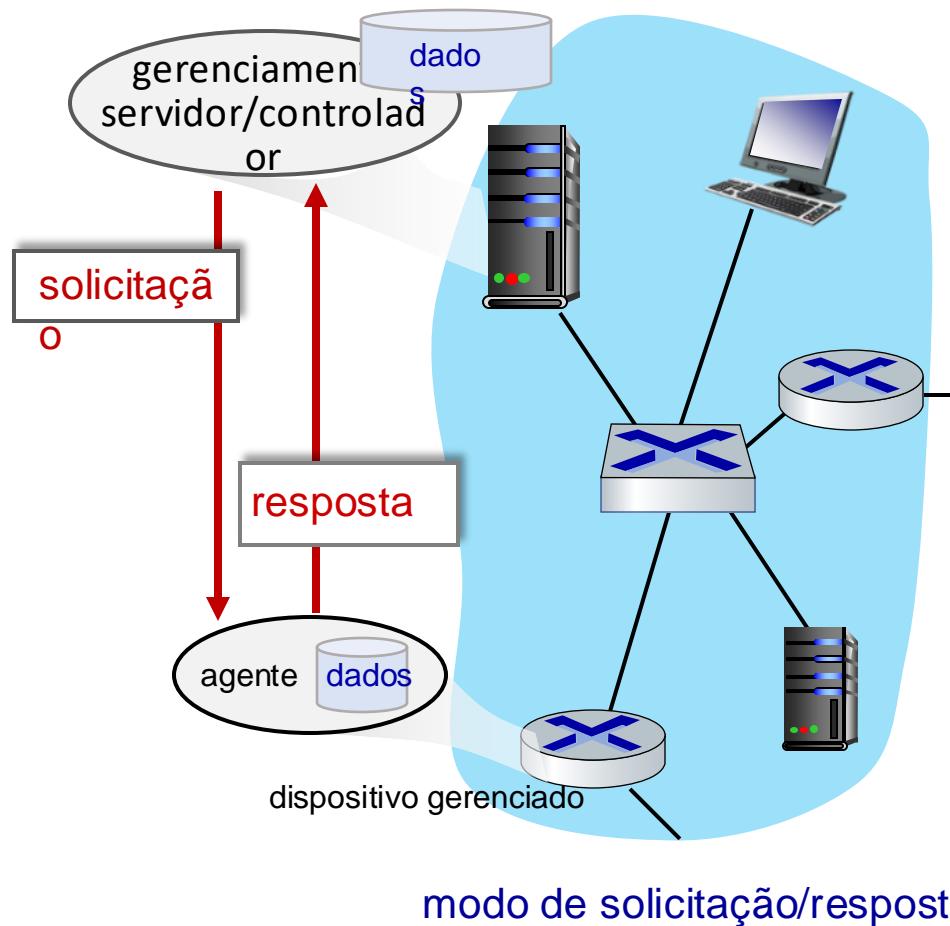
NETCONF/YANG

- mais abstrato, em toda a rede, holístico
- ênfase no gerenciamento de configuração de vários dispositivos.
- YANG: linguagem de modelagem de dados
- NETCONF: comunicar ações/dados compatíveis com YANG de/para/entre dispositivos remotos



