

Redes de Computadores

A Camada de Rede: Plano de Controle

Fabricio Breve

www.fabriciobreve.com

Nota: a maioria dos slides dessa apresentação são traduzidos ou adaptados dos slides disponibilizados gratuitamente pelos autores do livro [KUROSE, James F. e ROSS, Keith W. *Computer Networking: A Top-Down Approach*. 8th Edition. Pearson, 2020.](#) Todo o material pertencente aos seus respectivos autores está protegido por direito autoral.

Chapter 5

Network Layer: Control Plane

A note on the use of these PowerPoint slides:

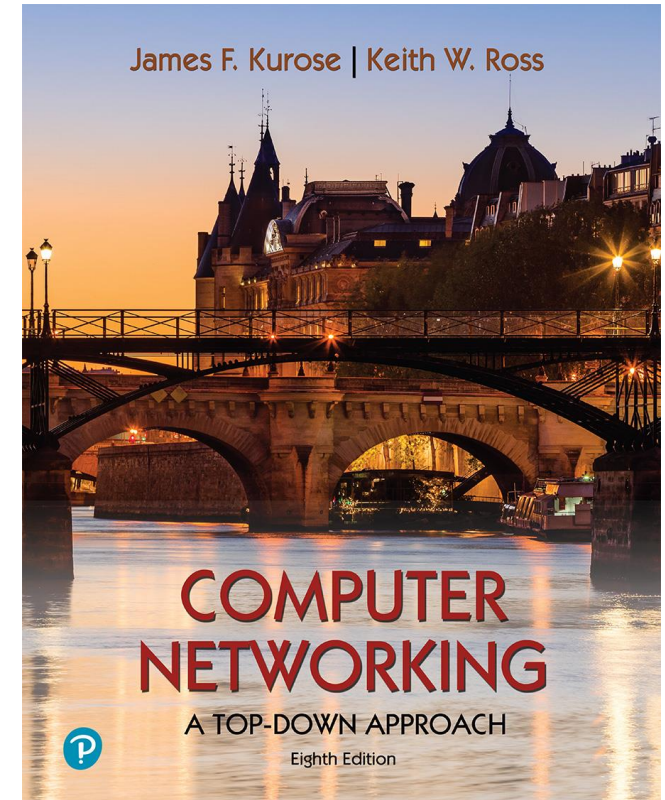
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2023
J.F Kurose and K.W. Ross, All Rights Reserved



*Computer Networking: A
Top-Down Approach*

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 de rede:
 - algoritmos de roteamento tradicionais
 - controladores SDN
 - gerenciamento e configuração de redes
- instanciamento e implementação na Internet :
 - OSPF, BGP
 - controladores OpenFlow, ODL e ONOS
 - Internet Control Message Protocol: ICMP
 - SNMP, YANG/NETCONF

Camada de rede: roteiro do “plano de controle”

- **introdução**
- protocolos de roteamento
 - estado de enlace
 - vetor de distância
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- plano de controle das SDN
- Internet Control Message Protocol



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

Funções da camada de rede

- **encaminhamento:** mover pacotes da entrada do roteador para a saída apropriada

plano de dados

- **roteamento:** determinar a rota percorrida pelos pacotes da origem ao destino

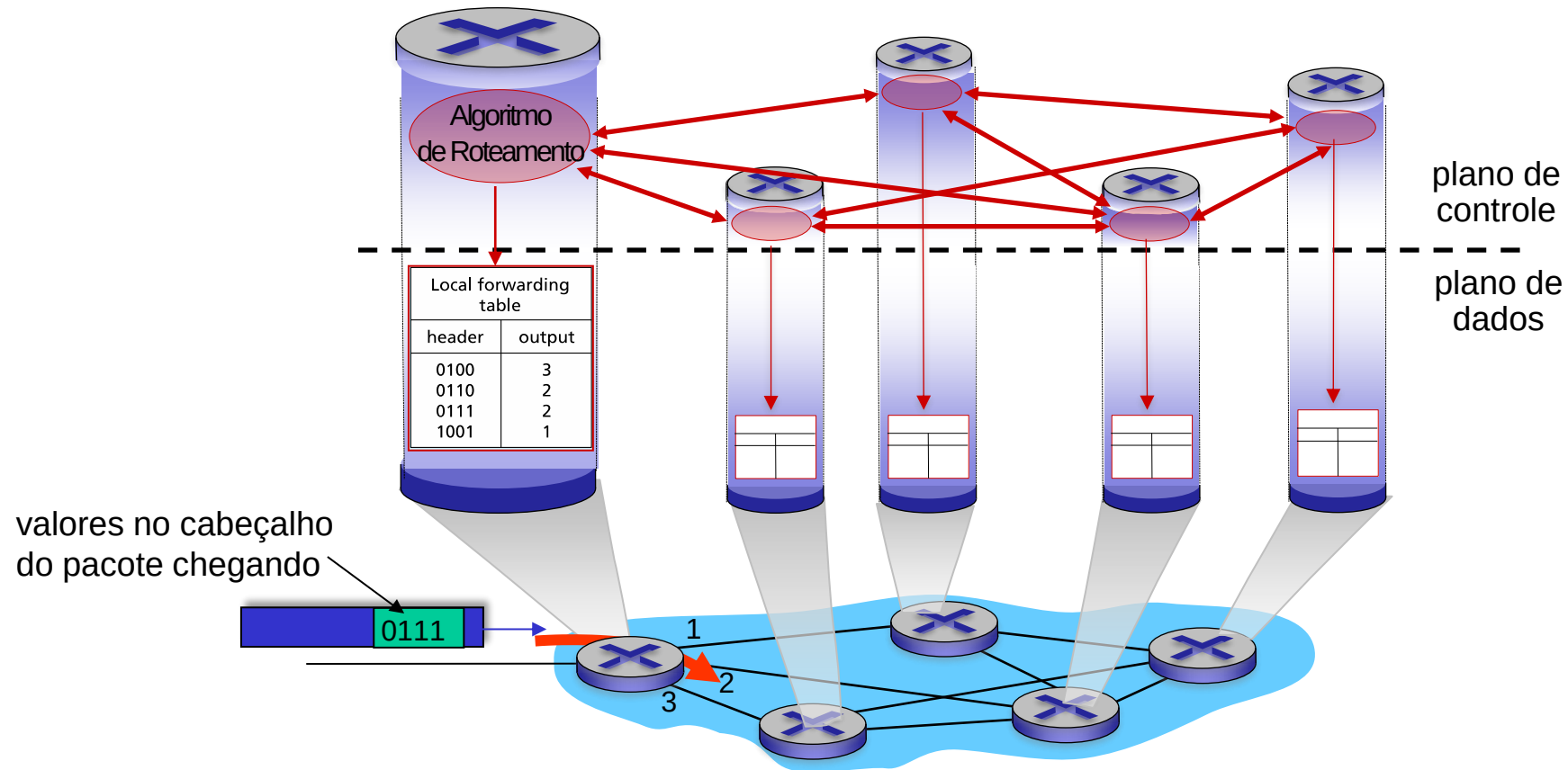
plano de controle

Duas abordagens para estruturar o plano de controle da rede:

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

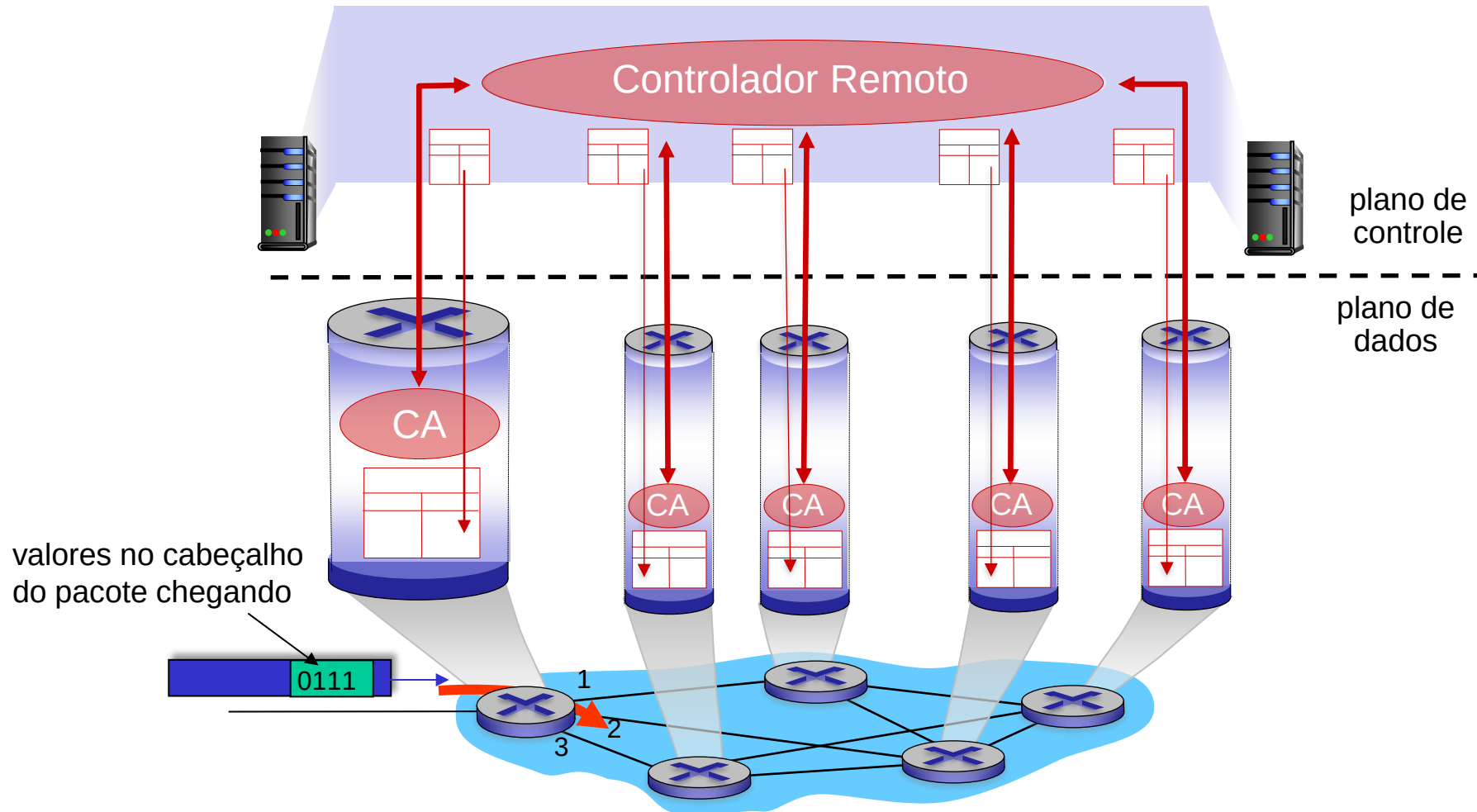
Plano de controle por roteador

Componentes individuais do algoritmo de roteamento *em todo e cada roteador* interagem no plano de controle

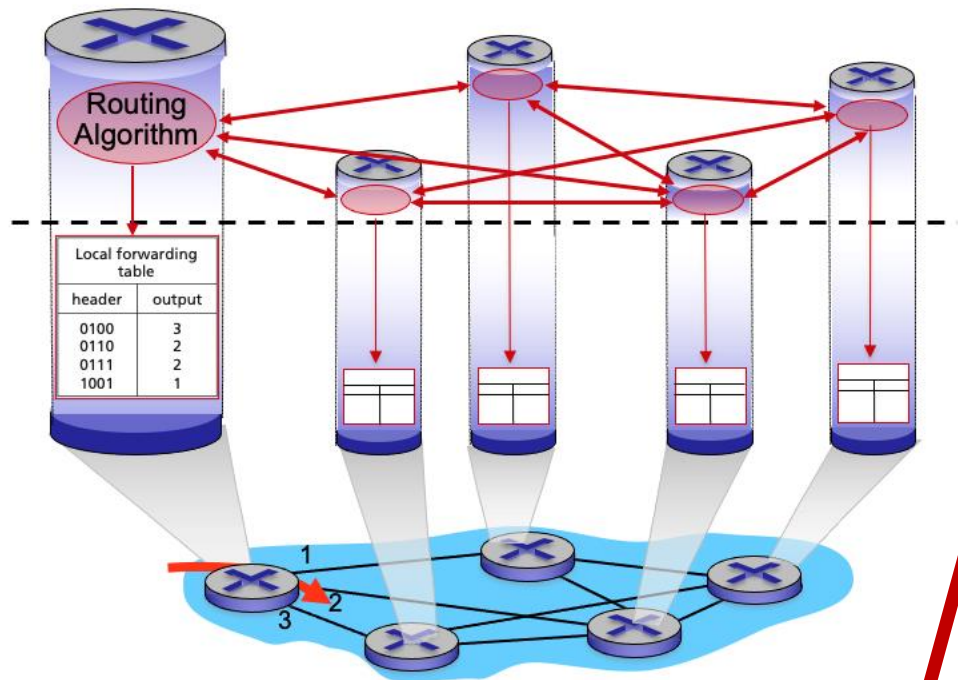


Plano de controle de Rede Definida por Software (SDN – Software Defined Network)

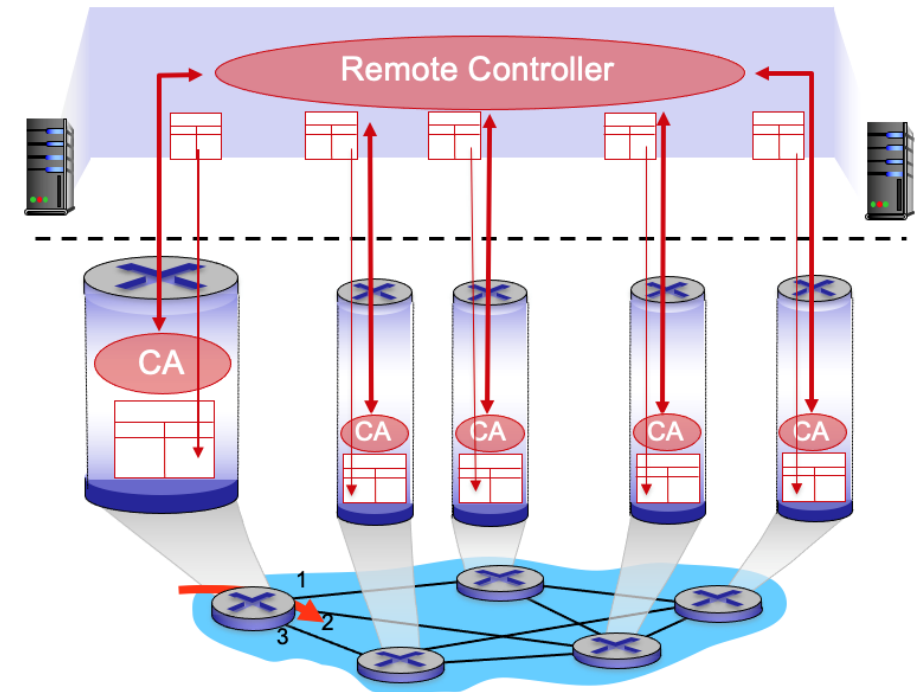
O controlador remoto calcula e instala tabelas de encaminhamento nos roteadores



Plano de controle por roteador



Plano de controle de SDN



Camada de rede: roteiro do “plano de controle”

- introdução
- protocolos de roteamento
 - estado de enlace
 - vetor de distância
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- plano de controle das SDN
- Internet Control Message Protocol

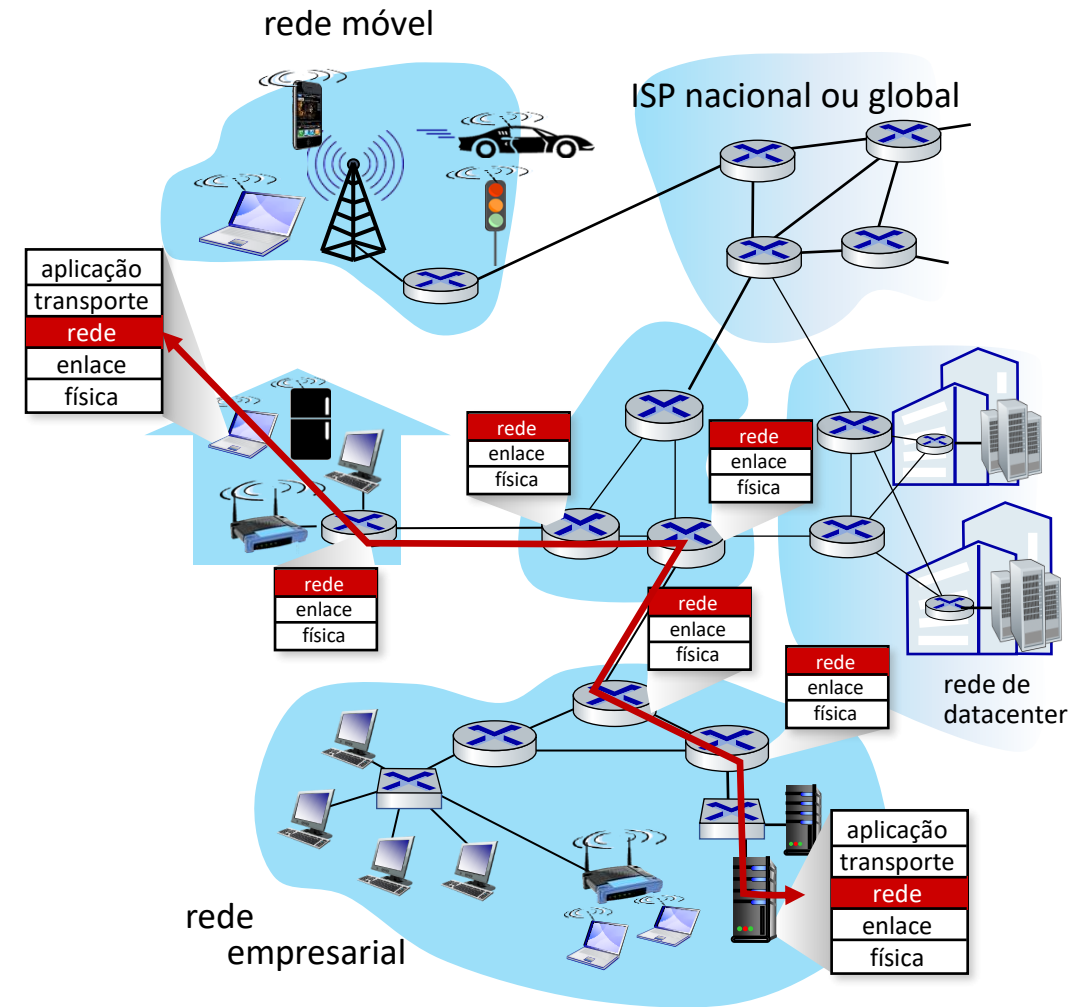


- gerenciamento e configuração de rede
 - SNMP
 - NETCONF/YANG

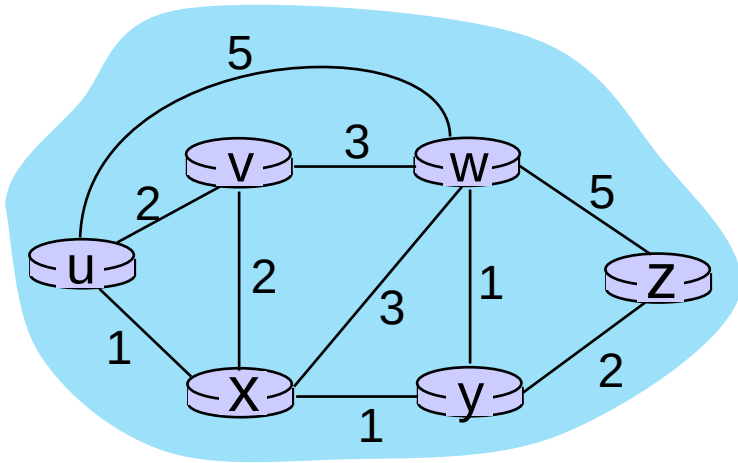
Protocolos de roteamento

Objetivo do protocolo de roteamento: determinar caminhos “bons” (equivalentemente, rotas), do hospedeiro de origem ao hospedeiro de destino, por meio da rede de roteadores

- **caminho:** sequência de roteadores que os pacotes atravessam de um dado hospedeiro de origem até um hospedeiro de destino final
- **“bom”:** menor “custo”, “mais rápido”, “menos congestionado”
- roteamento: um desafio de rede "top-10"!