Protocolo SNMP

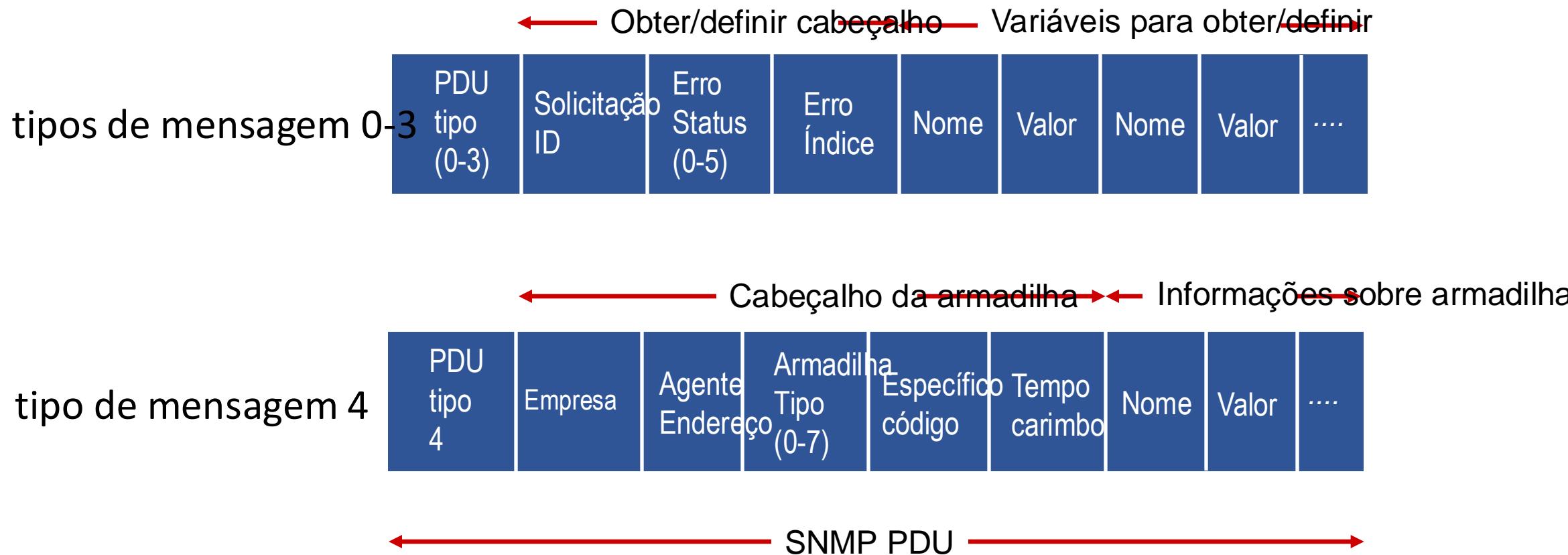
Duas maneiras de transmitir informações do MIB,
comandos:



Protocolo SNMP: tipos de mensagens

Tipo de mensagem	Função
GetRequest GetNextRequest GetBulkRequest	gerente para agente: "obtenha meus dados" (instância de dados, próximo dado na lista, bloco de dados).
SetRequest	gerente-para-agente: definir valor MIB
Resposta	Agente para gerente: valor, resposta à solicitação
Armadilha	Agente para gerente: informar o gerente de evento excepcional

Protocolo SNMP: formatos de mensagem



SNMP: Base de informações de gerenciamento (MIB)

- dados operacionais (e alguns dados de configuração) do dispositivo gerenciado
- reunidos no **módulo MIB** do dispositivo
 - 400 módulos MIB definidos em RFCs; muitos outros MIBs específicos de fornecedores
- **Estrutura de informações gerenciais (SMI):** linguagem de definição de dados
- Exemplo de variáveis MIB para o protocolo UDP:



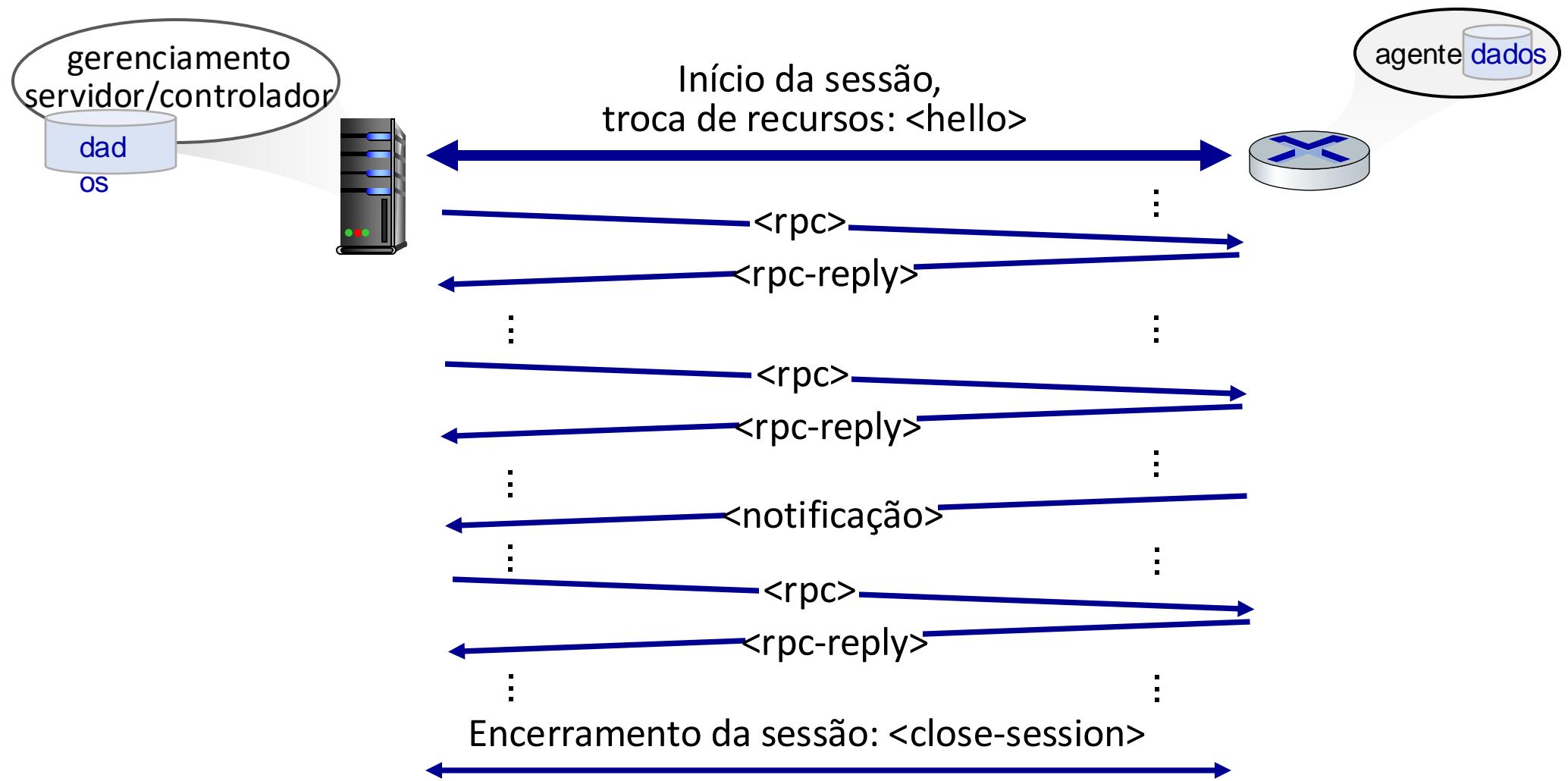
ID do objeto Nome Tipo Comentários

1.3.6.1.2.1.7.1	UDPIInDatagrams	Contador de 32 bits do total de datagramas entregues
1.3.6.1.2.1.7.2	UDPNoPorts	Contador de 32 bits # datagramas não entregues (nenhum aplicativo na porta)
1.3.6.1.2.1.7.3	UDPIInErrors	Contador de 32 bits # datagramas não entregues (todos os outros motivos)
1.3.6.1.2.1.7.4	UDPOutDatagrams	Contador de 32 bits do total de datagramas enviados
1.3.6.1.2.1.7.5	udpTable	SEQUENCE uma entrada para cada porta em uso no momento

Visão geral do NETCONF

- **objetivo:** gerenciar/configurar ativamente os dispositivos em toda a rede
- opera entre o servidor de gerenciamento e os dispositivos de rede gerenciados
 - ações: recuperar, definir, modificar, ativar configurações
 - ações **de confirmação atômica** em vários dispositivos
 - consultar dados operacionais e estatísticas
 - assinar notificações de dispositivos
- paradigma de chamada de procedimento remoto (RPC)
 - Mensagens do protocolo NETCONF codificadas em XML
 - trocados por um protocolo de transporte seguro e confiável (por exemplo, TLS)