Abstração de Grafo: custos de enlace



$c_{a,b}$: custo de enlace *direto* conectando a e b
ex.: $c_{w,z} = 5$, $c_{u,z} = \infty$

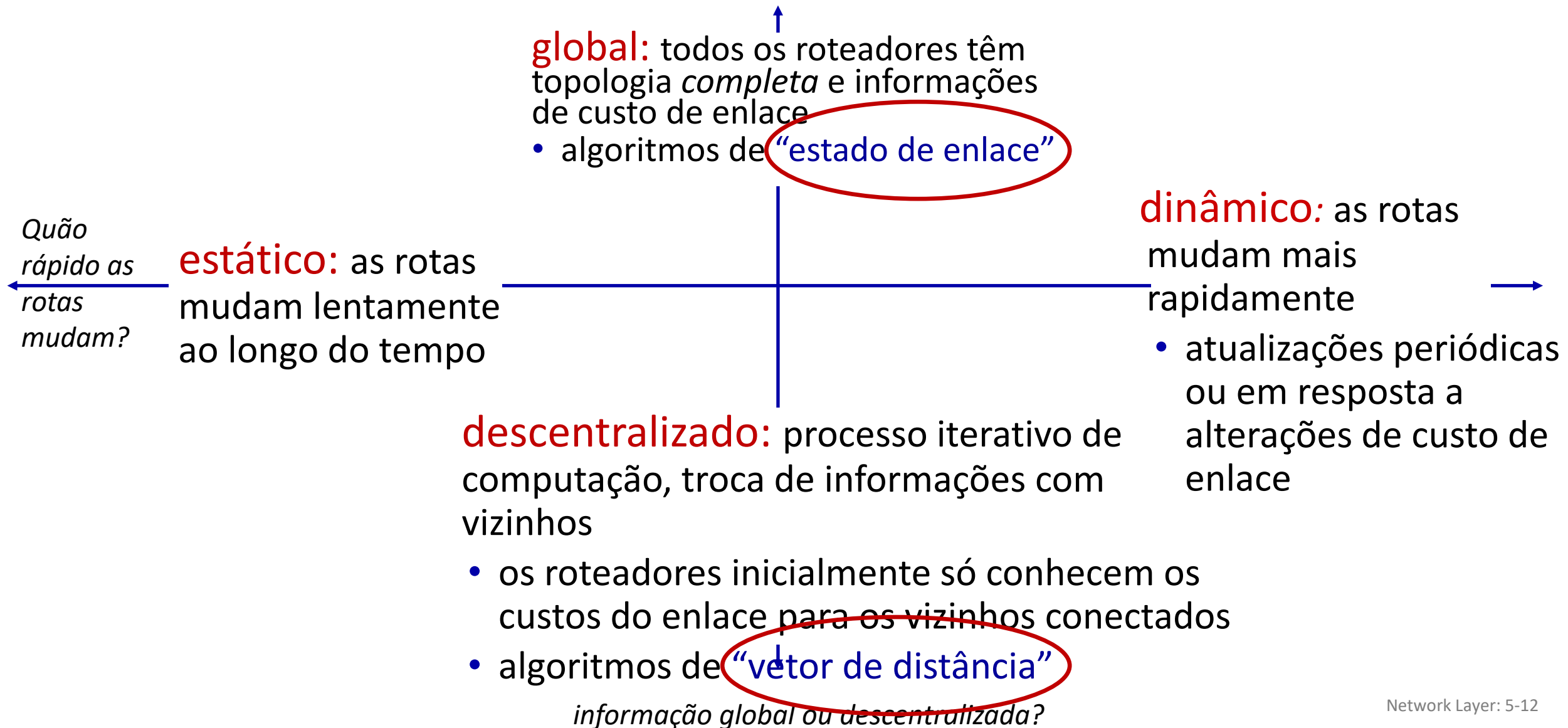
custo definido pelo operador de rede:
pode ser sempre 1, ou inversamente
relacionado à largura de banda, ou
inversamente relacionado ao
congestionamento

grafo: $G = (N, E)$

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

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

Classificação de algoritmos de roteamento



Camada de rede: roteiro do “plano de controle”

- introdução
- protocolos de roteamento
 - estado de enlace
 - vetor de distância
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- plano de controle das SDN
- Internet Control Message Protocol



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

Algoritmo de roteamento de estado de enlace de Dijkstra

- **centralizado:** topologia de rede e custos de enlace conhecidos por *todos* os nós
 - realizado via “transmissão de estado de enlace”
 - todos os nós têm as mesmas informações
- calcula os caminhos de menor custo de um nó (“origem”) para todos os outros nós
 - dá a tabela *tabela de encaminhamento* para aquele nó
- **iterativo:** após k iterações, sabe os caminhos de menores custos para k destinos

notação

- $c_{x,y}$: custo do enlace direto do nó x para o y ; $= \infty$ se não forem 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 origem para v
- N' : conjunto de nós cujo caminho de menor custo é *definitivamente* conhecido

Algoritmo de roteamento de estado de enlace de Dijkstra

1 *Inicialização:*

2 $N' = \{u\}$ /* calcule o caminho de menor custo de u para todos os outros nós */

3 Para todos os nós v

4 se v é adjacente a u /* u inicialmente conhece apenas o custo do caminho direto para os vizinhos diretos */

5 então $D(v) = c_{u,v}$ /* mas pode não ser o custo *mínimo*!

*/

6 senão $D(v) = \infty$

7

8 *Loop*

9 encontre w que não está em N' tal que $D(w)$ é um mínimo

10 adicione w em N'

11 atualize $D(v)$ para todo v adjacente a w e que não está em N' :

12 $D(v) = \min (D(v), D(w) + c_{w,v})$

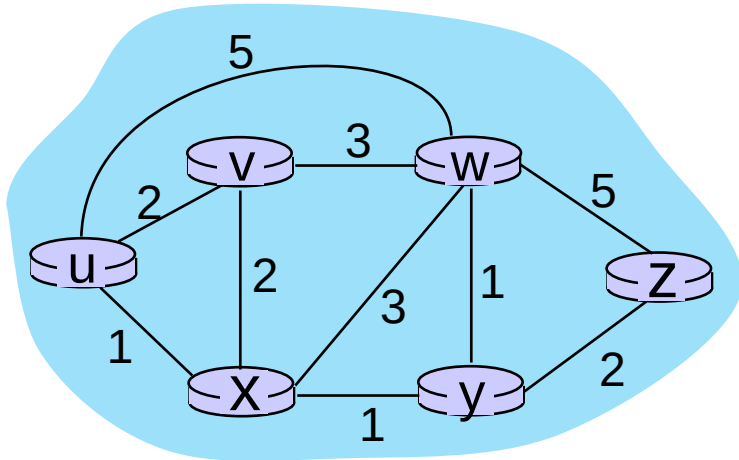
13 /* novo caminho de menor custo para v é o antigo caminho de menor custo para v

14 ou caminho de menor custo conhecido para w mais custo direto de w para v */

15 *até que todos os nós estejam em N'*

Algoritmo de Dijkstra: um exemplo

		v	w	x	y	z
Passo	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						
2						
3						
4						
5						

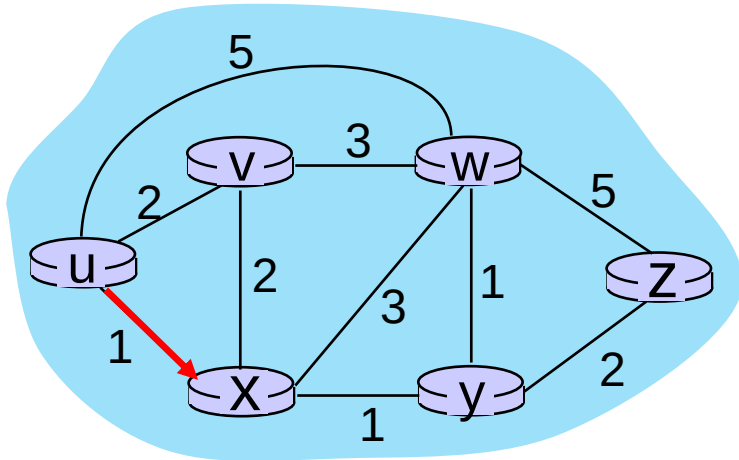


Inicialização (passo 0):

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

Algoritmo de Dijkstra: um exemplo

Passo	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	u, x					
2						
3						
4						
5						



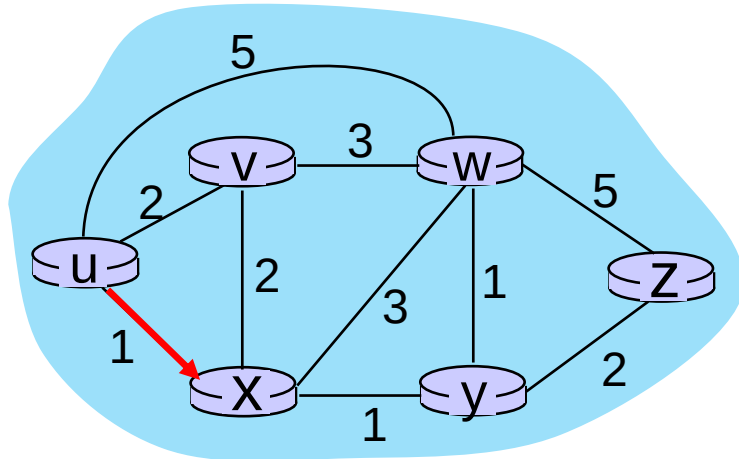
8 Loop

9 encontre a que não está em N' tal que $D(a)$ é um mínimo

10 adicione a a N'

Algoritmo de Dijkstra: um exemplo

		v	w	x	y	z
Passo	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						
3						
4						
5						



8 Loop

- 9 encontre a que não está em N' tal que $D(a)$ é um mínimo
- 10 adicione a a N'
- 11 atualize $D(b)$ para todo b adjacente a a e que não está em N' :

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

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

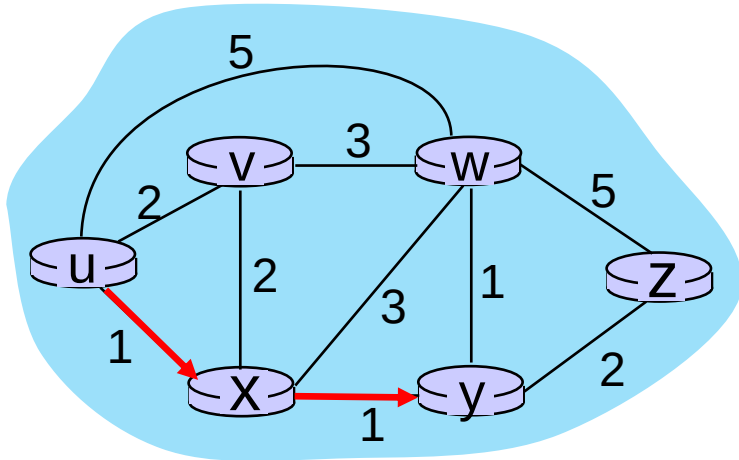
$$D(w) = \min (D(w), D(x) + c_{x,w}) = \min(5, 1+3) = 4$$

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



Algoritmo de Dijkstra: um exemplo

		v	w	x	y	z
Passo	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					
3						
4						
5						



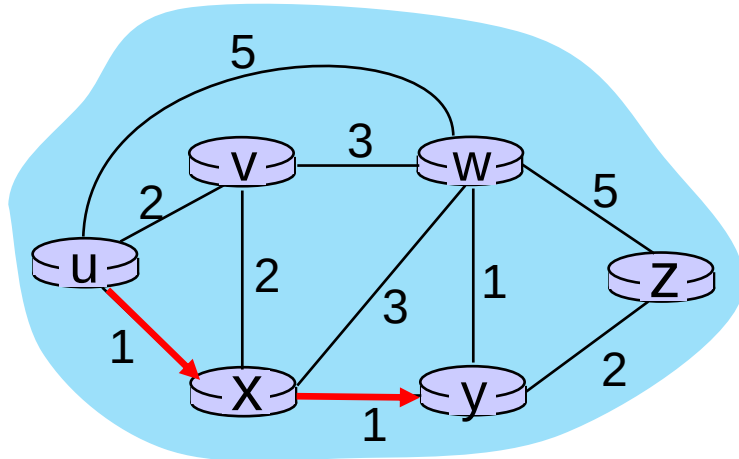
8 Loop

9 encontre a que não está em N' tal que $D(a)$ é um mínimo

10 adicione a a N'

Algoritmo de Dijkstra: um exemplo

		v	w	x	y	z
Passo	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*

- 9 encontre a que não está em N' tal que $D(a)$ é um mínimo
- 10 adicione a a N'
- 11 atualize $D(b)$ para todo b adjacente a a e que não está em N' :

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

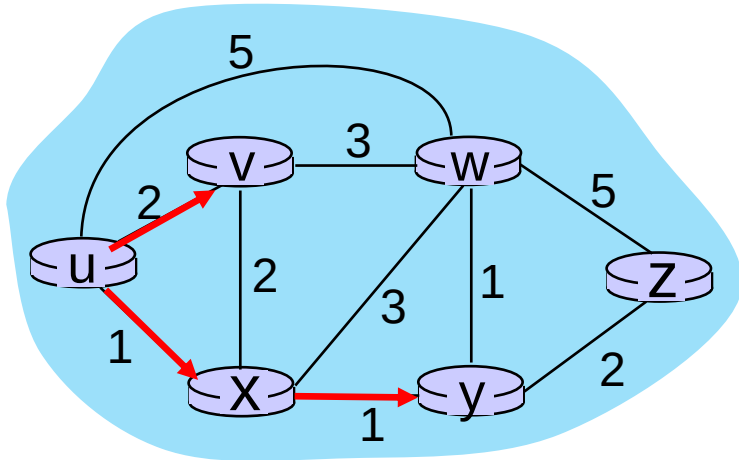
$$D(w) = \min (D(w), D(y) + c_{y,w}) = \min (4, 2+1) = 3$$

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

NEW!

Algoritmo de Dijkstra: um exemplo

		v	w	x	y	z
Passo	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						



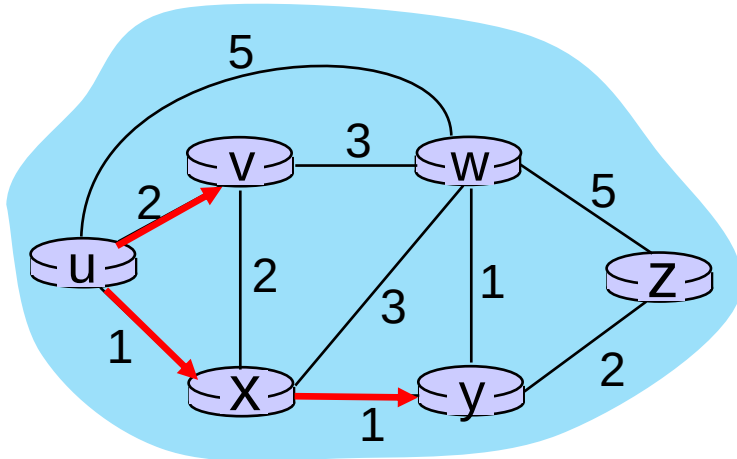
8 Loop

9 encontre a que não está em N' tal que $D(a)$ é um mínimo

10 adicione a a N'

Algoritmo de Dijkstra: um exemplo

		v	w	x	y	z
Passo	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		3,y			4,y
4						
5						



8 *Loop*

9 encontre a que não está em N' tal que $D(a)$ é um mínimo

10 adicione a a N'

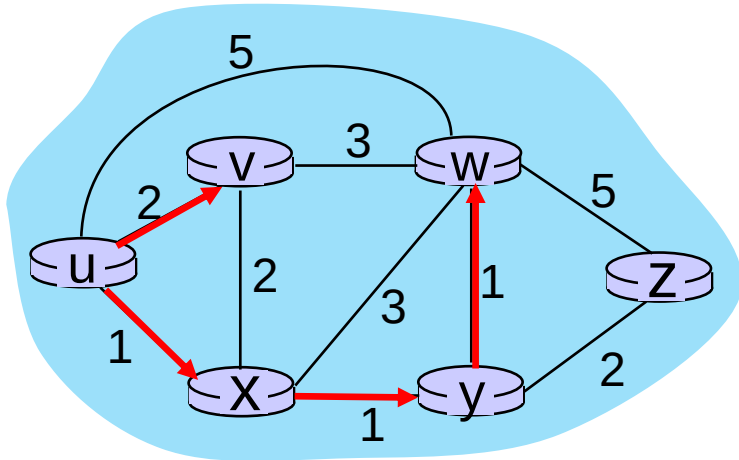
11 atualize $D(b)$ para todo b adjacente a a e que não está em N' :

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

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

Algoritmo de Dijkstra: um exemplo

		v	w	x	y	z
Passo	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		3, y			4, y
4	uxyvw					
5						



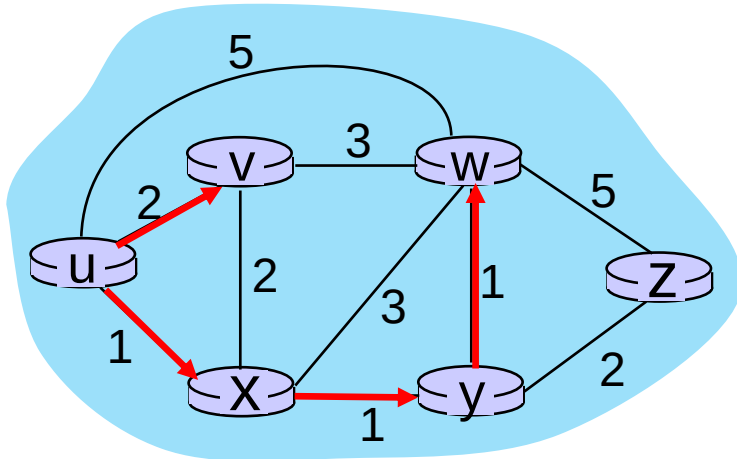
8 *Loop*

9 encontre a que não está em N' tal que $D(a)$ é um mínimo

10 adicione a a N'

Algoritmo de Dijkstra: um exemplo

		v	w	x	y	z
Passo	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		3,y			4,y
4	uxyvw					4,y
5						



8 Loop

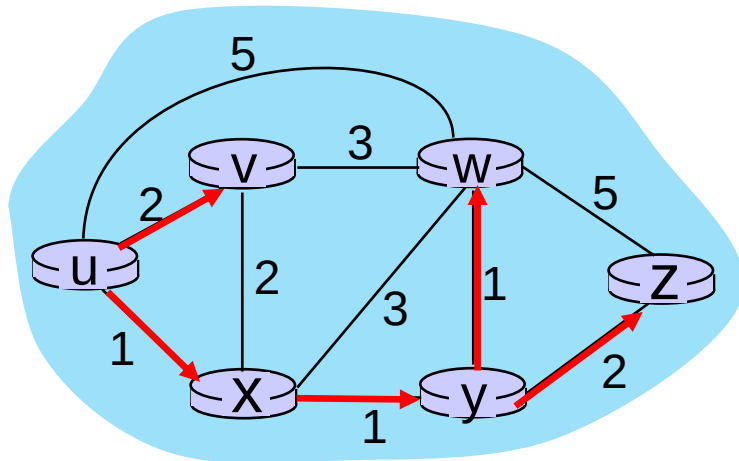
- 9 encontre a que não está em N' tal que $D(a)$ é um mínimo
- 10 adicione a a N'
- 11 atualize $D(b)$ para todo b adjacente a a e que não está em N' :

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

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

Algoritmo de Dijkstra: um exemplo

		v	w	x	y	z
Passo	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		3, y			4, y
4	uxyvw					4, y
5	uxyvwz					

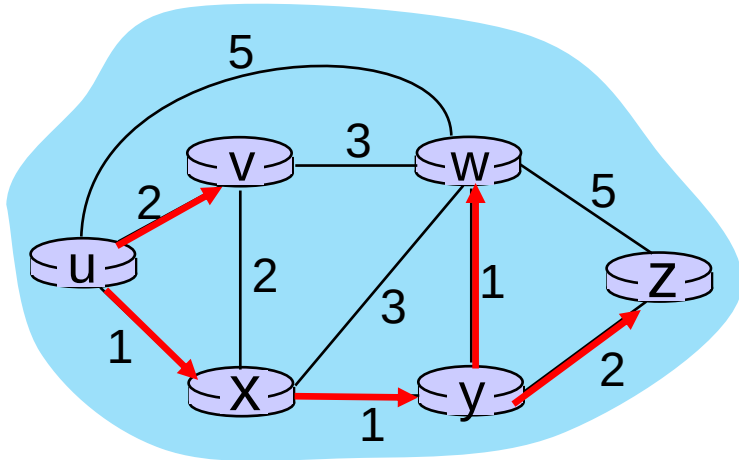


8 Loop

- 9 encontre a que não está em N' tal que $D(a)$ é um mínimo
- 10 adicione a a N'

Algoritmo de Dijkstra: um exemplo

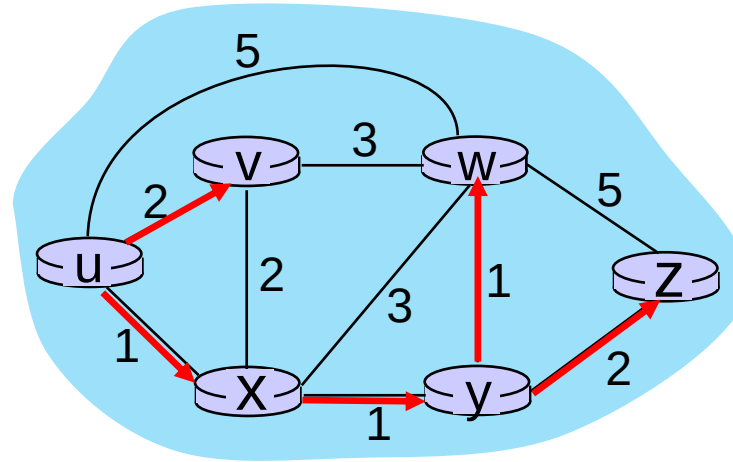
		v	w	x	y	z
Passo	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



8 Loop

- 9 encontre a que não está em N' tal que $D(a)$ é um mínimo
- 10 adicione a a N'
- 11 atualize $D(b)$ para todo b adjacente a a e que não está em N' :
$$D(b) = \min (D(b), D(a) + c_{a,b})$$

Algoritmo de Dijkstra: um exemplo



árvore de caminhos de menores custos a partir de u resultante:

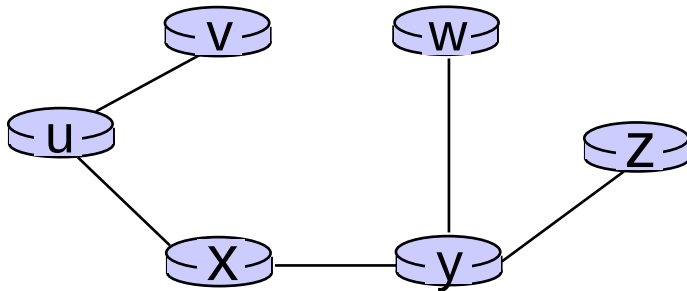


tabela de encaminhamento resultante em u :

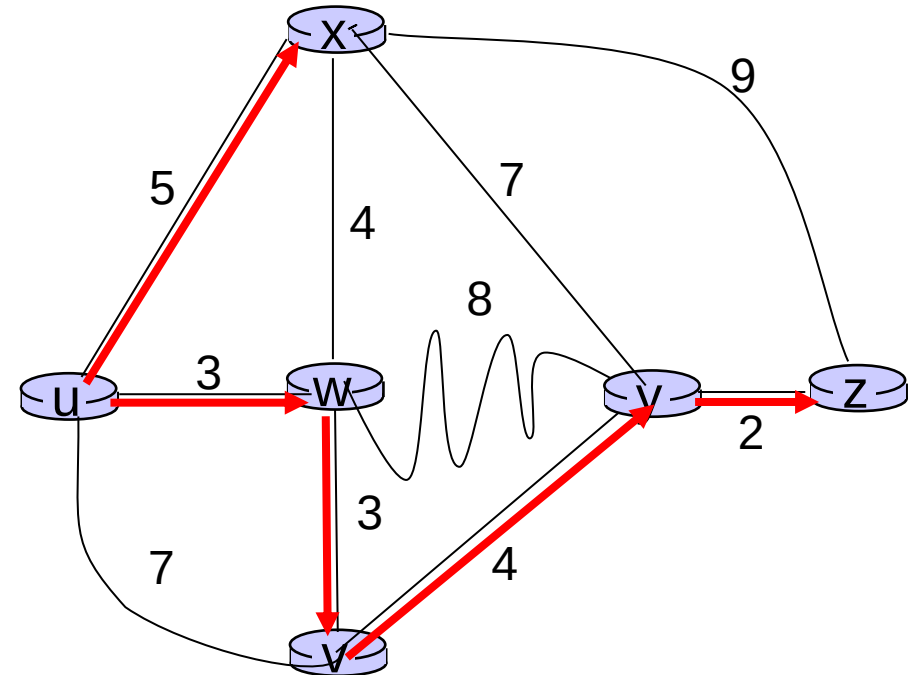
destino	enlace de saída
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

rota de u para v diretamente

rota de u para todos os outros destinos via x

Algoritmo de Dijkstra: outro exemplo

Passo	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	uwXvyz					



notas:

- Construa a árvore de caminhos de menor custo rastreando nós predecessores
- empates podem existir (podem ser decididos arbitrariamente)

Algoritmo de Dijkstra: discussão

complexidade do algoritmo: n nós

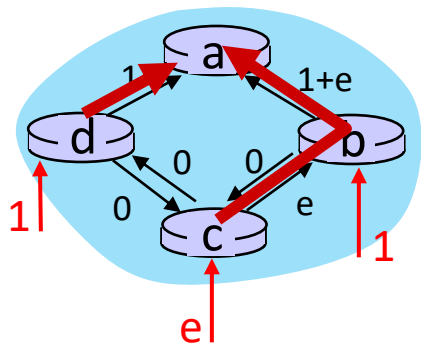
- cada iteração n : precisa checar todos os nós, w , não em N'
- $n(n+1)/2$ comparações: complexidade $O(n^2)$
- implementações mais eficientes possíveis: $O(n \log n)$

complexidade de mensagem:

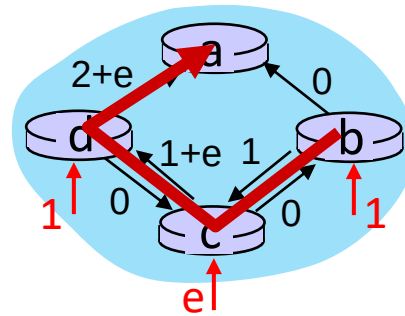
- cada roteador precisa *difundir* suas informações de estado de enlace para outros n roteadores
- algoritmos de transmissão eficientes (e interessantes!): $O(n)$ cruzamentos de enlaces para disseminar uma mensagem de difusão a partir de uma origem
- a mensagem de um roteador cruza $O(n)$ enlaces: complexidade geral de mensagem: $O(n^2)$

Algoritmo de Dijkstra: possíveis oscilações

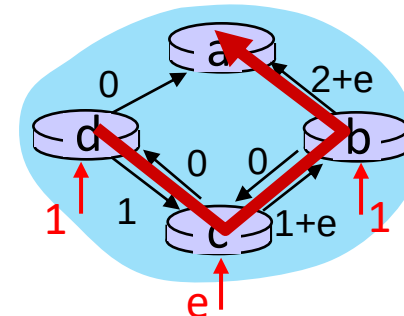
- quando os custos do enlace dependem do volume de tráfego, **oscilações de rota** podem ocorrer
- cenário de exemplo:
 - roteamento para o destino a , tráfego entrando em d , c , e b com taxas 1, e (<1), e 1, respectivamente.
 - os custos do enlace são direcionais e dependentes do volume



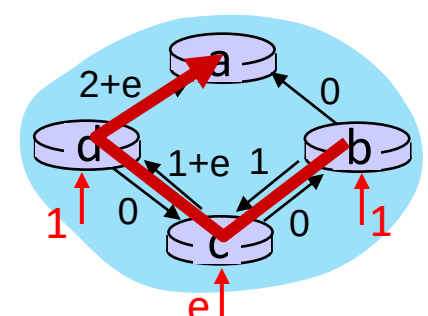
inicialmente



dados esses custos,
encontre um novo
roteamento....
resultando em
novos custos



dados esses custos,
encontre um novo
roteamento....
resultando em
novos custos



dados esses custos,
encontre um novo
roteamento....
resultando em
novos custos

Camada de rede: roteiro do “plano de controle”

- introdução
- protocolos de roteamento
 - estado de enlace
 - **vetor de distância**
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- plano de controle das SDN
- Internet Control Message Protocol



- gerenciamento e configuração de rede
 - 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:

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

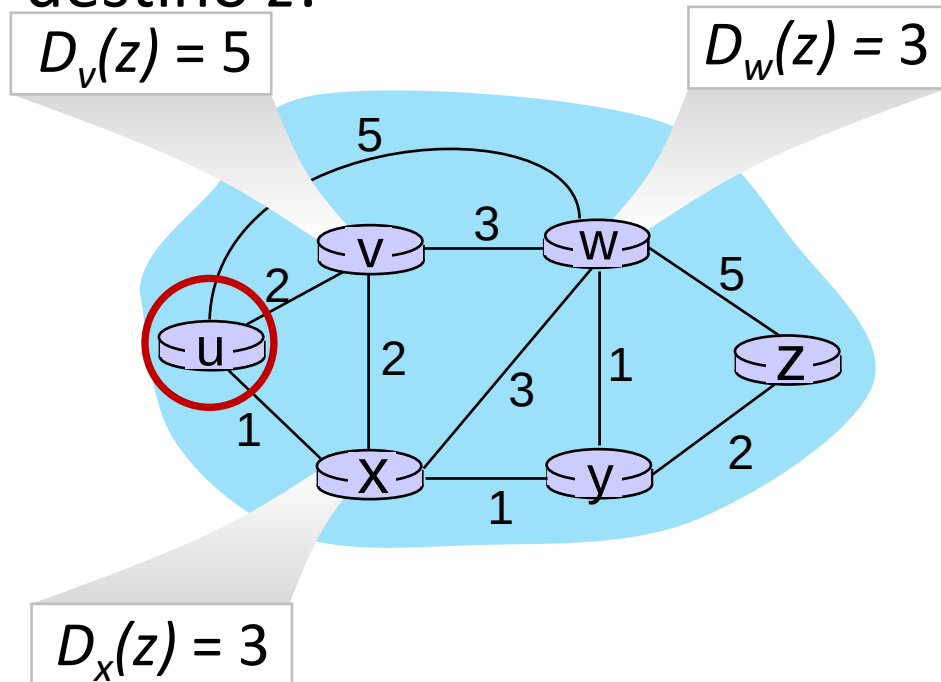
min tomado de todos os vizinhos v de x

custo do caminho de menor custo
estimado de v para y

custo direto do enlace de x para v

Exemplo de Bellman-Ford

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



A equação de Bellman-Ford diz que:

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

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

Algoritmo de vetor de distância

ideia chave:

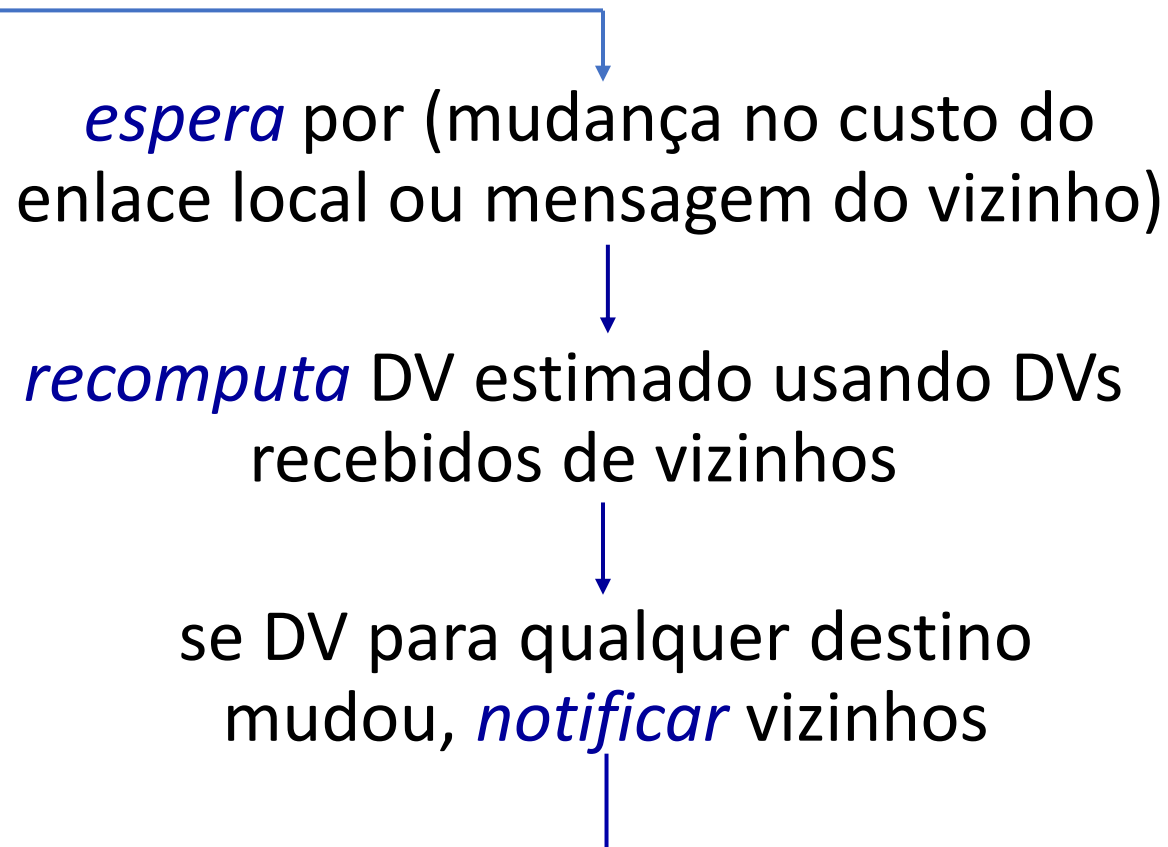
- de tempos em tempos, cada nó envia sua própria estimativa de vetor de distância (DV) para os vizinhos
- quando x recebe novo DV estimado de qualquer vizinho, ele atualiza seu próprio DV usando a equação B-F:

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

- sob condições naturais, a estimativa $D_x(y)$ converge para o menor custo real $d_x(y)$

Algoritmo de vetor de distância:

cada nó:



iterativo, assíncrono: cada iteração local causada por:

- alteração do custo do enlace local
- mensagem de atualização de DV de vizinho

distribuído, auto-parada: cada nó notifica os vizinhos *apenas* quando seu DV muda

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

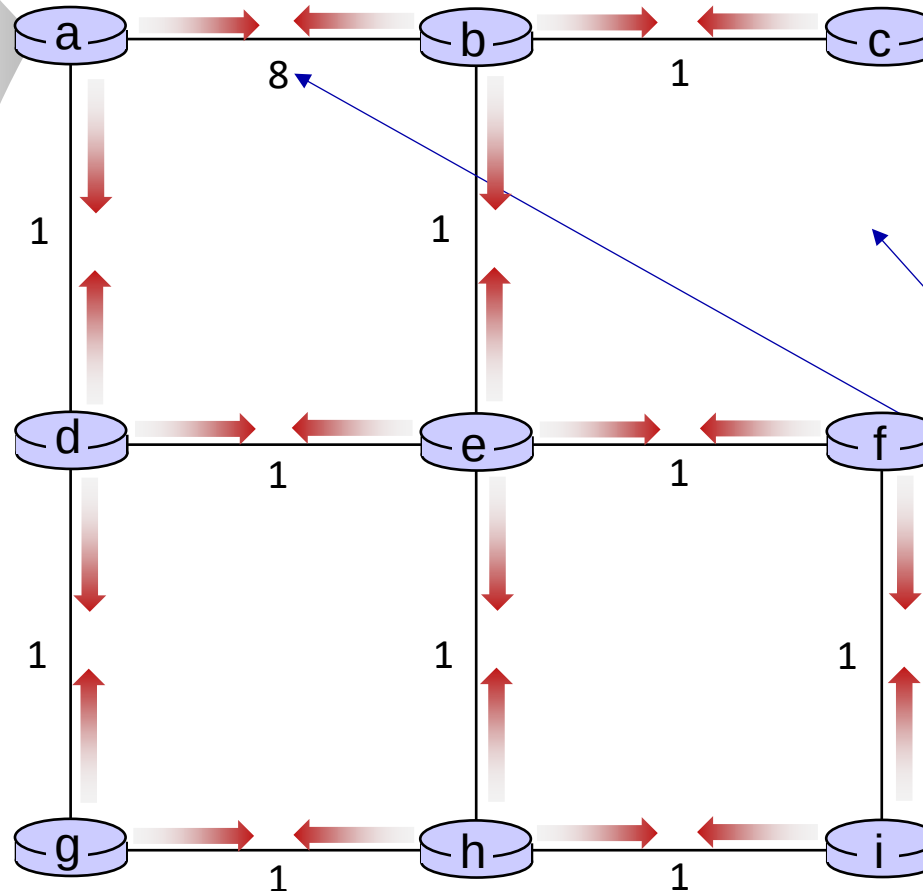
Vetor de distância: exemplo



t=0

- Todos os nós têm estimativas de distância para os vizinhos mais próximos (somente)
- 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:

- enlace perdido
- custo maior

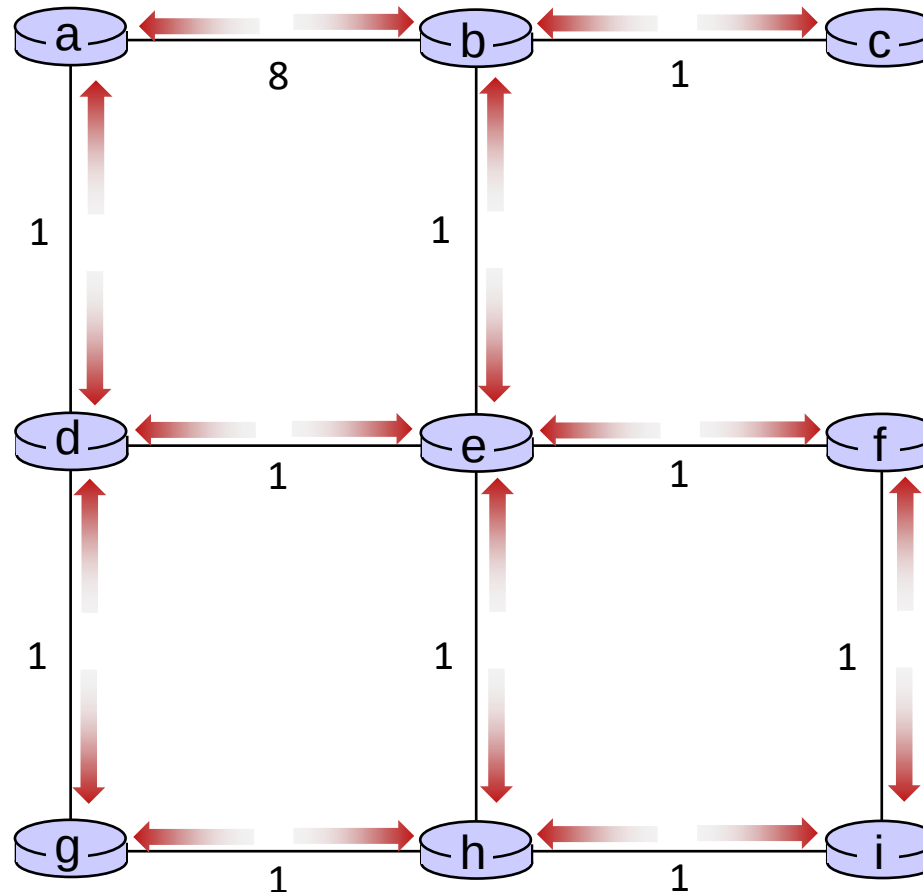
Exemplo de vetor de distância: iteração



$t=1$

Todos os nós:

- recebem vetores de distância de vizinhos
- computam seu novo vetor de distância local
- enviam seu novo vetor de distância local para os vizinhos



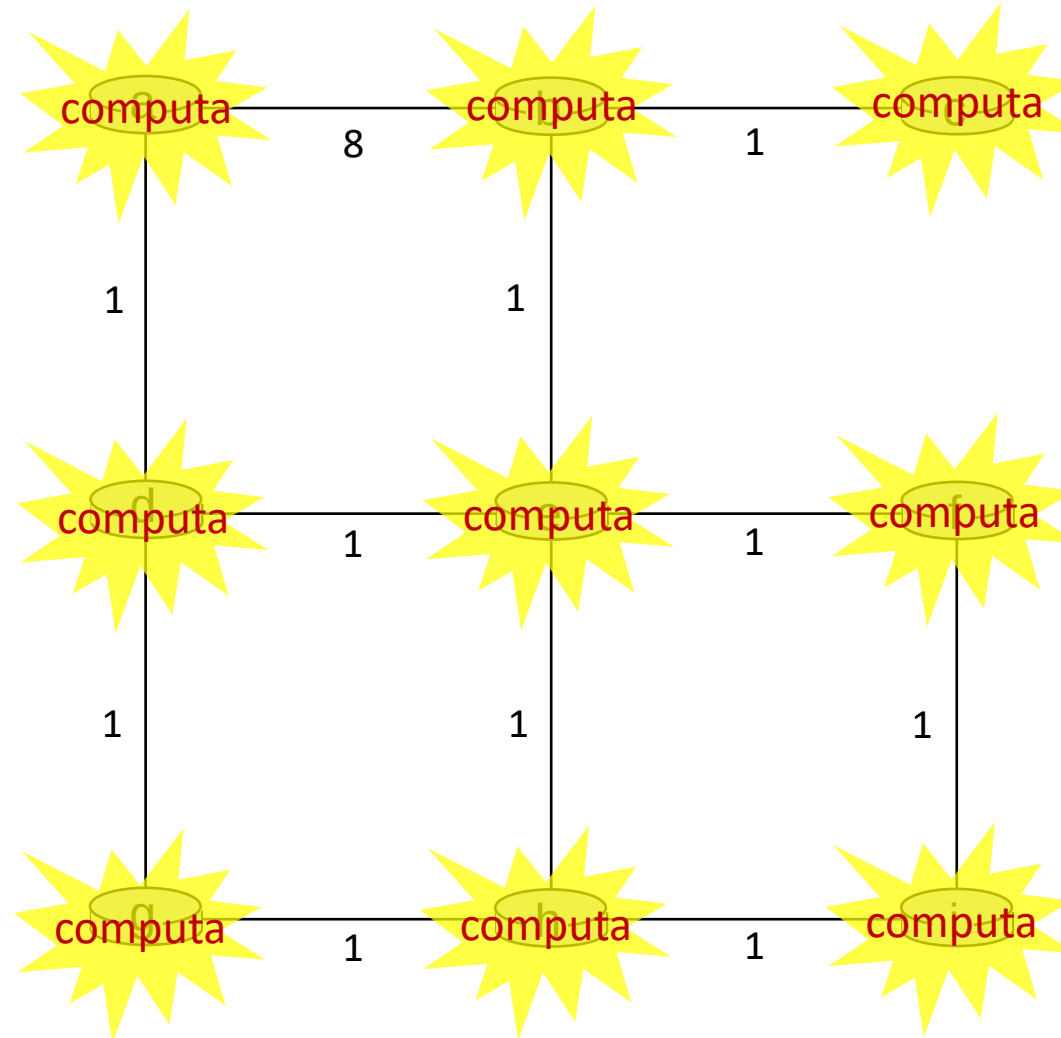
Exemplo de vetor de distância: iteração



$t=1$

Todos os nós:

- recebem vetores de distância de vizinhos
- computam seu novo vetor de distância local
- enviam seu novo vetor de distância local para os vizinhos



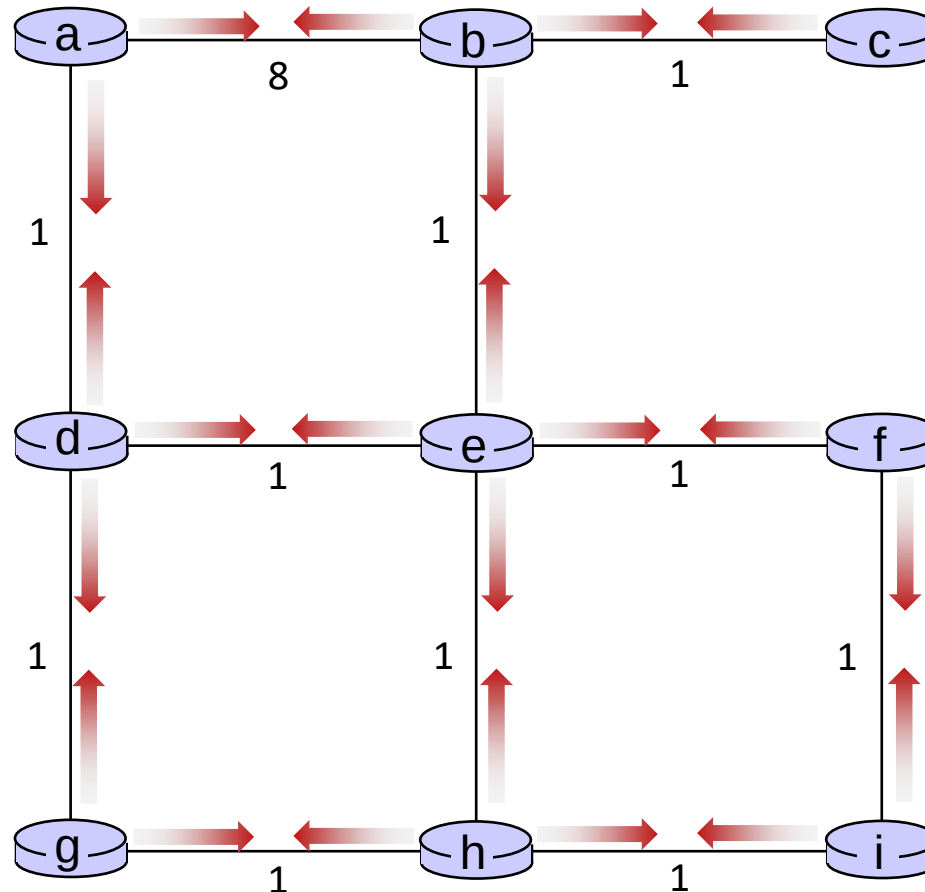
Exemplo de vetor de distância: iteração



$t=1$

Todos os nós:

- recebem vetores de distância de vizinhos
- computam seu novo vetor de distância local
- enviam seu novo vetor de distância local para os vizinhos



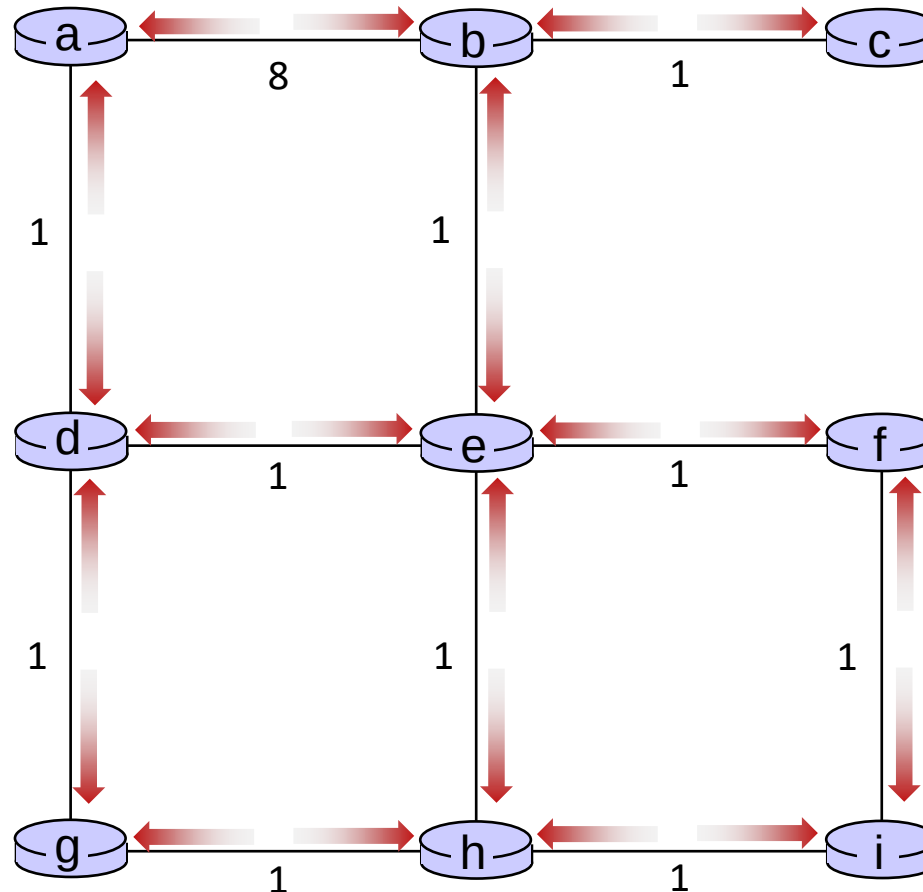
Exemplo de vetor de distância: iteração



t=2

Todos os nós:

- recebem vetores de distância de vizinhos
- computam seu novo vetor de distância local
- enviam seu novo vetor de distância local para os vizinhos



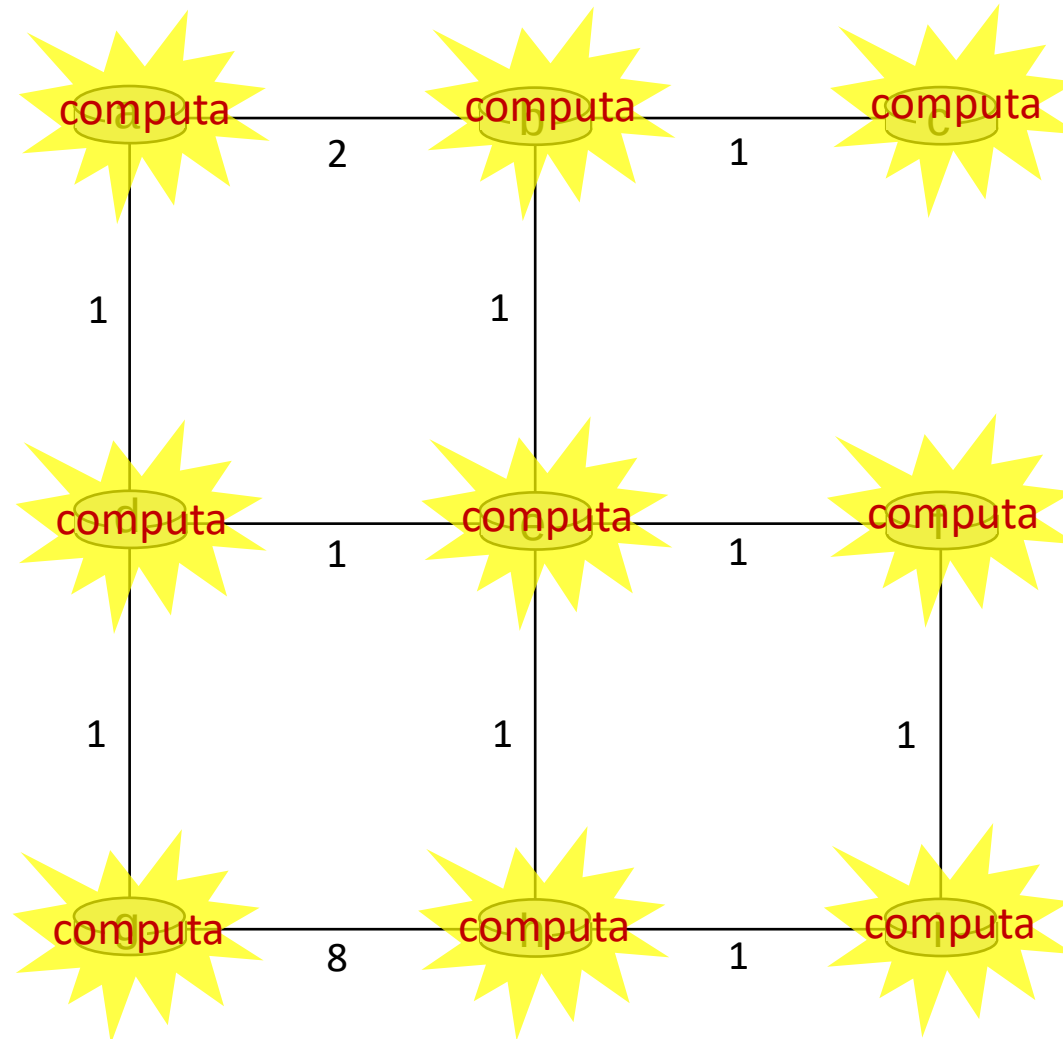
Exemplo de vetor de distância: iteração



t=2

Todos os nós:

- recebem vetores de distância de vizinhos
- computam seu novo vetor de distância local
- enviam seu novo vetor de distância local para os vizinhos



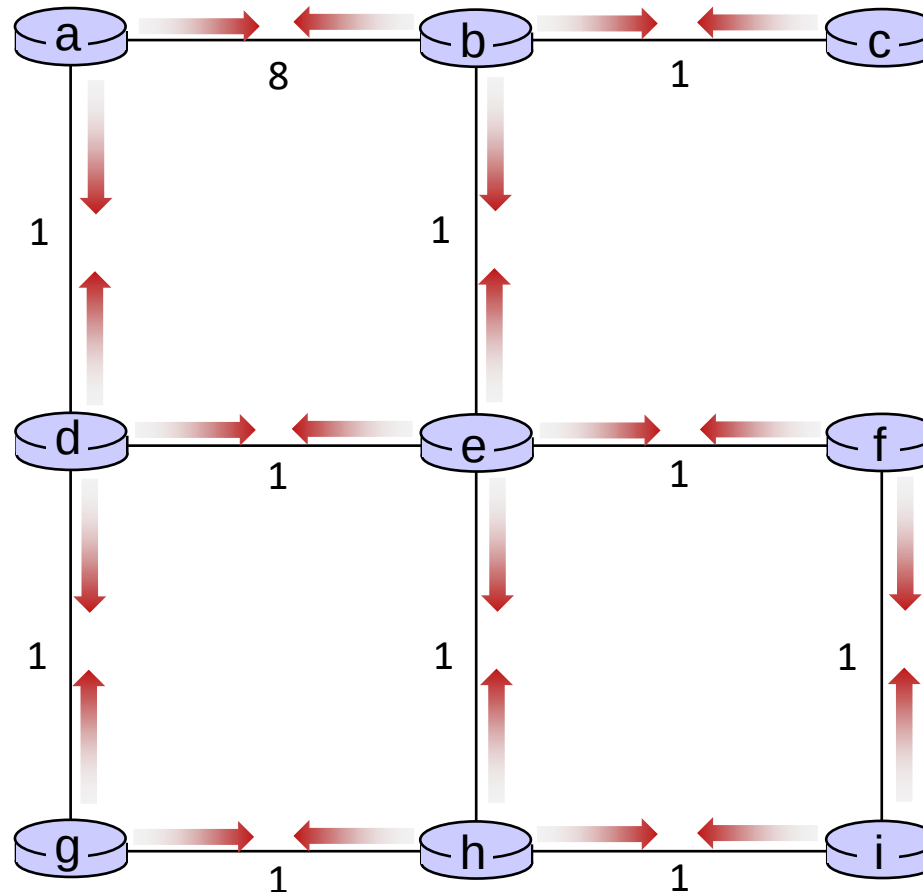
Exemplo de vetor de distância: iteração



t=2

Todos os nós:

- recebem vetores de distância de vizinhos
- computam 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

Vamos agora dar uma olhada nos *cálculos* iterativos nos nós

Exemplo de vetor de distâ



t=1

- b recebe DVs de a, c, e 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:

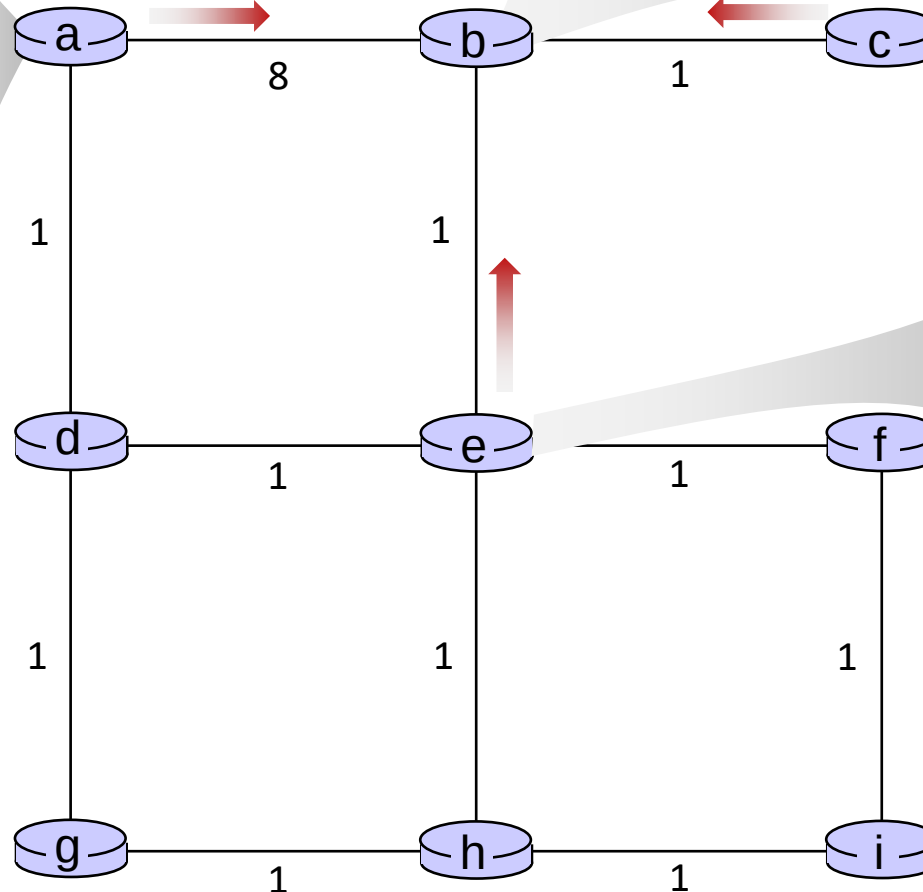
$D_b(a)=8$ $D_b(f)=\infty$
 $D_b(c)=1$ $D_b(g)=\infty$
 $D_b(d)=\infty$ $D_b(h)=\infty$
 $D_b(e)=1$ $D_b(i)=\infty$

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:

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



Exemplo de vetor de distâ

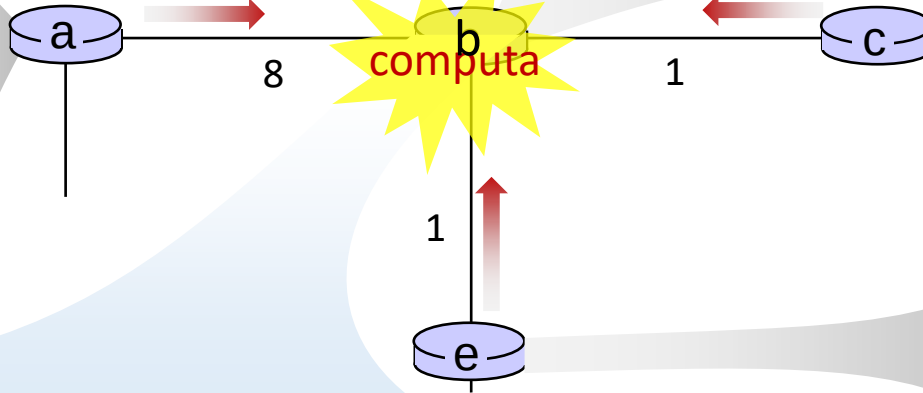


t=1

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

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:

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

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:

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

$D_b(a) = \min\{c_{b,a}+D_a(a), c_{b,c}+D_c(a), c_{b,e}+D_e(a)\} = \min\{8, \infty, \infty\} = 8$
 $D_b(c) = \min\{c_{b,a}+D_a(c), c_{b,c}+D_c(c), c_{b,e}+D_e(c)\} = \min\{\infty, 1, \infty\} = 1$
 $D_b(d) = \min\{c_{b,a}+D_a(d), c_{b,c}+D_c(d), c_{b,e}+D_e(d)\} = \min\{9, \infty, 2\} = 2$
 $D_b(e) = \min\{c_{b,a}+D_a(e), c_{b,c}+D_c(e), c_{b,e}+D_e(e)\} = \min\{\infty, \infty, 1\} = 1$
 $D_b(f) = \min\{c_{b,a}+D_a(f), c_{b,c}+D_c(f), c_{b,e}+D_e(f)\} = \min\{\infty, \infty, 2\} = 2$
 $D_b(g) = \min\{c_{b,a}+D_a(g), c_{b,c}+D_c(g), c_{b,e}+D_e(g)\} = \min\{\infty, \infty, \infty\} = \infty$
 $D_b(h) = \min\{c_{b,a}+D_a(h), c_{b,c}+D_c(h), c_{b,e}+D_e(h)\} = \min\{\infty, \infty, 2\} = 2$
 $D_b(i) = \min\{c_{b,a}+D_a(i), c_{b,c}+D_c(i), c_{b,e}+D_e(i)\} = \min\{\infty, \infty, \infty\} = \infty$

Exemplo de vetor de distâ



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:

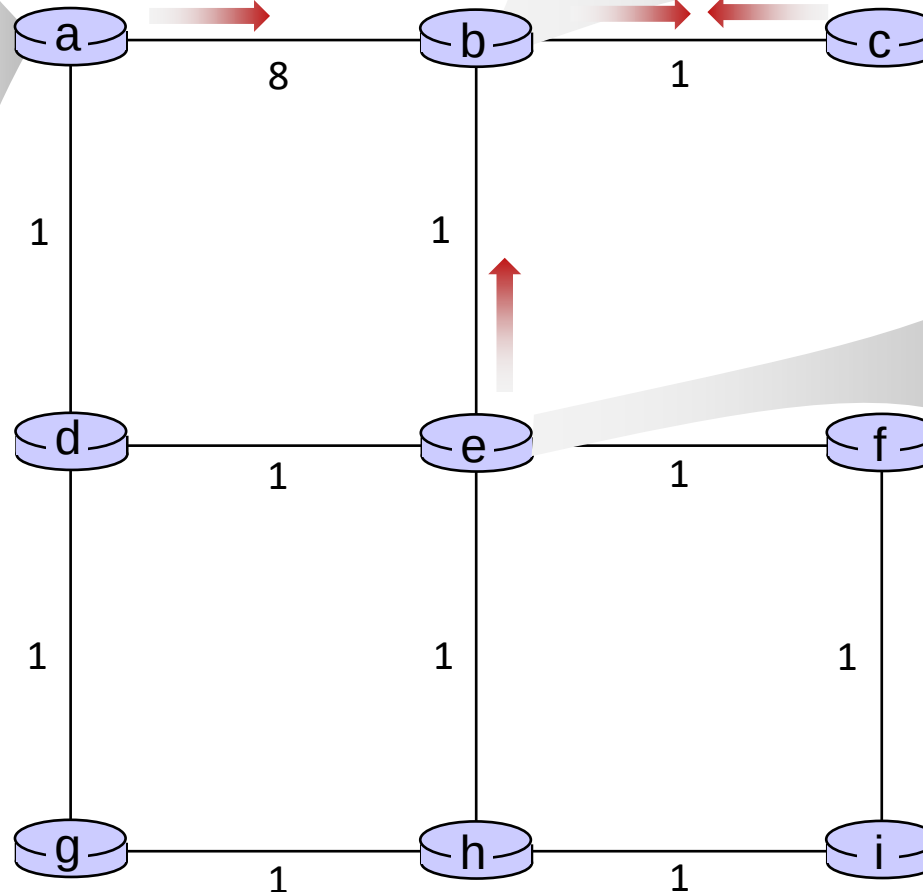
$D_b(a)=8$ $D_b(f)=\infty$
 $D_b(c)=1$ $D_b(g)=\infty$
 $D_b(d)=\infty$ $D_b(h)=\infty$
 $D_b(e)=1$ $D_b(i)=\infty$

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:

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



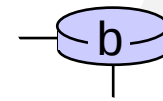
Exemplo de vetor de distâ



t=1

- c recebe DVs de b e computa:

$$\begin{aligned}D_c(a) &= \min\{c_{c,b} + D_b(a)\} = 1 + 8 = 9 \\D_c(b) &= \min\{c_{c,b} + D_b(b)\} = 1 + 0 = 1 \\D_c(d) &= \min\{c_{c,b} + D_b(d)\} = 1 + \infty = \infty \\D_c(e) &= \min\{c_{c,b} + D_b(e)\} = 1 + 1 = 2 \\D_c(f) &= \min\{c_{c,b} + D_b(f)\} = 1 + \infty = \infty \\D_c(g) &= \min\{c_{c,b} + D_b(g)\} = 1 + \infty = \infty \\D_c(h) &= \min\{c_{c,b} + D_b(h)\} = 1 + \infty = \infty \\D_c(i) &= \min\{c_{c,b} + D_b(i)\} = 1 + \infty = \infty\end{aligned}$$



1

computa

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 online para mais exemplos:
http://gaia.cs.umass.edu/kurose_ross/interactive/

Exemplo de vetor de distância



t=1

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

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

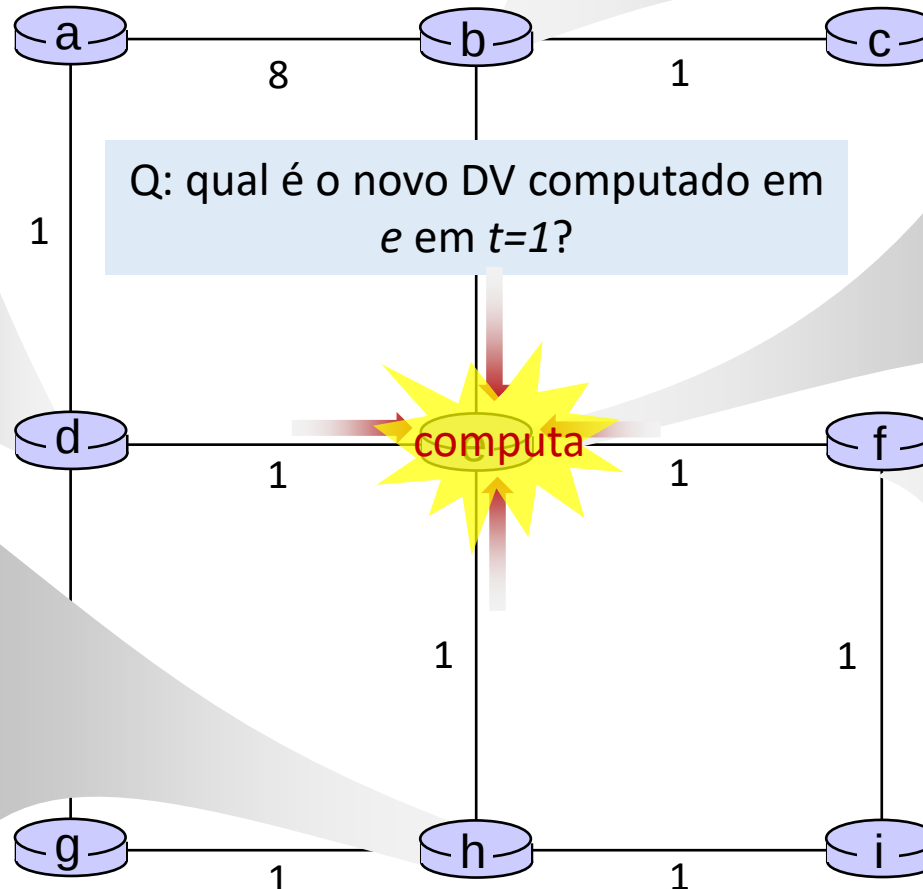
DV em h:	
$D_c(a) = \infty$	
$D_c(b) = \infty$	
$D_c(c) = \infty$	
$D_c(d) = \infty$	
$D_c(e) = 1$	
$D_c(f) = \infty$	
$D_c(g) = 1$	
$D_c(h) = 0$	
$D_c(i) = 1$	

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

ição

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

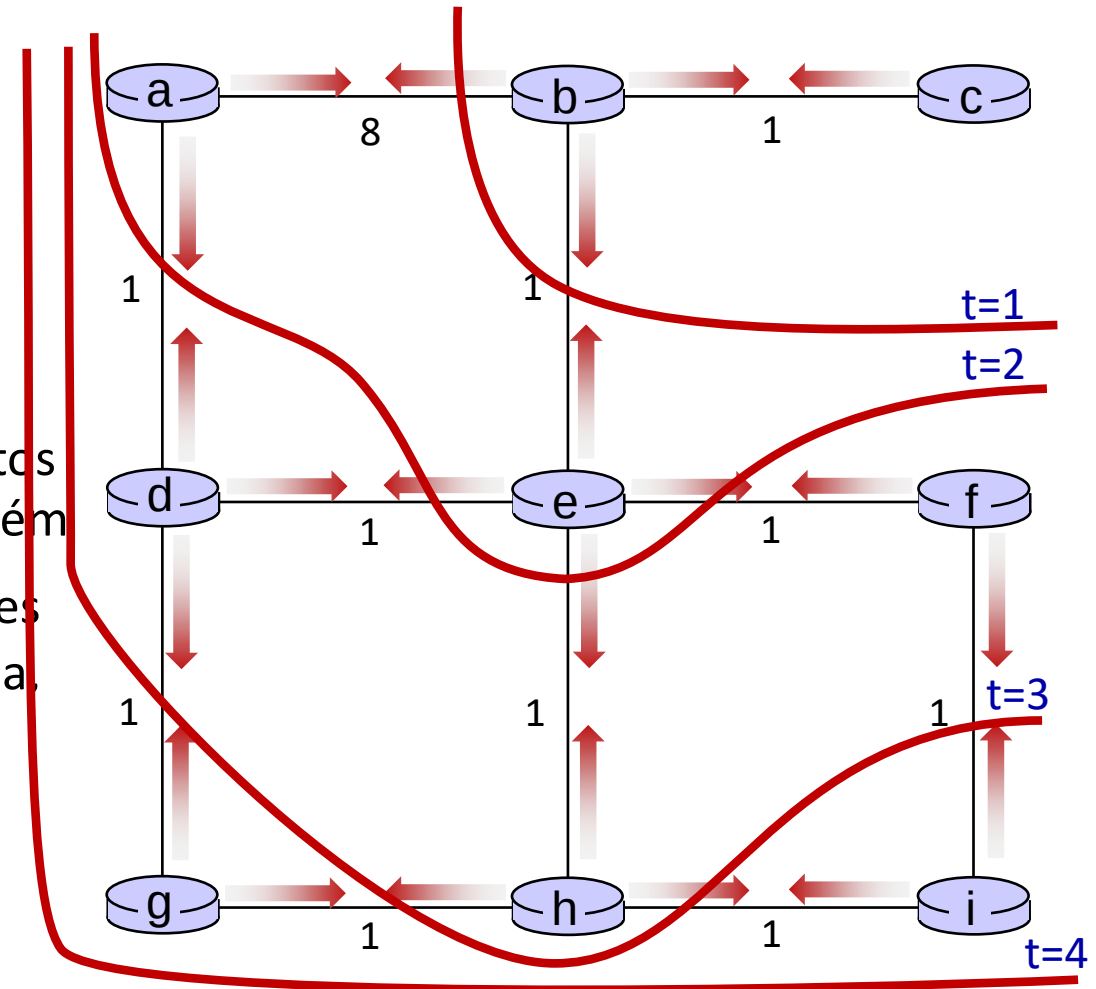
DV em f:	
$D_c(a) = \infty$	
$D_c(b) = \infty$	
$D_c(c) = \infty$	
$D_c(d) = \infty$	
$D_c(e) = 1$	
$D_c(f) = 0$	
$D_c(g) = \infty$	
$D_c(h) = \infty$	
$D_c(i) = 1$	



Vetor de distância: difusão de informações de estado

Comunicação iterativa, etapas de computação difundem informações através da rede:

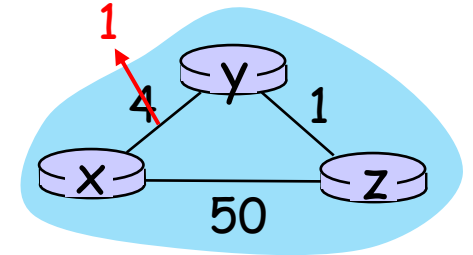
-  $t=0$ estado de c em $t=0$ está somente em c
-  $t=1$ estado de c em $t=0$ propagou-se para b , e pode influenciar computações de vetor de distância em até **1** salto de distância, isto é, em b
-  $t=2$ estado de c em $t=0$ pode agora influenciar computações de vetor de distância em até **2** saltos de distância, isto é, em b e agora em a , e e também
-  $t=3$ estado de c em $t=0$ pode influenciar computações de vetor de distância em até **3** saltos de distância, isto é, em b , a , e e , e agora em c , f , e h também
-  $t=4$ estado de c em $t=0$ pode influenciar computações de vetor de distância em até **4** nós de distância, isto é, em b , a , e , c , f , h e agora em g e i também



Vetor de distância: alterações no custo do enlace

alterações de custo de enlace:

- nó detecta mudança no custo do enlace local
- atualiza informações de roteamento, recalcula o DV local
- se DV mudar, notifica os vizinhos



“boas
notícias
viajam
rápido”

t_0 : y detecta mudança de custo de enlace, atualiza seu DV, informa seus vizinhos.