Inicialização, troca e fechamento do NETCONF



Operações NETCONF selecionadas

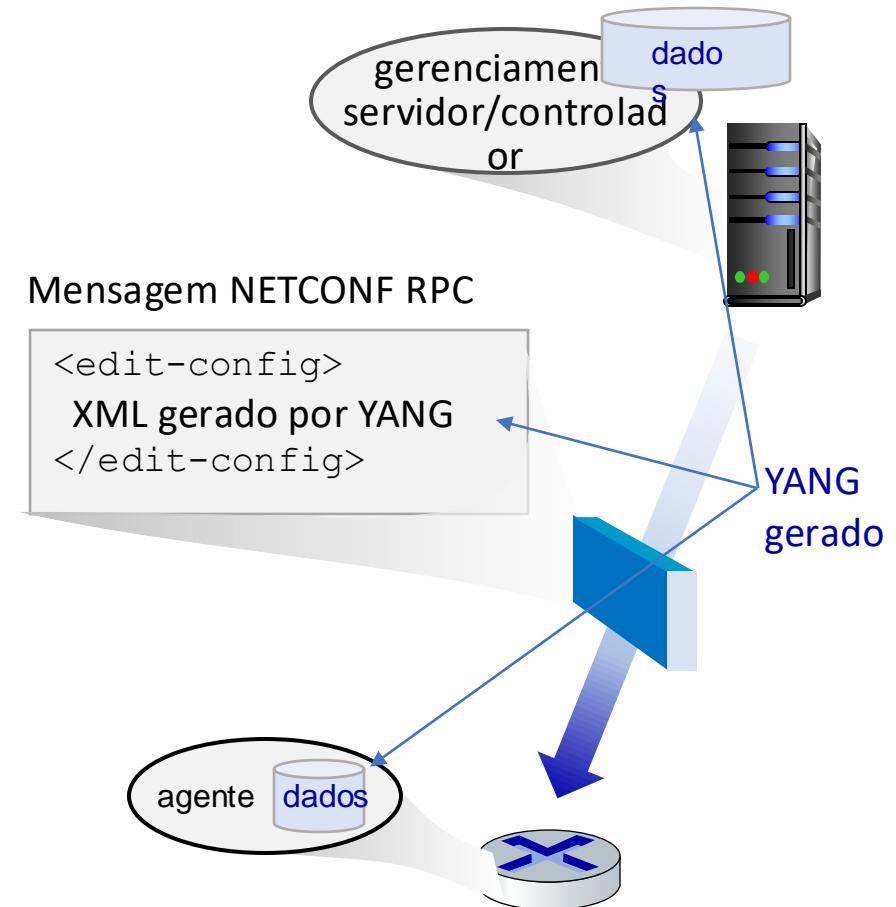
Descrição da operação do NETCONF
<get-config> Recupera toda ou parte de uma determinada configuração. Um dispositivo pode ter várias configurações.
<get> Recupera todos ou parte dos dados do estado de configuração e do estado operacional.
<edit-config> Altera a configuração especificada (possivelmente em execução) no dispositivo gerenciado. O dispositivo gerenciado <rpc-reply> contém <ok> ou <rpcerror> com reversão.
<lock> , <unlock> Bloquear (desbloquear) o armazenamento de dados de configuração no dispositivo gerenciado (para bloquear comandos NETCONF, SNMP ou CLIs de outras fontes).
<create-subscription> , Ativar assinatura de notificação de eventos do dispositivo gerenciado <notificação>

Exemplo de mensagem NETCONF RPC

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <rpc message-id="101" ID da mensagem da nota
03   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
04     <edit-config> alterar uma configuração
05       <target>
06         <running/> alterar a configuração em execução
07       </target>
08     <config>
09       <top xmlns="http://example.com/schema/
10         1.2/config">
11           <interface>
12             <name>Ethernet0/0</name> alterar o MTU da interface Ethernet 0/0 para 1500
13             <mtu>1500</mtu>
14           </interface>
15         </top>
16       </config>
17     </edit-config>
18   </rpc>
```

YANG

- linguagem de modelagem de dados usada para especificar a estrutura, a sintaxe e a semântica dos dados de gerenciamento de rede NETCONF
 - tipos de dados incorporados, como SMI
- Documento XML que descreve o dispositivo, os recursos podem ser gerados a partir da descrição YANG
- pode expressar restrições entre os dados que devem ser atendidas por uma configuração NETCONF válida
 - Garantir que as configurações do NETCONF satisfaçam a correção e as restrições de consistência



Camada de rede: Resumo

Aprendemos muito!

- abordagens para o plano de controle da rede
 - controle por roteador (tradicional)
 - controle logicamente centralizado (rede definida por software)
- algoritmos de roteamento tradicionais
 - implementação na Internet: OSPF , BGP
- Controladores SDN
 - implementação na prática: ODL, ONOS
- Protocolo de Mensagens de Controle da Internet
- gerenciamento de rede

Próxima parada: camada de links!

Camada de rede, plano de controle: Pronto!

- introdução
- protocolos de roteamento
 - estado do link
 - vetor de distância
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- Plano de controle SDN
- Protocolo de Mensagens de Controle da Internet



- gerenciamento de rede, configuração
 - SNMP
 - NETCONF/YANG

Slides adicionais do Capítulo 5

Vetor de distância: outro exemplo

$D_x ()$

custo para		
	x	y
x	0	2
y	∞	∞
z	∞	∞

$D_y ()$

custo para		
	x	y
x	∞	∞
y	2	0
z	∞	∞

$D_z ()$

custo para		
	x	y
x	∞	∞
y	∞	∞
z	7	1

$custo para$

custo para		
	x	y
x	0	2
y	2	0
z	7	1

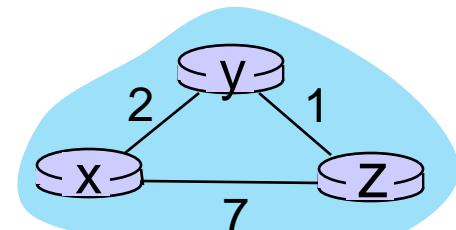
$D_x (y) = \min\{c_{x,y} + D_y (y), c_{x,z} + D_z (y)\}$

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

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

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

tempo



Vetor de distância: outro exemplo