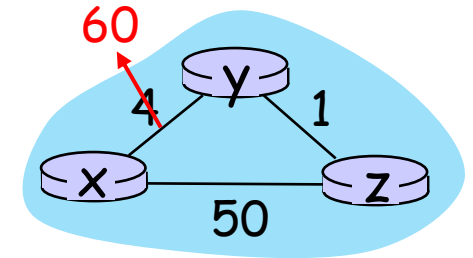
t_1 : z recebe atualização de y, atualiza sua tabela, computa novo custo mínimo para x, envia seu DV para seus vizinhos

t_2 : y recebe atualização de z, atualiza sua tabela de distância.
Menores custos de y *não* mudam, portanto y *não* envia uma mensagem para z.

Vetor de distância: alterações no custo do enlace

alterações de custo de enlace:

- nó detecta mudança no custo do enlace local
- “**más notícias viajam devagar**” – problema de contagem ao infinito:
 - y vê que o enlace direto para x tem um novo custo de 60, mas z disse que tem um caminho com um custo de 5. Então y calcula “meu novo custo para x será 6, via z); notifica z de novo custo de 6 a x.
 - z aprende que o caminho para x via y tem um novo custo 6, então z calcula “meu novo custo para x será 7 via y”, notifica y do novo custo de 7 para x.
 - y descobre que o caminho para x via z tem um novo custo 7, então y calcula “meu novo custo para x será 8 via y”, notifica z do novo custo de 8 para x.
 - z descobre que o caminho para x via y tem um novo custo 8, então z calcula “meu novo custo para x será 9 via y”, notifica y do novo custo de 9 para x.
 - ...
- veja o texto para soluções. Algoritmos distribuídos são complicados!



Comparação de algoritmos de estado de enlace (LS) e vetor de distância (DV)

complexidade de mensagem

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

DV: troca entre vizinhos; o tempo de convergência varia

velocidade de convergência

LS: algoritmo $O(n^2)$, requer $O(n^2)$ mensagens

- pode ter oscilações

DV: tempo de convergência varia

- pode ter loops de roteamento
- problema de contagem ao infinito

robustez: o que acontece se o roteador tiver um mau funcionamento ou estiver comprometido?

LS:

- roteador pode anunciar custo de *enlace* incorreto
- cada roteador computa somente sua *própria* tabela

DV:

- roteador DV pode anunciar custos de *caminho* incorretos (“Eu tenho um caminho de custo *realmente* baixo para todos os lugares”): buraco negro
- tabela de cada roteador usada por outros: erro propagado 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 das SDN
 - Internet Control Message Protocol
- 
- gerenciamento e configuração de rede
 - SNMP
 - NETCONF/YANG

Tornando o roteamento escalável

nosso estudo de roteamento até agora - idealizado

- todos os roteadores idênticos
- rede “fixa”

... não é verdade na prática

escala: bilhões de destinos:

- não pode armazenar todos os destinos nas tabelas de roteamento!
- troca de tabela de roteamento inundaria enlaces!

autonomia administrativa:

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

Abordagem da Internet para roteamento escalável

agregar roteadores em regiões conhecidas como “sistemas autônomos” (AS) (também chamados “domínios”)

intra-AS (ou “intra-domínio”):
roteamento *dentro do mesmo AS* (“rede”)

- todos os roteadores no AS devem executar o mesmo protocolo intradomínio
- roteadores em diferentes AS podem executar diferentes protocolos de roteamento intradomínio
- **roteador gateway:** na “borda” de seu próprio AS, tem enlace(s) para roteador(es) em outros AS'es

inter-AS (ou “inter-domínio”):
roteamento *entre* AS'es

- gateways executam roteamento entre domínios (assim como roteamento intradomínio)

AS'es interconectados

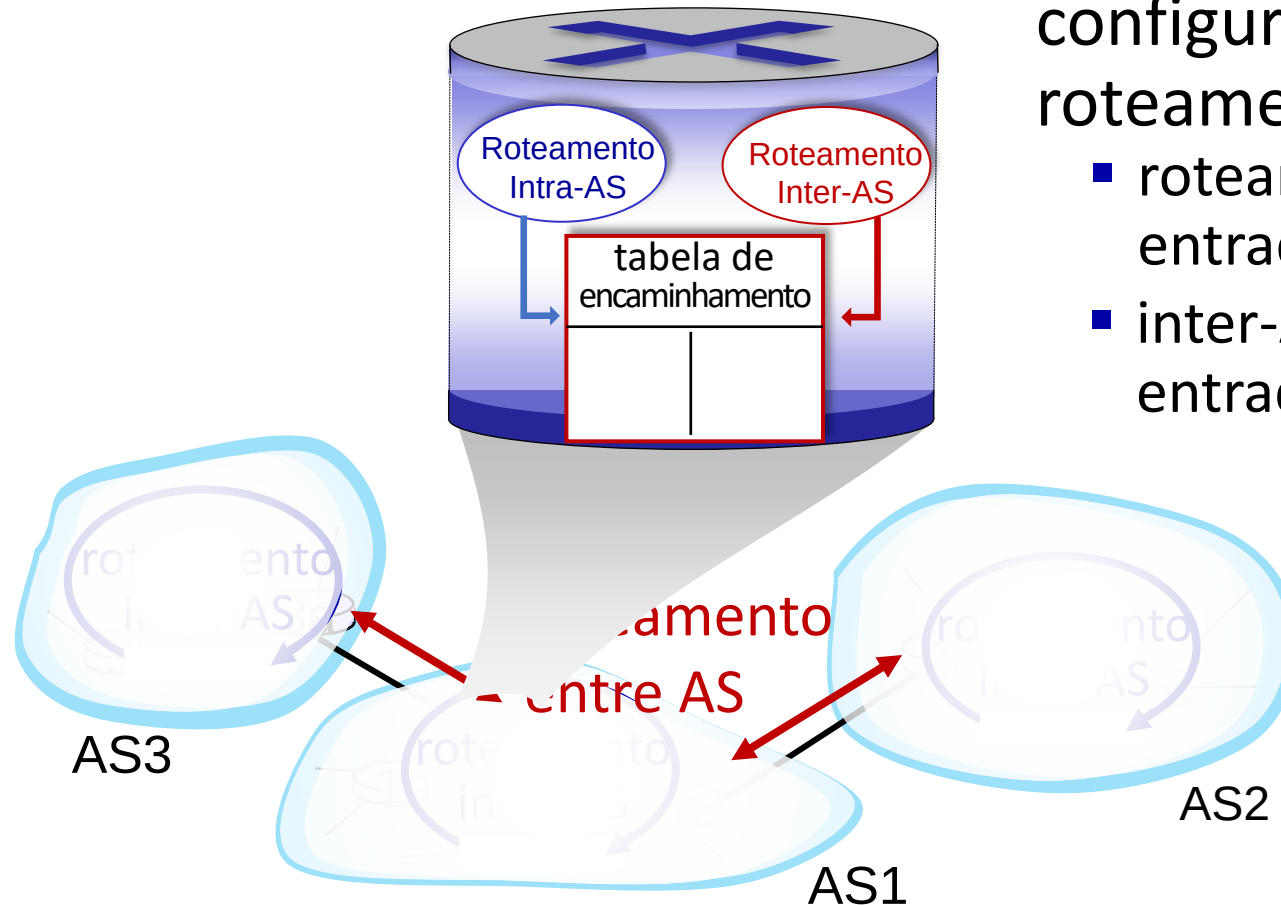


tabela de encaminhamento configurada por algoritmos de roteamento intra e inter-AS

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

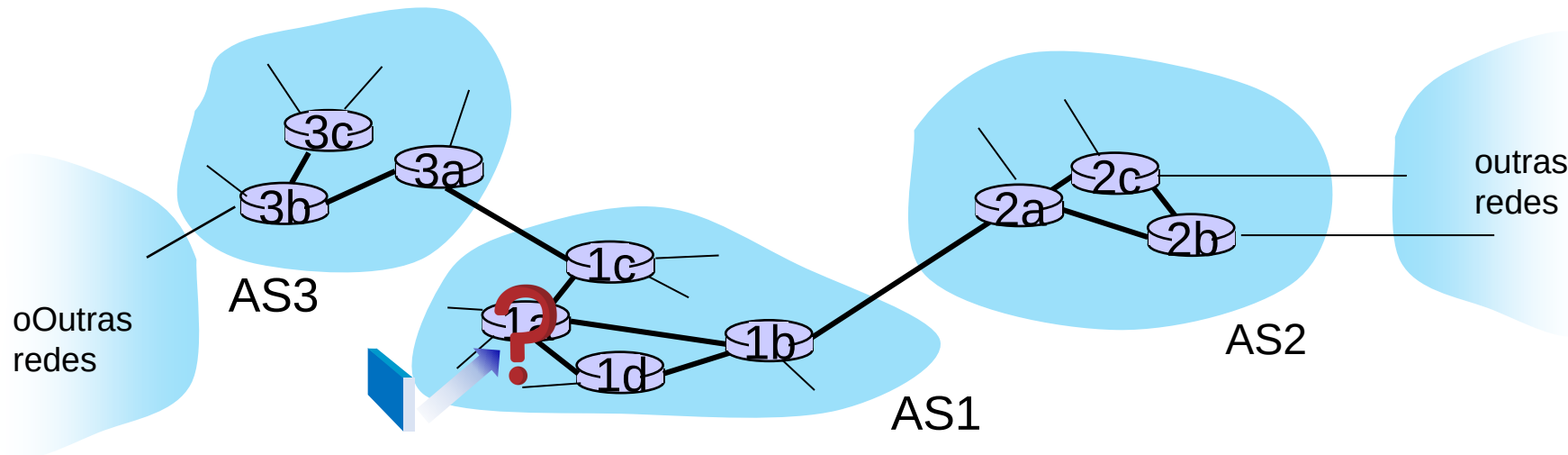
Roteamento Inter-AS: uma função no encaminhamento intradomínio

- suponha que o roteador no AS1 receba um datagrama destinado para fora de AS1 :

? • roteador deve encaminhar o pacote para o roteador gateway em AS1, mas qual deles?

O roteamento entre domínios de AS1 deve:

1. aprender quais destinos são alcançáveis por meio de AS2 e quais são acessíveis por meio de AS3
2. propagar esta informação de acessibilidade para todos os roteadores em AS1



Roteamento Intra-AS: roteando dentro de um AS

protocolos de roteamento intra-AS mais comuns :

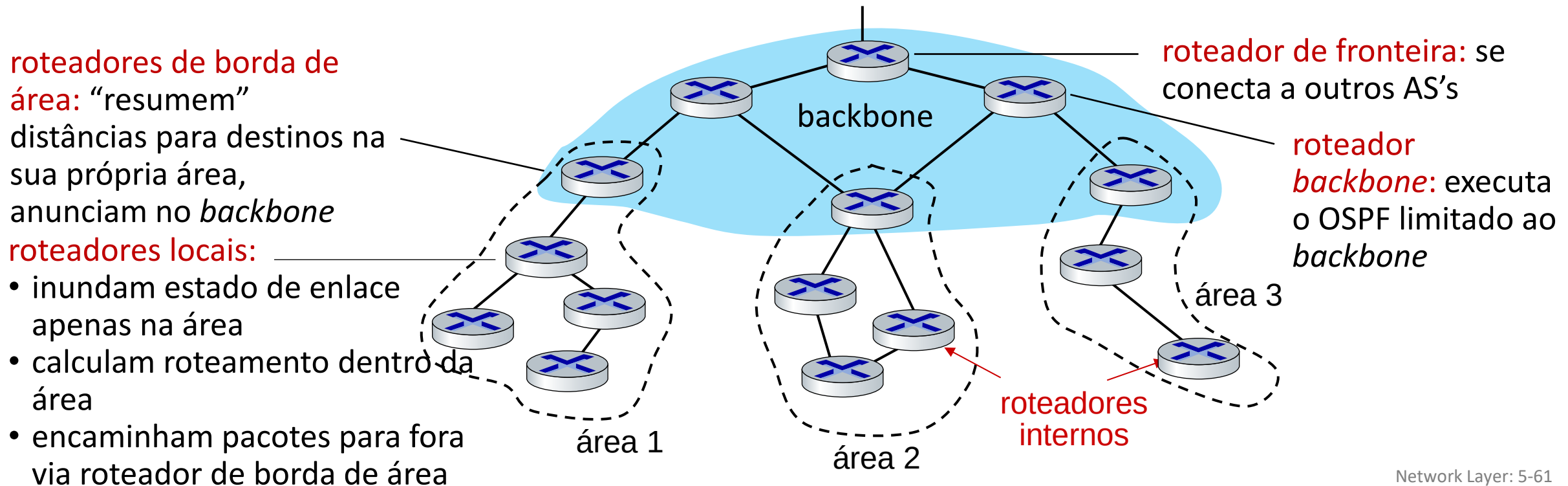
- **RIP: Routing Information Protocol** [RFC 1723]
 - DV clássico: DVs trocados a cada 30 segundos
 - não é mais amplamente utilizado
- **EIGRP: Enhanced Interior Gateway Routing Protocol**
 - Baseado em DV
 - foi proprietário da Cisco por décadas (tornou-se aberto em 2013 [RFC 7868])
- **OSPF: Open Shortest Path First** [RFC 2328]
 - roteamento de estado de enlace
 - Protocolo IS-IS (padrão ISO, não é padrão RFC) essencialmente igual ao OSPF

Roteamento do OSPF (Open Shortest Path First)

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

OSPF hierárquico

- hierarquia de dois níveis : área local e *backbone*.
 - anúncios de estado de enlace inundados apenas na área ou *backbone*
 - cada nó possui topologia de área detalhada; só sabe direção para chegar a outros destinos



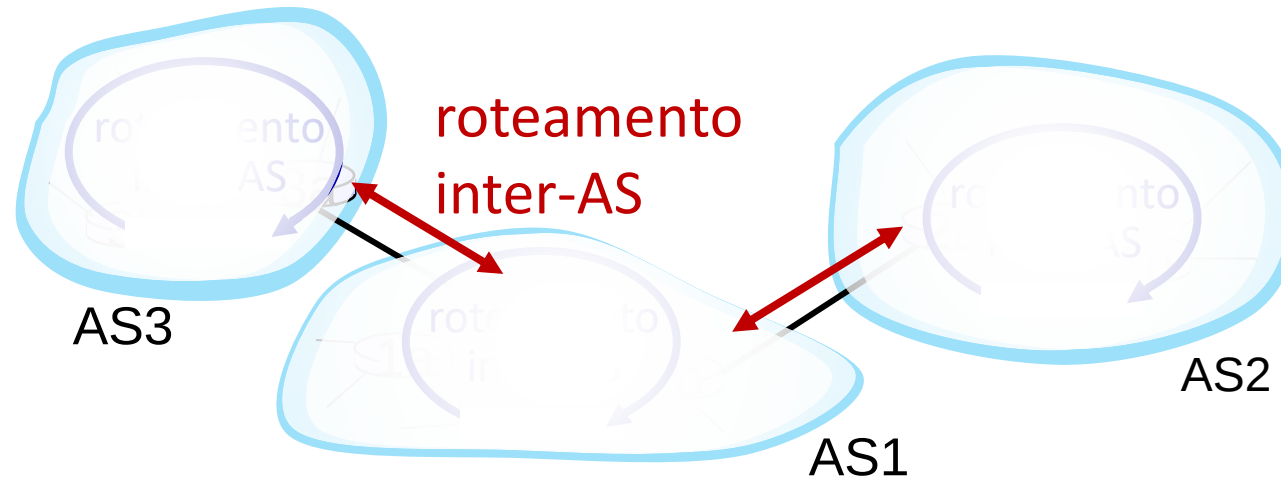
Camada de rede: roteiro do “plano de controle”

- introdução
- protocolos de roteamento
- roteamento intra-ISP: OSPF
- roteamento entre ISPs:
BGP
- plano de controle das SDN
- Internet Control Message Protocol



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

AS's interconectados

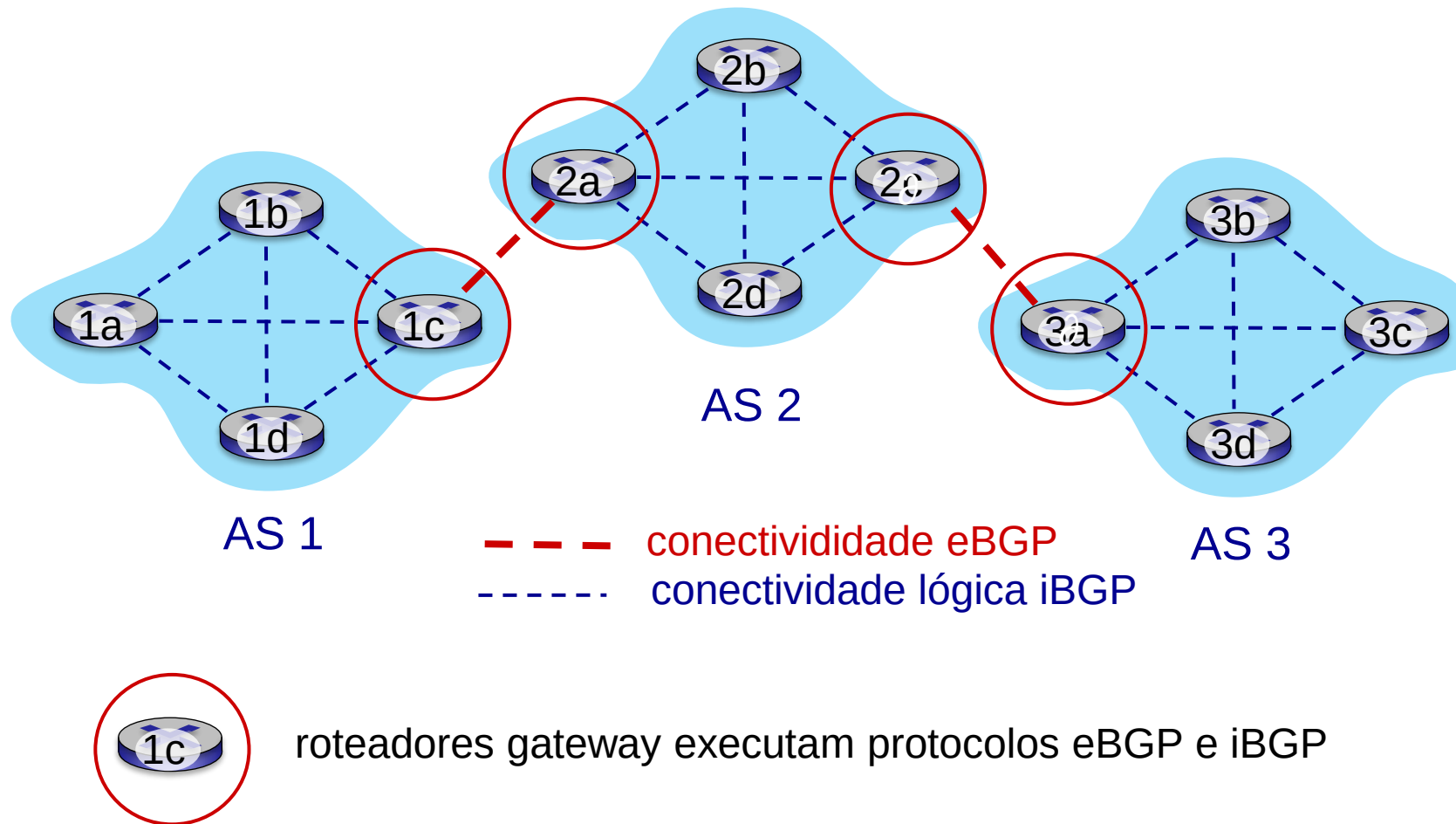


- ✓ **intra-AS** (também chamado “intra-domínio”): roteamento entre roteadores *dentro do mesmo AS* (“rede”)
- ➡ **inter-AS** (também chamado “inter-domínio”): roteamento *entre* AS's

Roteamento inter-AS da Internet: BGP

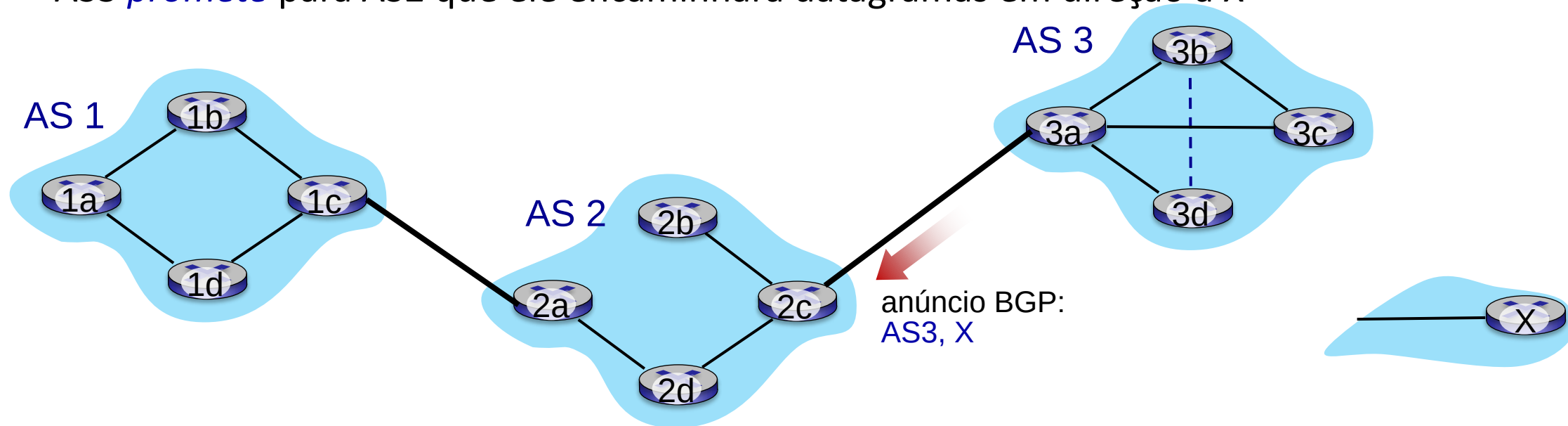
- **BGP (Border Gateway Protocol):** o protocolo de roteamento entre domínios de fato
 - “cola que mantém a Internet unida”
- permite que uma sub-rede anuncie sua existência e os destinos que ela pode alcançar para o resto da Internet : *“Eu estou aqui, aqui está quem eu posso alcançar, e como”*
- O BGP fornece a cada AS um meio para :
 - obter informações de acessibilidade de sub-rede de AS’s 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 de destinos

Conexões eBGP e iBGP



Noções básicas de BGP

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



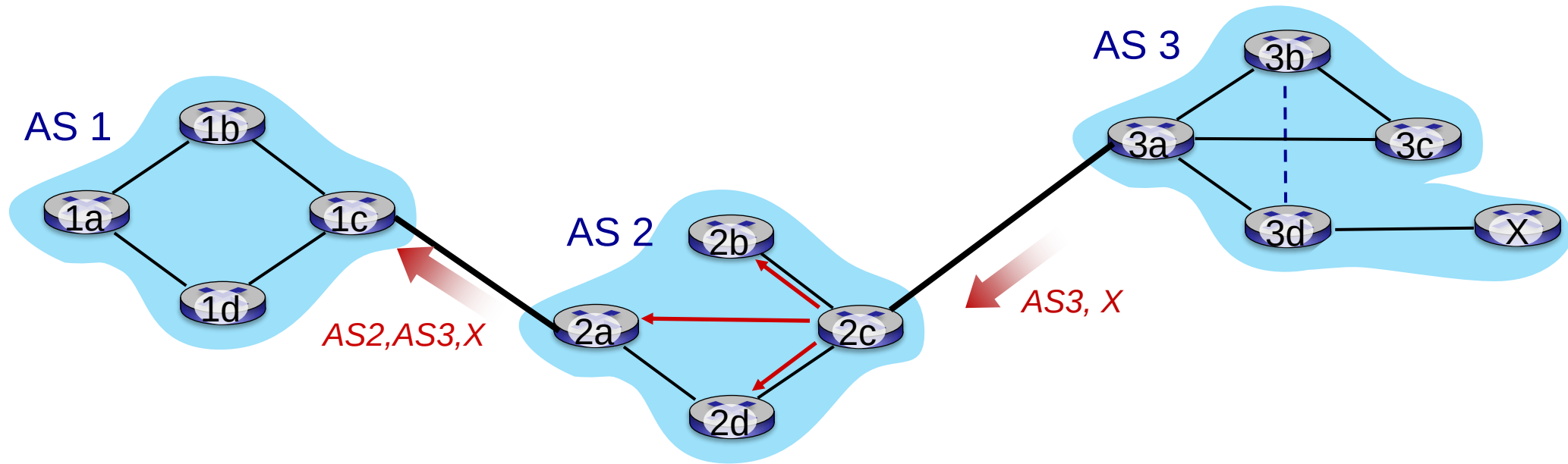
Mensagens BGP

- Mensagens BGP trocadas entre pares por conexão TCP
- Mensagens BGP [RFC 4371]:
 - **OPEN**: abre a conexão TCP com o par BGP remoto e autentica o par BGP emissor
 - **UPDATE**: anuncia novo caminho (ou retira antigo)
 - **KEEPALIVE**: mantém a conexão ativa na ausência de UPDATES; também envia ACKs para solicitações OPEN
 - **NOTIFICATION**: reporta erros em mensagem anterior; também usado para fechar a conexão

Atributos de caminho e rotas BGP

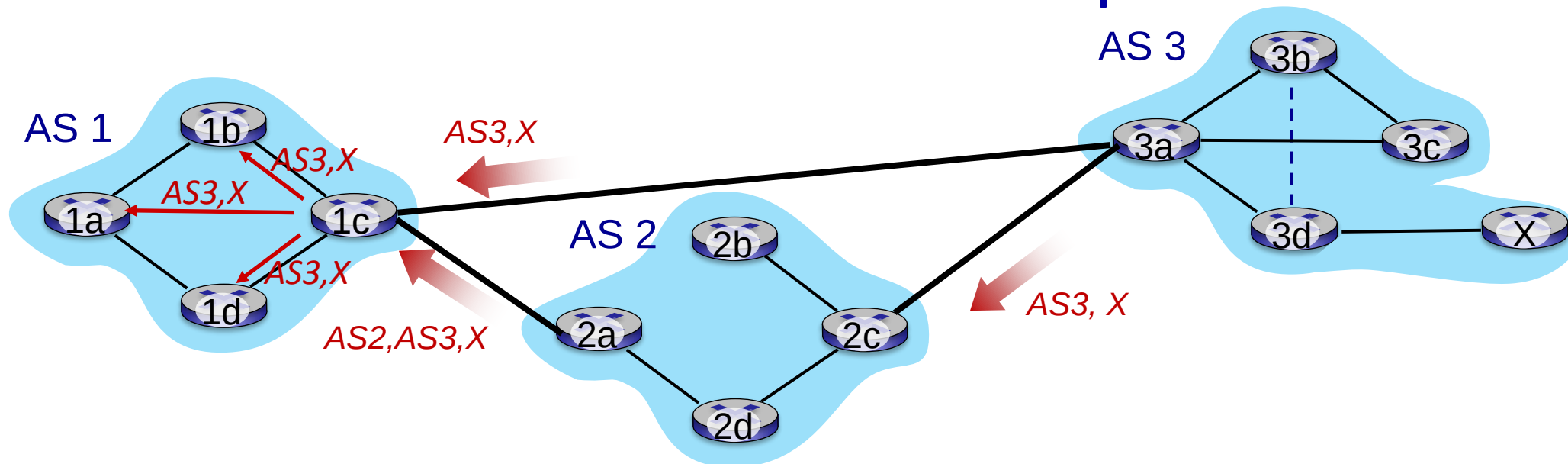
- Rota anunciada do BGP: prefixo + atributos
 - prefixo: destino sendo anunciado
 - dois atributos importantes:
 - **AS-PATH**: lista de AS's pelos quais o anúncio de prefixo passou
 - **NEXT-HOP**: indica roteador AS interno específico para AS de próximo salto
- **roteamento baseado em política :**
 - gateway recebendo anúncio de rota usa *política de importação* para aceitar/recusar o caminho (por exemplo, nunca rotear por AS Y)
 - A política de AS também determina se um determinado caminho será *anunciado* para outros AS's vizinhos

Anúncio de caminho do BGP



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

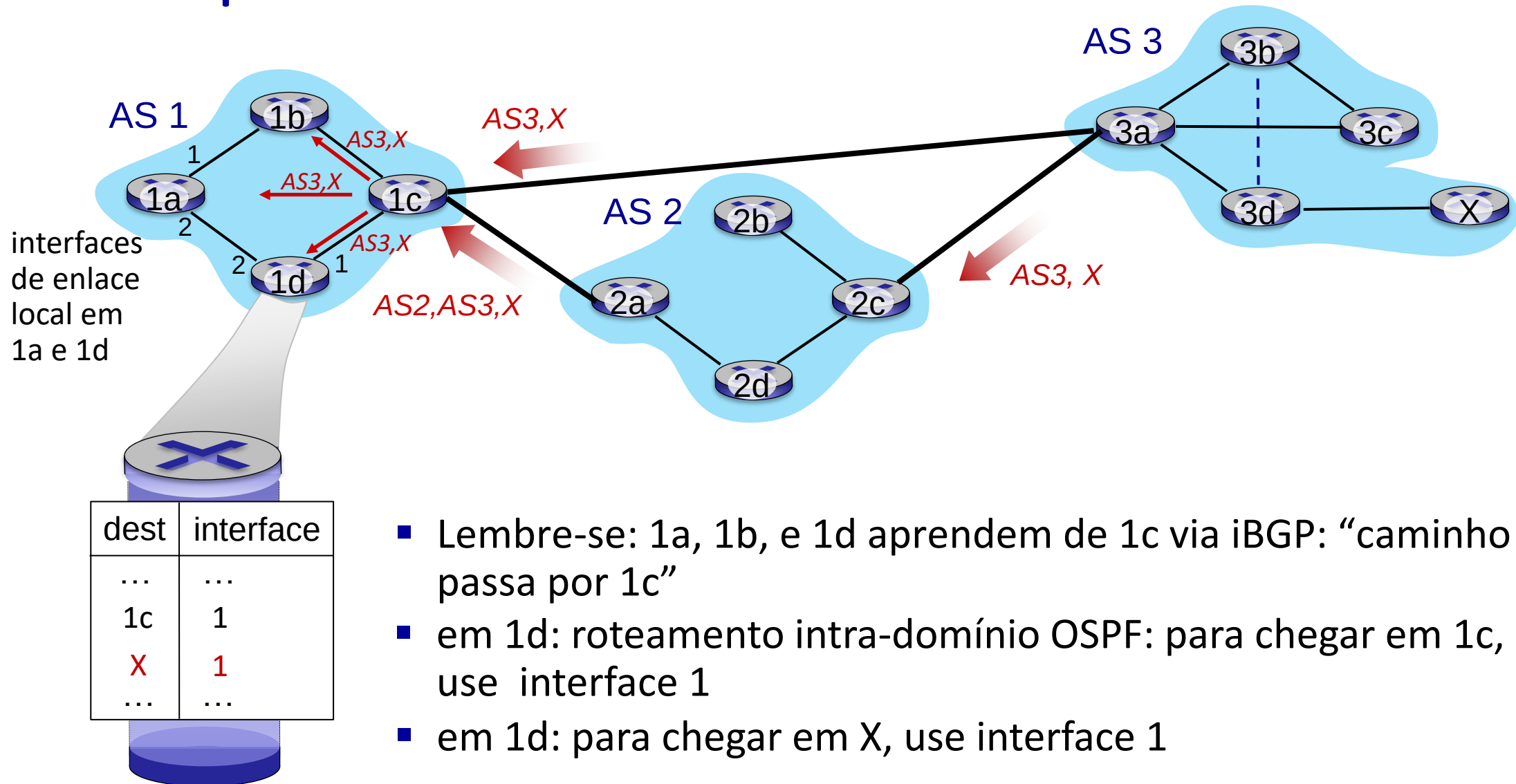
Anúncio de caminho do BGP: múltiplos caminhos



roteador gateway pode aprender sobre **múltiplos** caminhos para o destino :

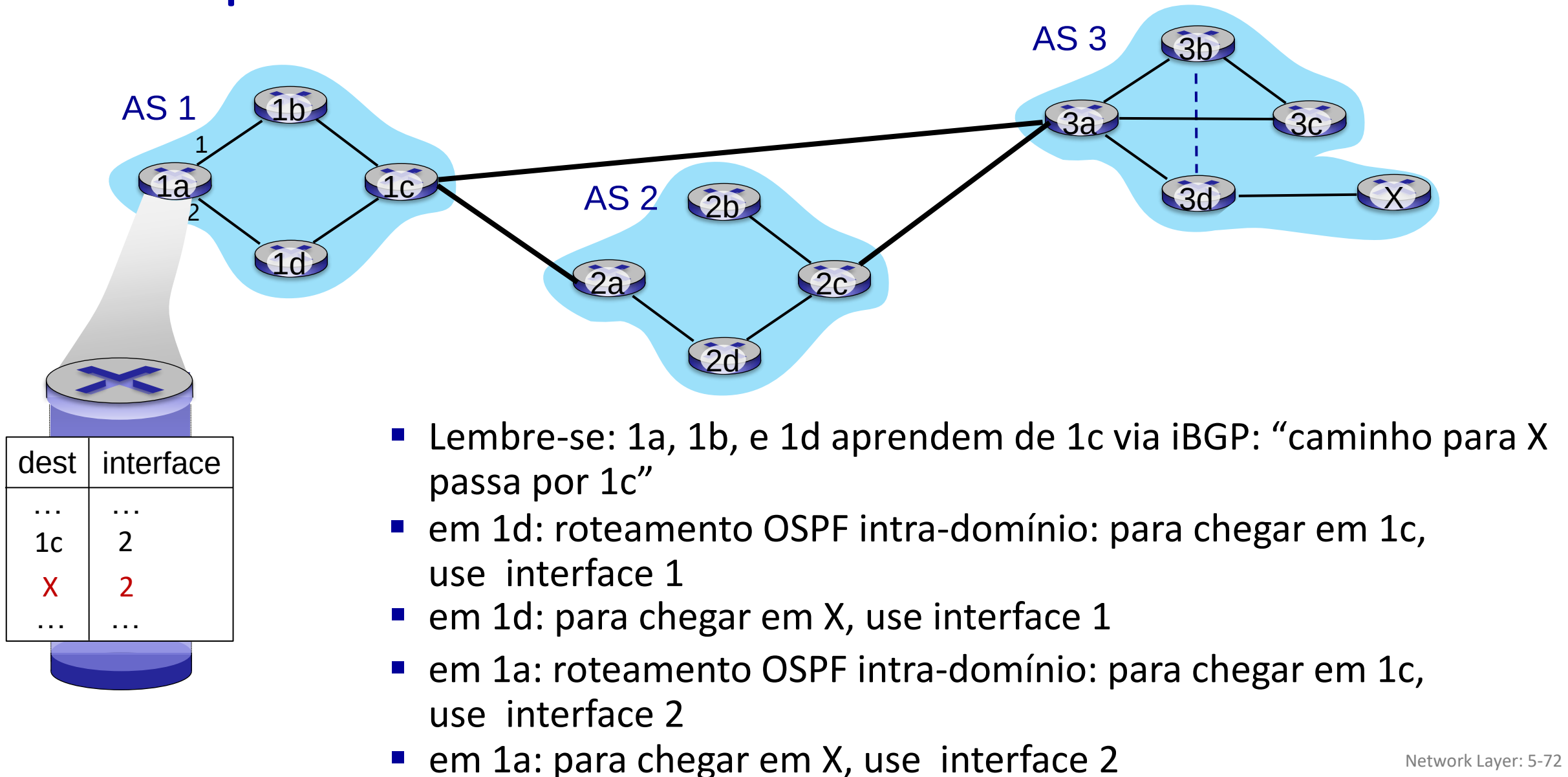
- roteador gateway 1c de AS1 aprende caminho **AS2,AS3,X** de 2a
- roteador gateway 1c de AS1 aprende caminho **AS3,X** de 3a
- baseado em **política**, roteador gateway 1c de AS1 escolhe caminho **AS3,X** e anuncia caminho dentro de AS1 via iBGP

BGP: preenchendo tabelas de encaminhamento

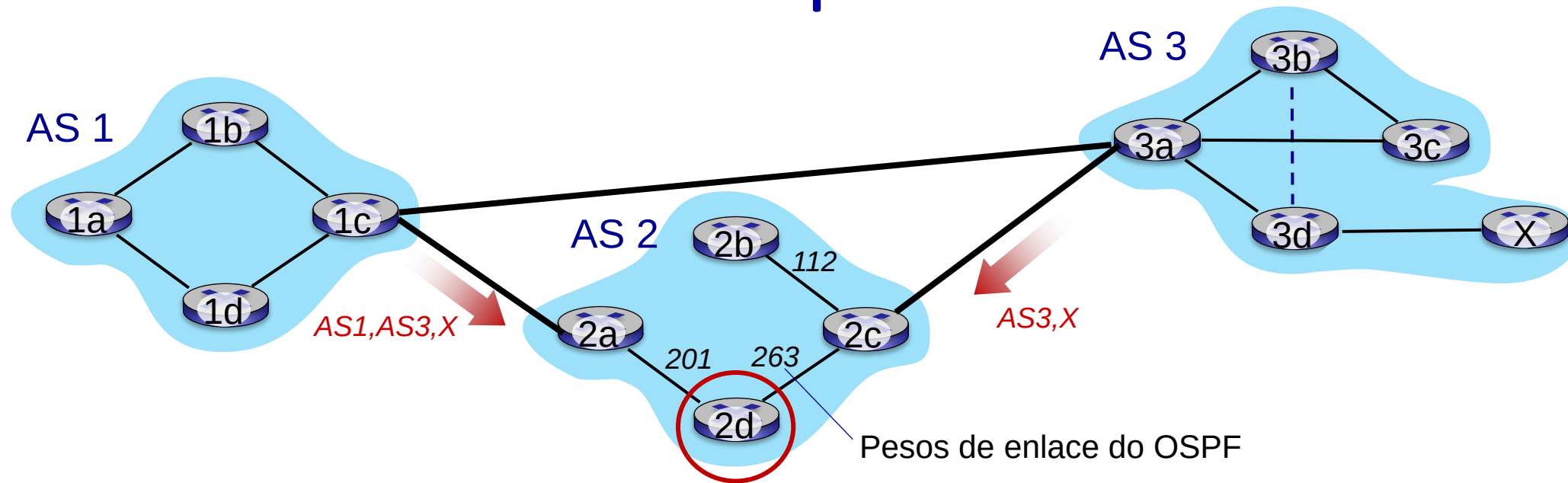


- Lembre-se: 1a, 1b, e 1d aprendem de 1c via iBGP: “caminho para X passa por 1c”
- em 1d: roteamento intra-domínio OSPF: para chegar em 1c, use interface 1
- em 1d: para chegar em X, use interface 1

BGP: preenchendo tabelas de encaminhamento

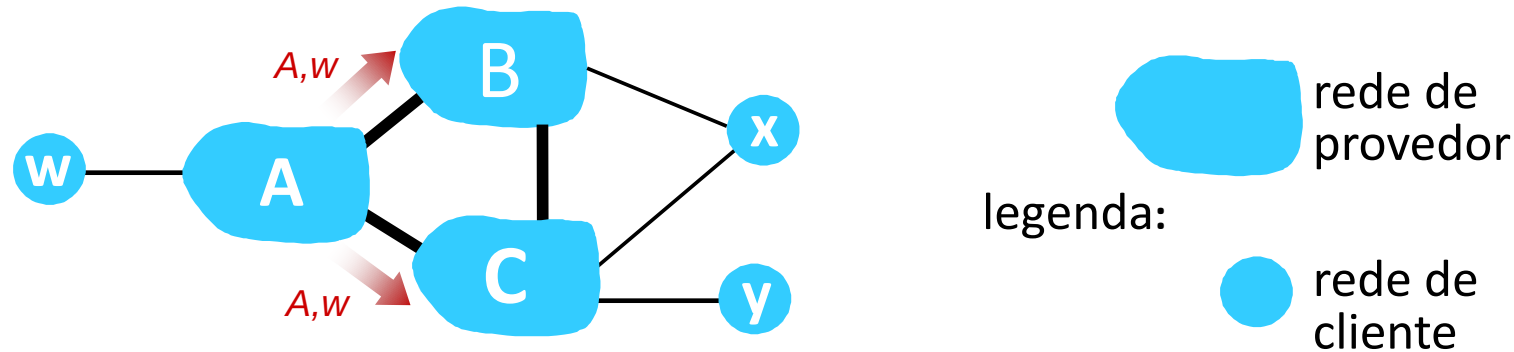


Roteamento de batata quente



- 2d aprende (via iBGP) que ele pode rotear para X via 2a ou 2c
- **roteamento de batata quente:** escolhe o gateway local que tenha o menor custo *intra-domínio* (por exemplo, 2d escolhe 2a, embora ele tenha mais saltos de AS para chegar em X): não se preocupe com o custo entre domínios!

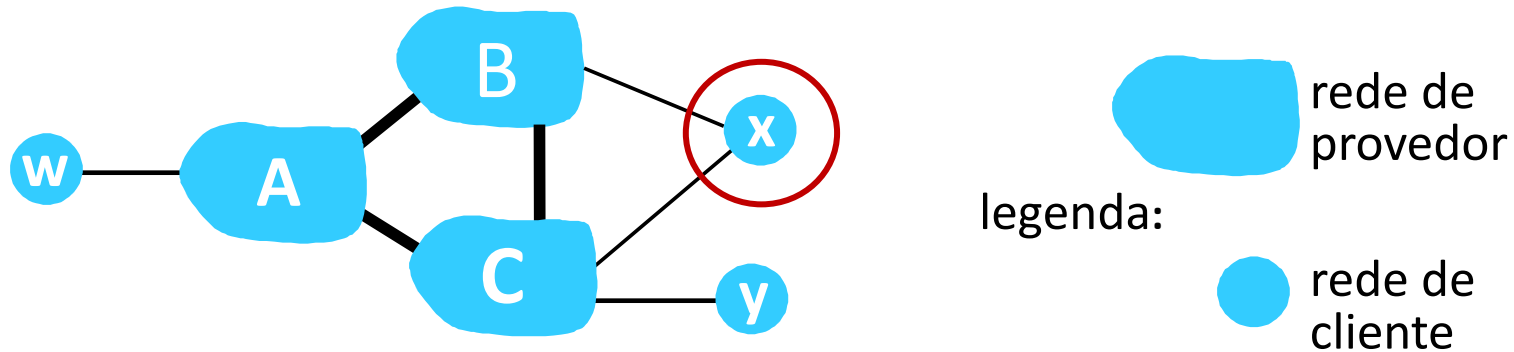
BGP: cumprindo a política por meio de anúncios



O ISP só quer rotear o tráfego de/para redes de seus clientes (não quer transportar tráfego de trânsito entre outros ISPs – uma política típica do “mundo real”)

- A anuncia caminho Aw para B e para C
- B *escolhe não anunciar* BA_w para C!
 - B não recebe “receita” por rotear CBA_w, pois nenhum deles (C, A, w) é cliente de B
 - C *não* aprende sobre o caminho CBA_w
- C roteará via CA_w (sem usar B) para chegar em w

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



O ISP só quer rotear o tráfego de/para redes de seus clientes (não quer transportar 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 em duas redes
- **política para impor**: x não quer que B roteie para C via x
 - .. então x não anunciará para B que tem uma rota para C

Seleção de rota do BGP

- roteador pode aprender sobre mais de uma rota para o AS de destino, seleciona a rota com base em:
 1. atributo de valor de preferência local: decisão de política
 2. AS-PATH mais curto
 3. roteador NEXT-HOP mais próximo: roteamento de batata quente
 4. critérios adicionais

Por que diferentes roteamentos Intra-AS e Inter-AS?

política:

- inter-AS: o administrador quer controlar como o tráfego é roteado, quem roteia pela sua rede
- intra-AS: administrador único, portanto, a política é menos problemática

escala:

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

desempenho:

- intra-AS: pode focar no desempenho
- inter-AS: política domina 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 das SDN**
- Internet Control Message Protocol



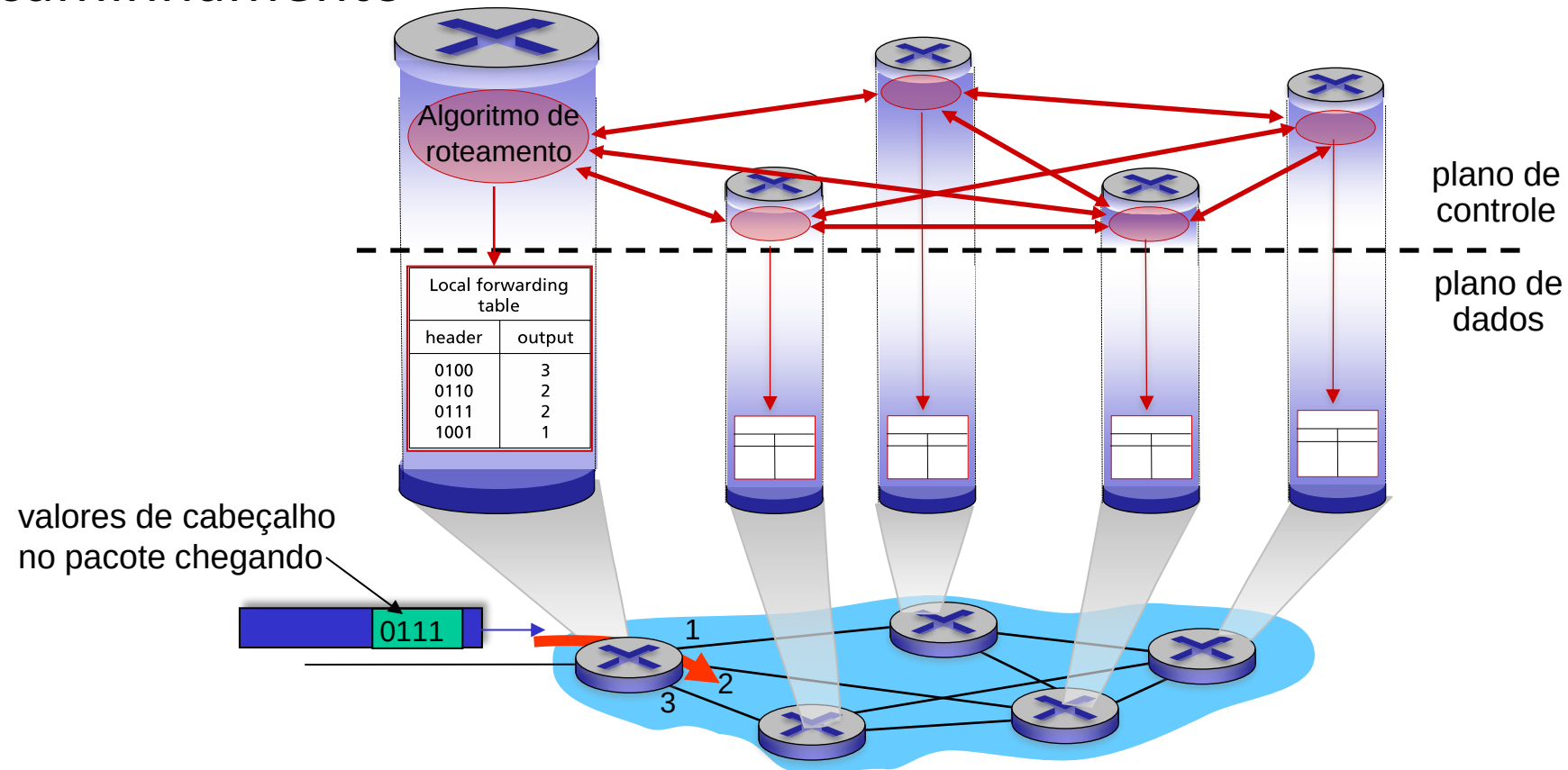
- gerenciamento e configuração de rede
 - SNMP
 - NETCONF/YANG

Software defined networking (SDN)

- Camada de rede da Internet: historicamente implementada por meio de abordagem de controle distribuído por roteador:
 - roteador *monolítico* contém hardware de comutação, executa implementação proprietária de protocolos padrão da Internet (IP, RIP, IS-IS, OSPF, BGP) no SO de roteador proprietário (por exemplo, Cisco IOS)
 - diferentes “middleboxes” para diferentes funções da camada de rede: firewalls, balanceadores de carga, caixas NAT, ..
- ~2005: interesse renovado em repensar o plano de controle de rede

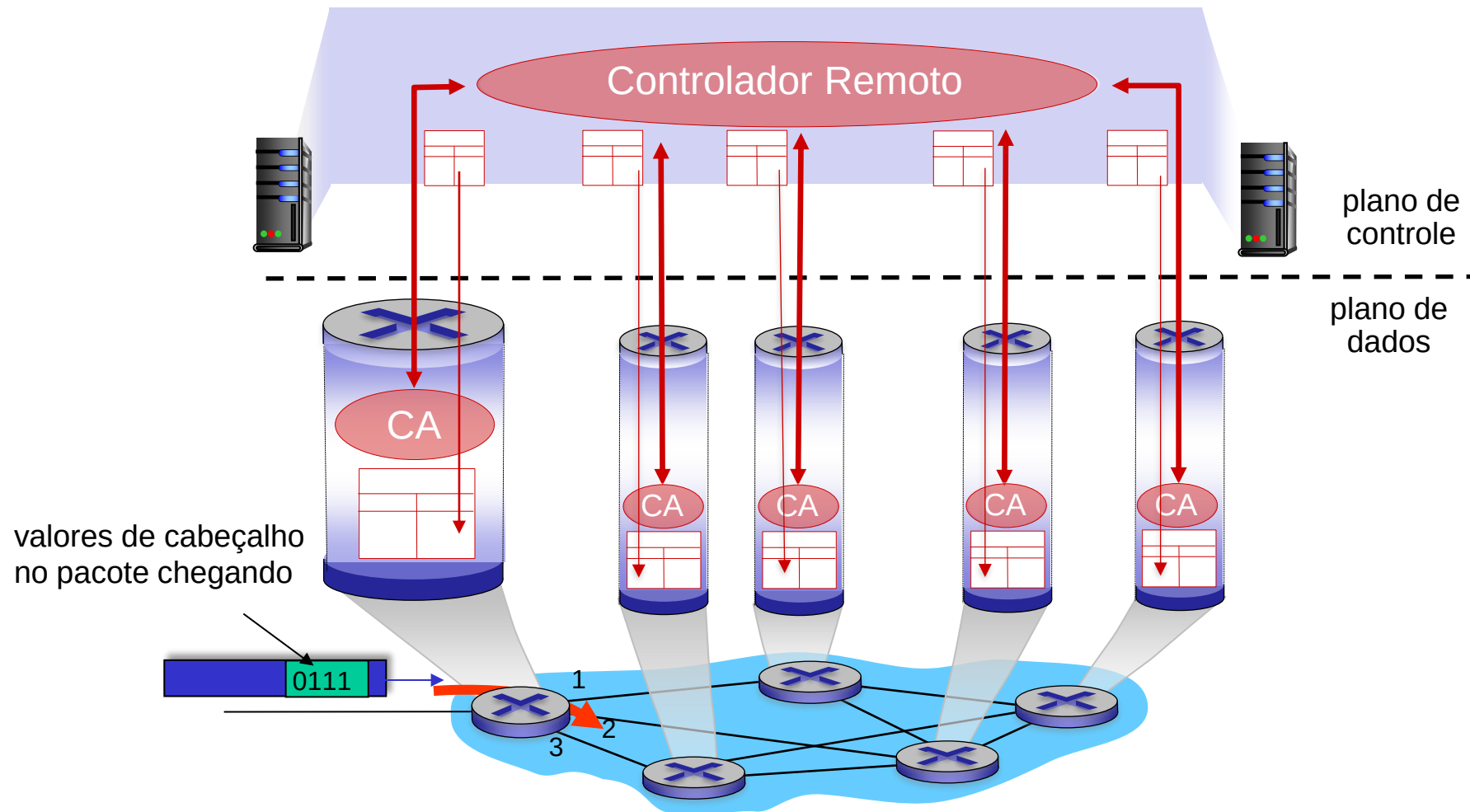
Plano de controle por roteador

Componentes individuais do algoritmo de roteamento *em todo e cada roteador* interagem no plano de controle para computar as tabelas de encaminhamento



Plano de controle de Rede Definida por Software (SDN – Software Defined Network)

O controlador remoto calcula e instala tabelas de encaminhamento em roteadores

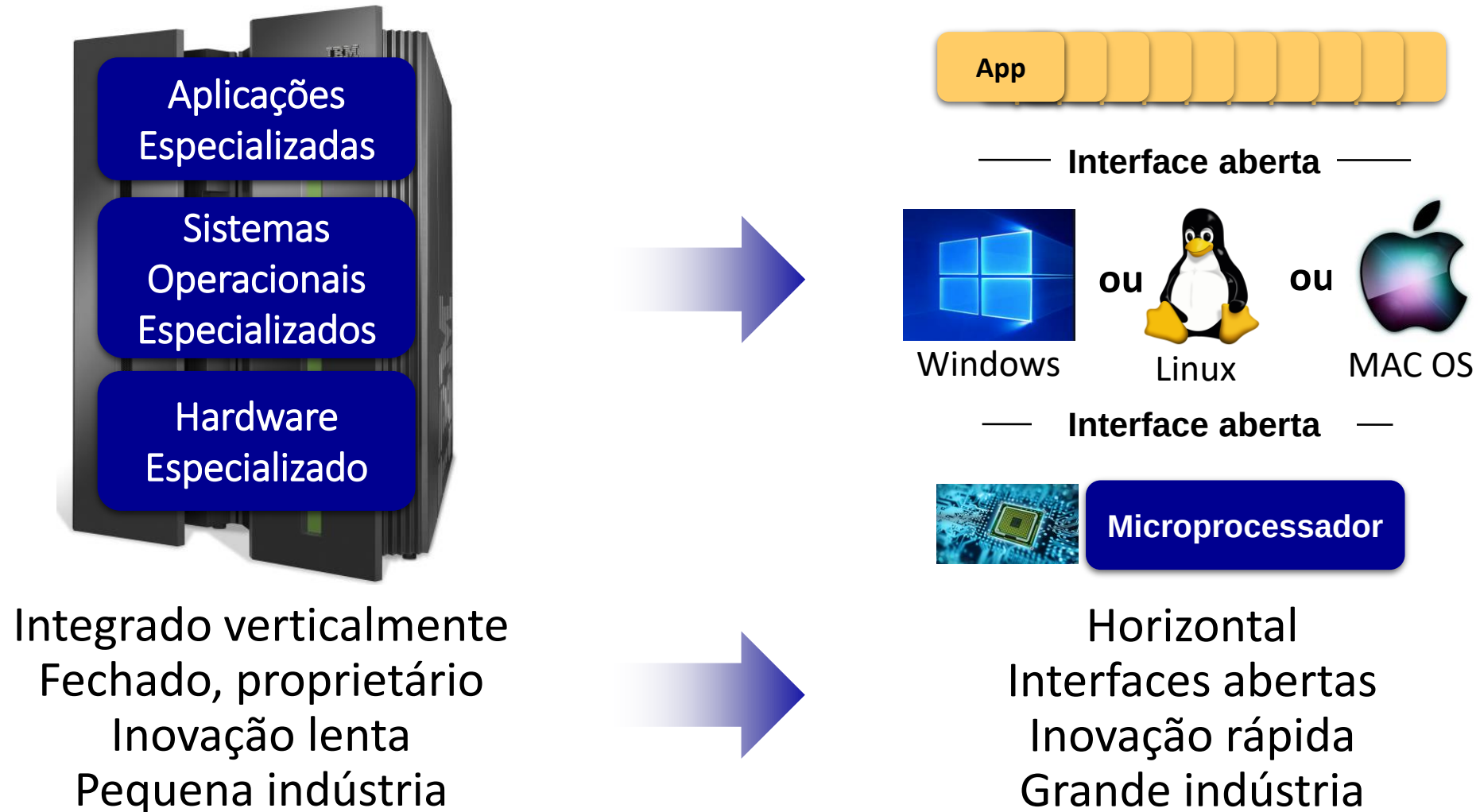


Software defined networking (SDN)

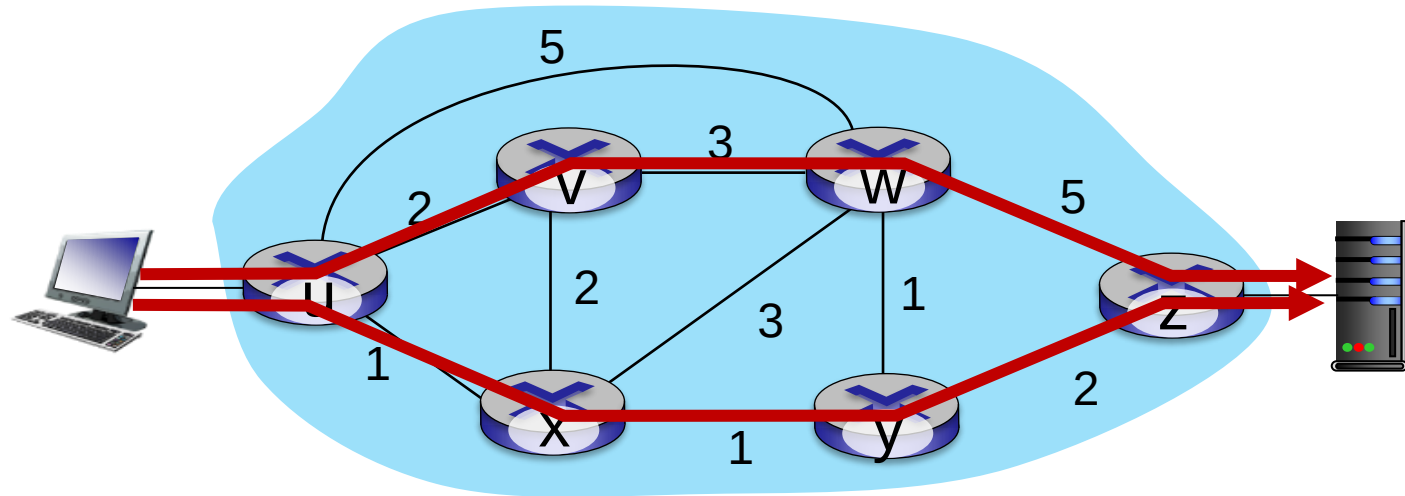
Por que um plano de controle *logicamente centralizado*?

- gerenciamento de rede mais fácil: evita configurações incorretas de roteadores, maior flexibilidade nos fluxos de tráfego
- encaminhamento baseado em tabela (lembre-se da API OpenFlow) permite “programação” de roteadores
 - “programação” centralizada é mais fácil: calcule tabelas centralmente e distribua
 - “programação” distribuída é mais difícil: calcule tabelas como resultado de algoritmo distribuído (protocolo) implementado em cada roteador
- implementação aberta (não proprietária) do plano de controle
 - inovação mais rápida: encoraja novas ideias de muitas fontes

Analogia SDN: revolução do mainframe para o PC



Engenharia de tráfego: difícil com roteamento tradicional



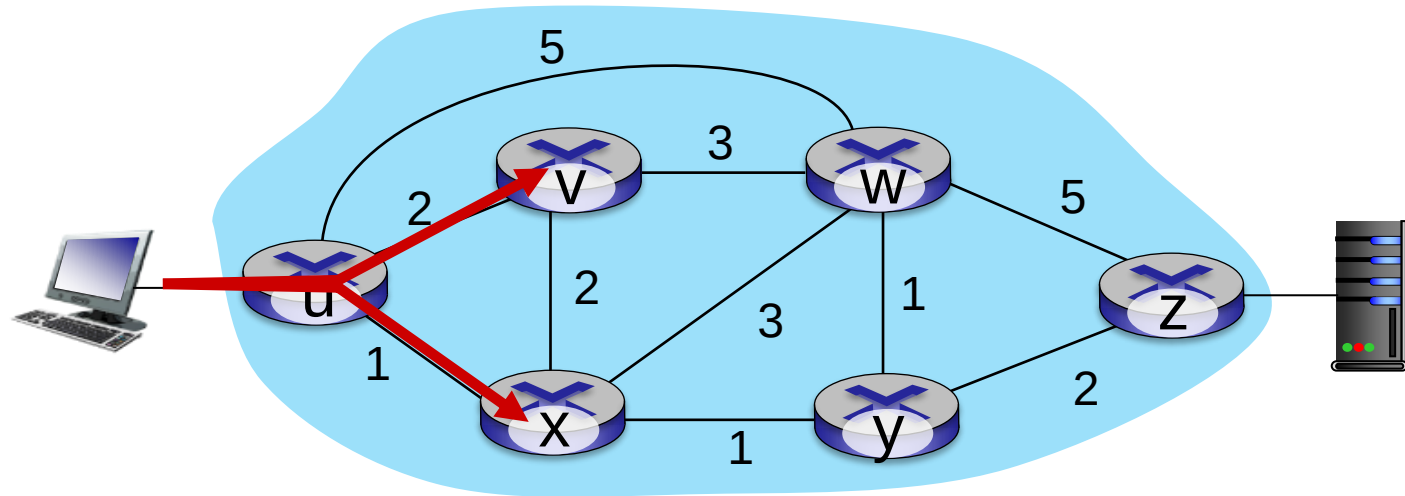
Q: e se o operador de rede quiser que o tráfego de u-para-z flua ao longo de $uvwz$, em vez de $uxyz$?

R: precisa redefinir os pesos dos enlaces 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 “knobs”: não há muito controle!



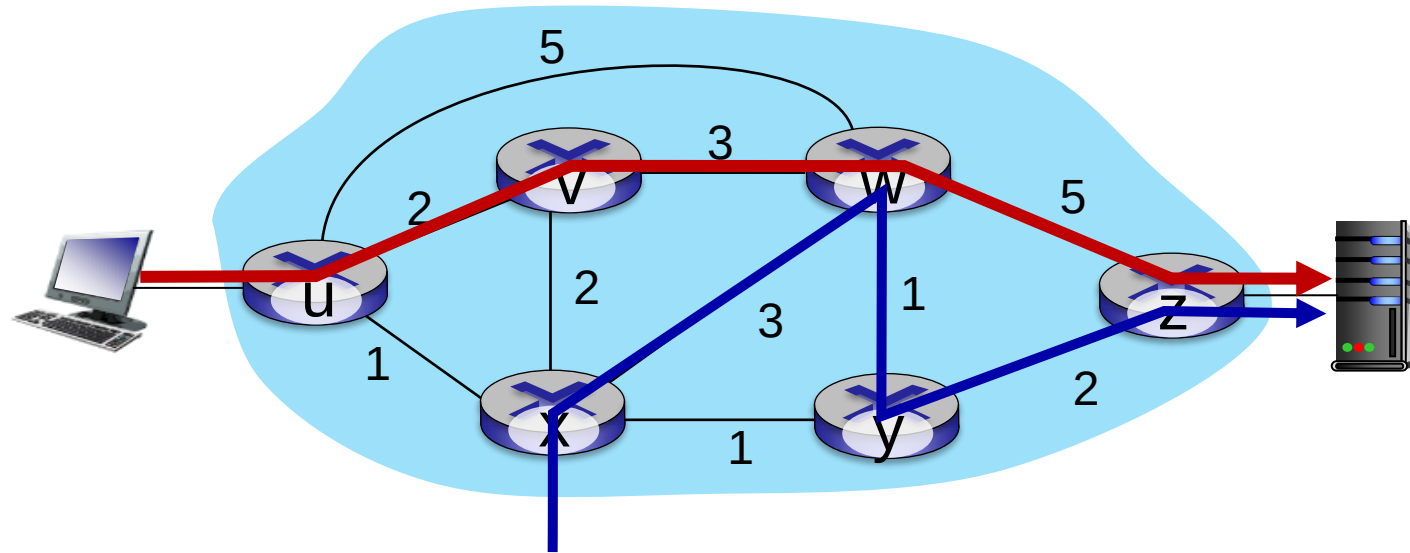
Engenharia de tráfego: difícil com roteamento tradicional



Q: e se o operador de rede quiser dividir o tráfego u-para-z entre uvwz *e* uxyz (balanceamento de carga)?

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

Engenharia de tráfego: difícil com roteamento tradicional

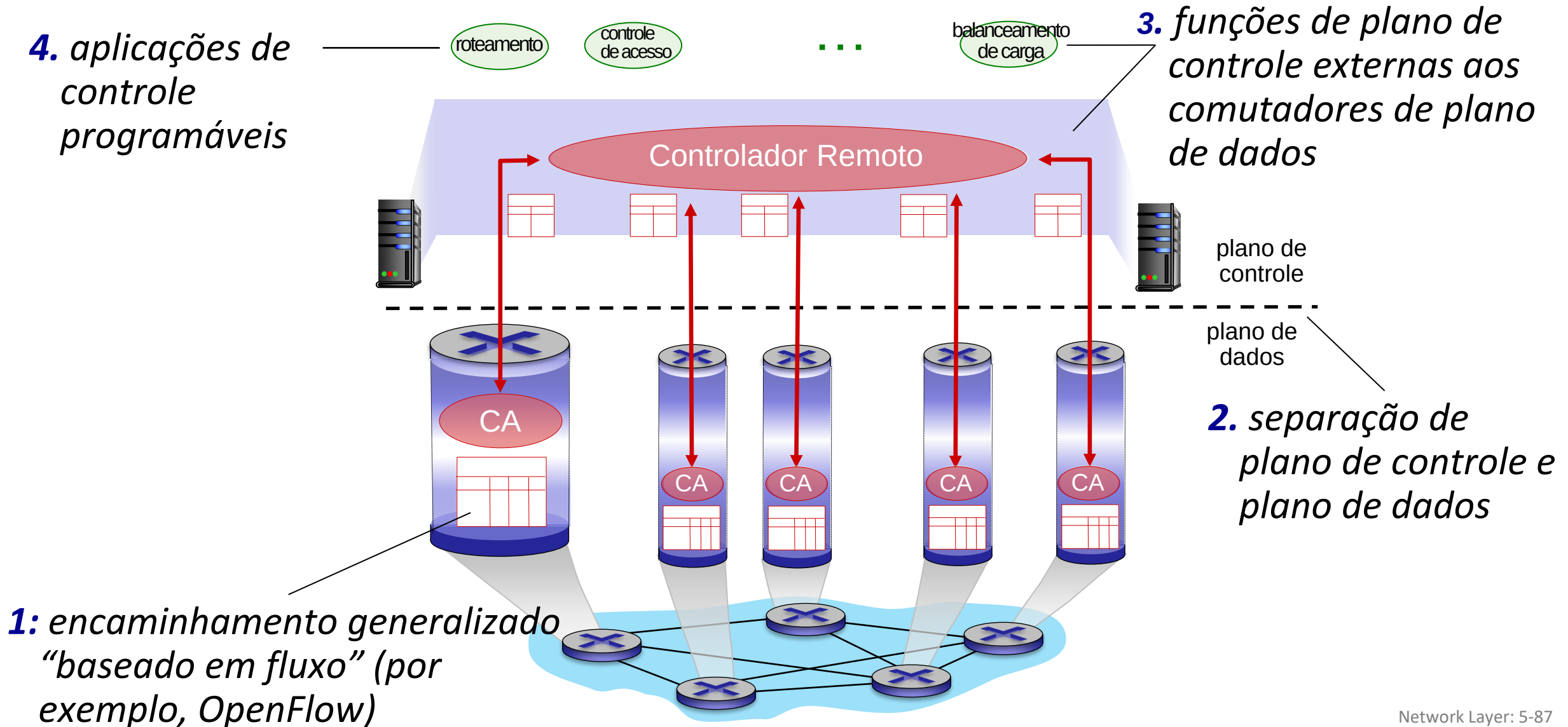


Q: e se w quisesse direcionar o tráfego azul e vermelho de forma diferente de w para z?

R: não pode fazer isso (com encaminhamento baseado em destino e roteamento de estado de enlace ou vetor de distância)

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

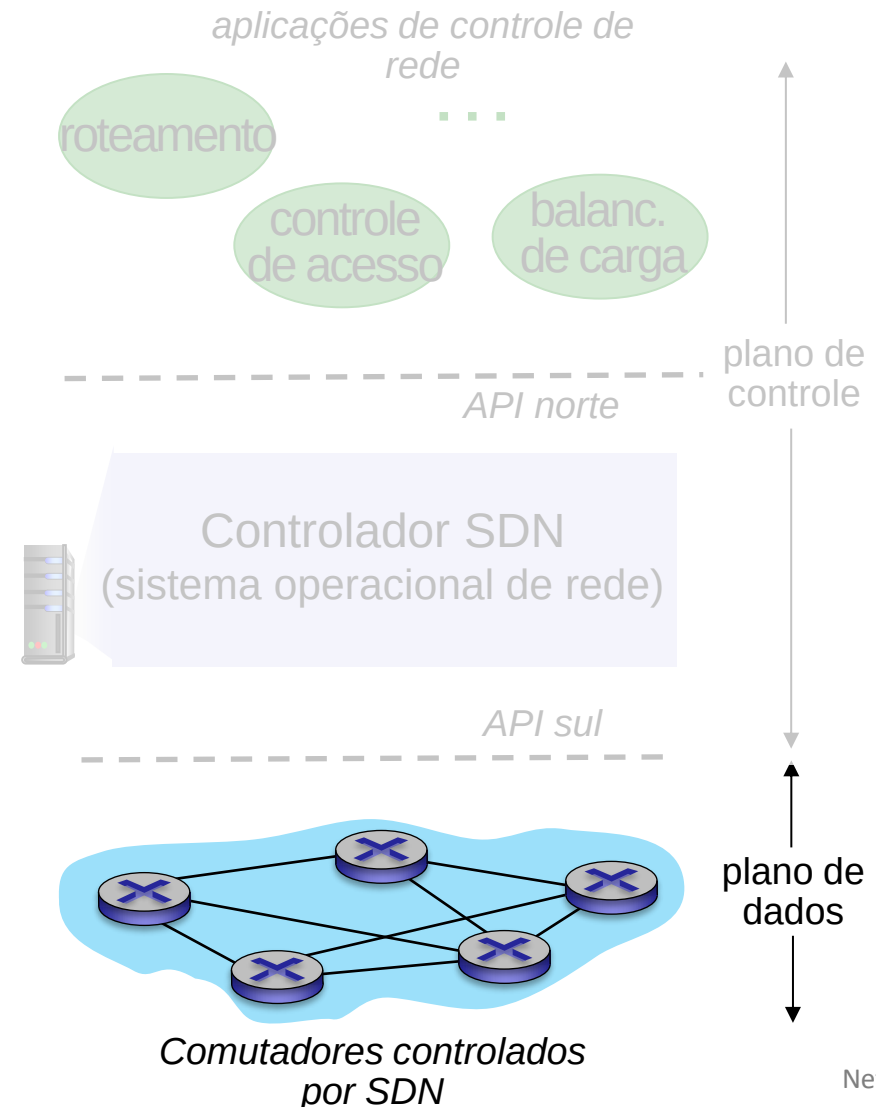
Software defined networking (SDN)



Software defined networking (SDN)

Comutadores de plano de dados:

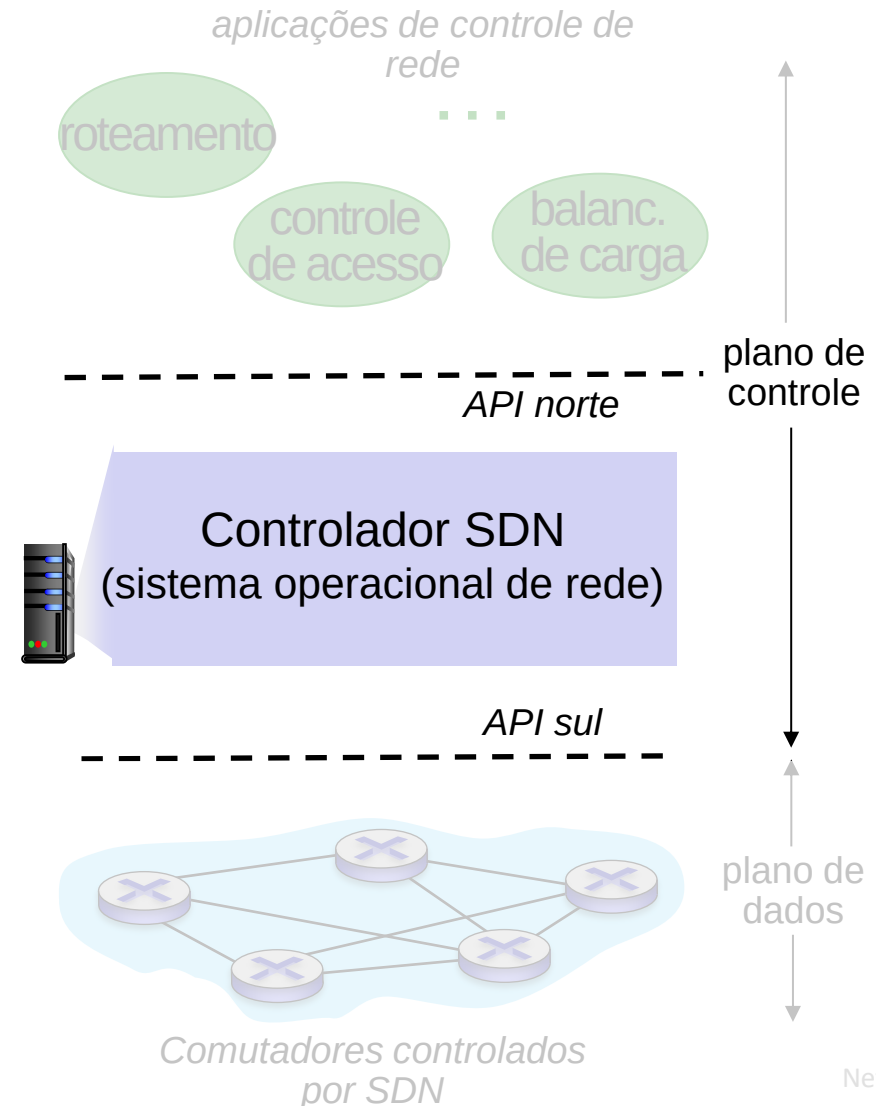
- switches simples e rápidos que implementam o encaminhamento generalizado do plano de dados (Seção 4.4) em hardware
- tabela de fluxo (encaminhamento) computada e instalada sob supervisão do controlador
- API para controle de comutação baseado em tabela (por exemplo, OpenFlow)
 - define o que é controlável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



Software defined networking (SDN)

Controlador SDN (SO de rede):

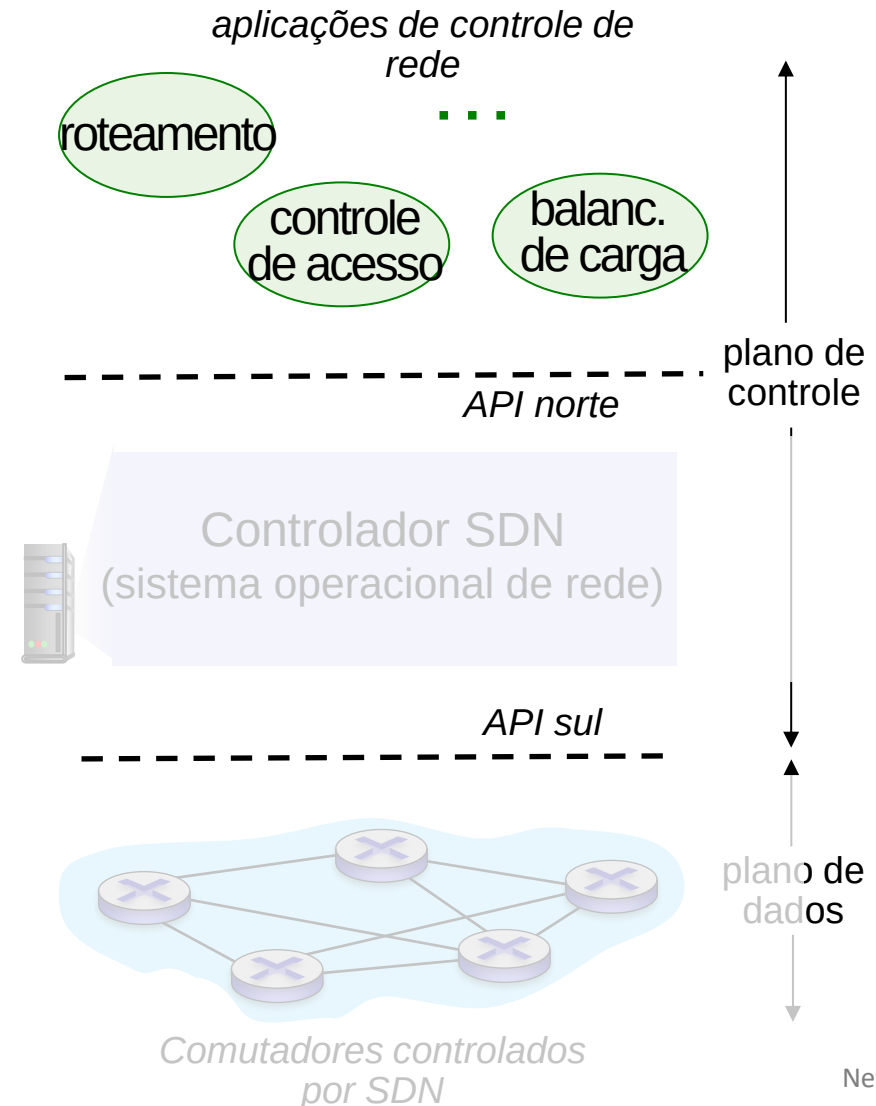
- mantem as informações do estado da rede
- interage com aplicativos de controle de rede “acima” via API norte
- interage com comutadores de rede “abaixo” via API sul
- implementado como sistema distribuído para desempenho, escalabilidade, tolerância a falhas, e robustez



Software defined networking (SDN)

aplicativos de controle de rede:

- “cérebros” de controle: implementam funções de controle usando serviços de baixo nível e a API fornecida pelo controlador SDN
- *desagregado*: pode ser fornecido por terceiros: distintos do fornecedor de roteamento ou controlador SDN

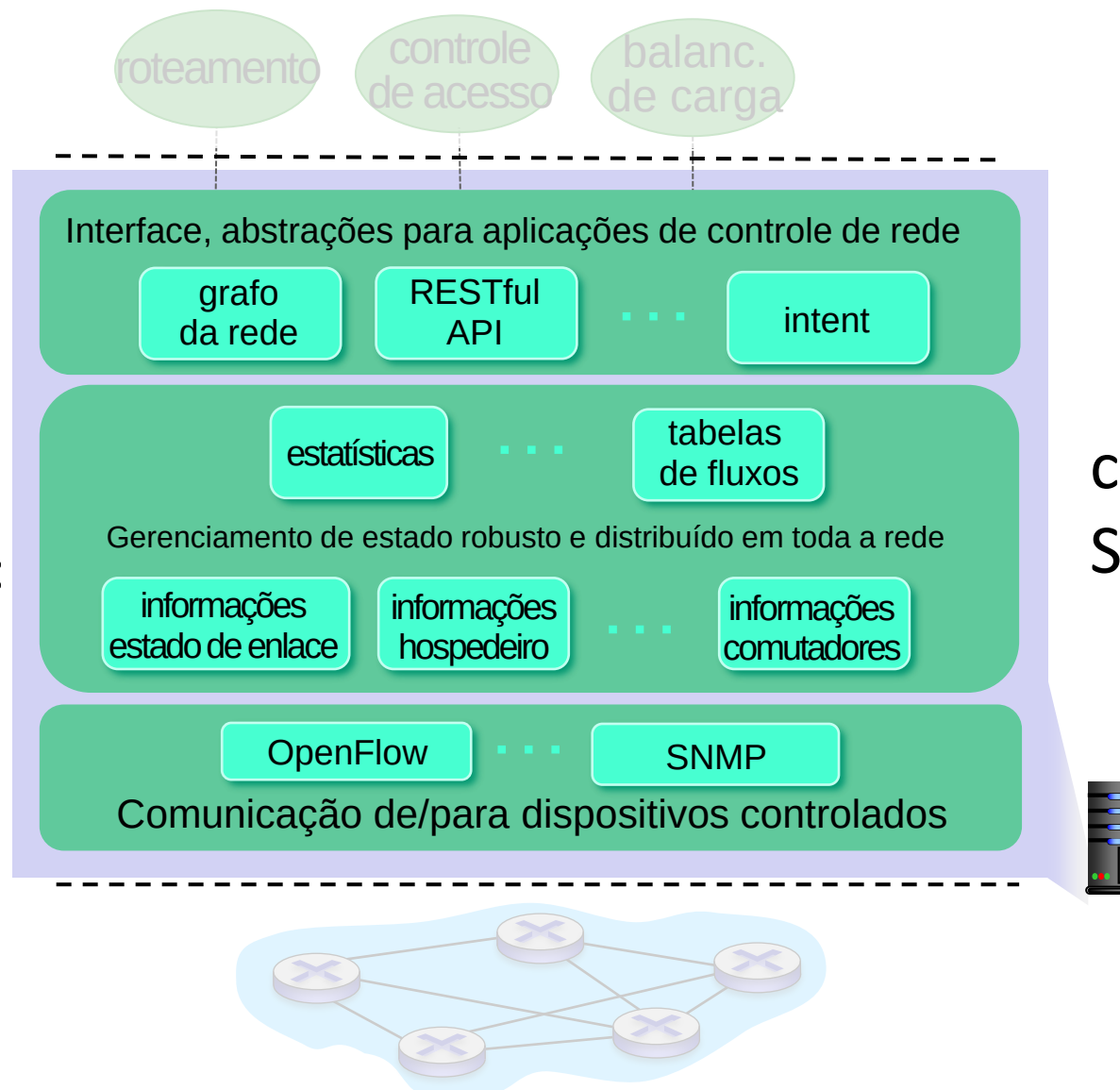


Componentes de um controlador SDN

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

gerenciamento de estado em toda a rede: estado das redes, enlaces, comutadores, serviços:
um *banco de dados distribuído*

comunicação: comunicação entre o controlador SDN e os comutadores controlados

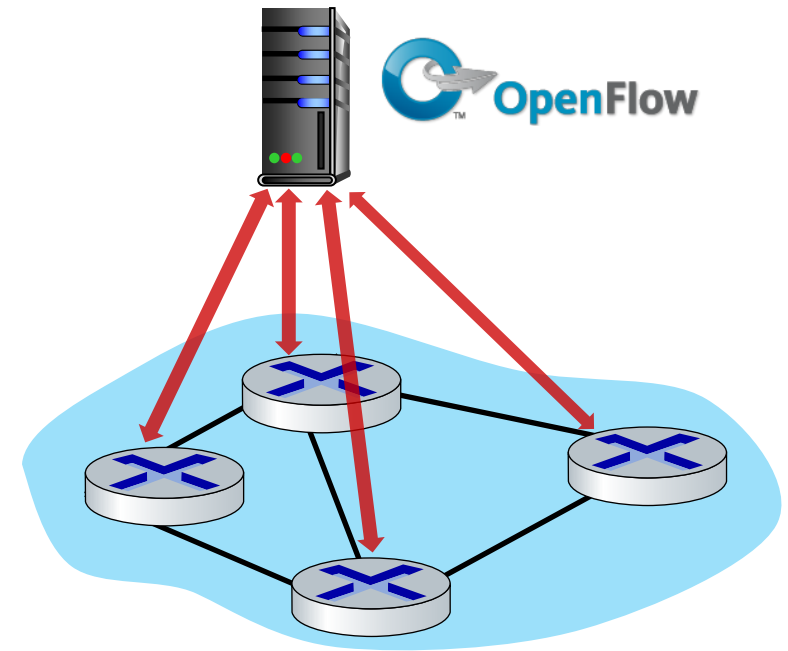


controlador
SDN

Protocolo OpenFlow

- opera entre controlador e comutadores
- TCP usado para trocar mensagens
 - criptografia opcional
- três classes de mensagens OpenFlow:
 - controlador-para-comutador
 - assíncronas (comutador para controlador)
 - simétricas (misc.)
- distinto da API OpenFlow
 - API usada para especificar ações de encaminhamento generalizadas

Controlador OpenFlow

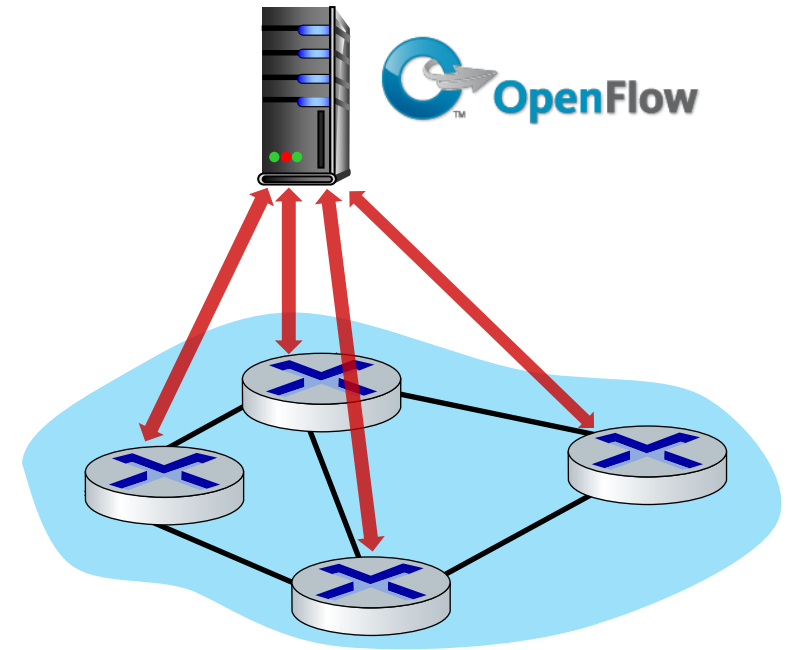


OpenFlow: mensagens controlador-para-comutador

Mensagens chave do controlador para o comutador

- *features*: o controlador consulta recursos do comutador, o comutador responde
- *configure*: o controlador consulta/define os parâmetros de configuração do comutador
- *modify-state*: adiciona, exclui, modifica entradas de fluxo nas tabelas OpenFlow
- *packet-out*: o controlador pode enviar este pacote a partir de uma porta específica do comutador

Controlador OpenFlow



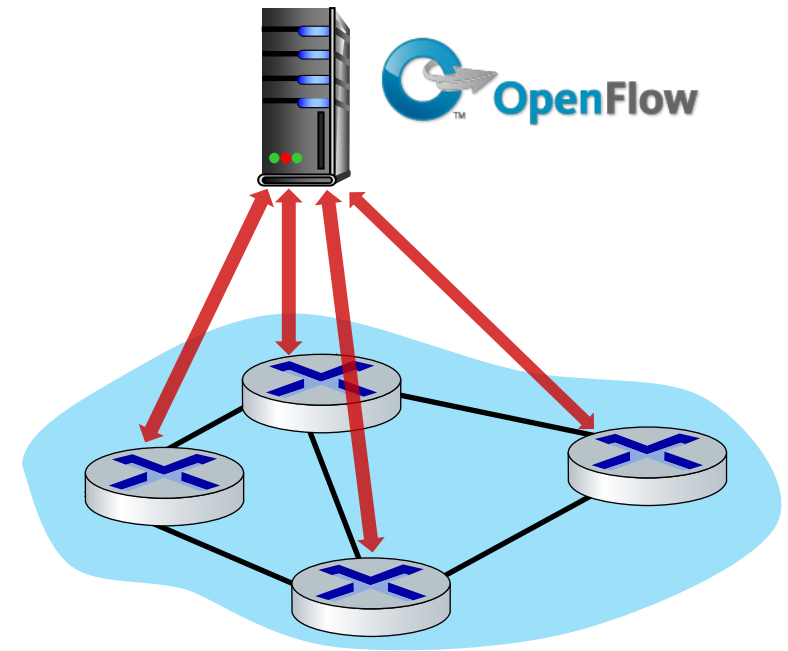
OpenFlow: mensagens comutador-para-controlador

Mensagens chave do comutador para o controlador

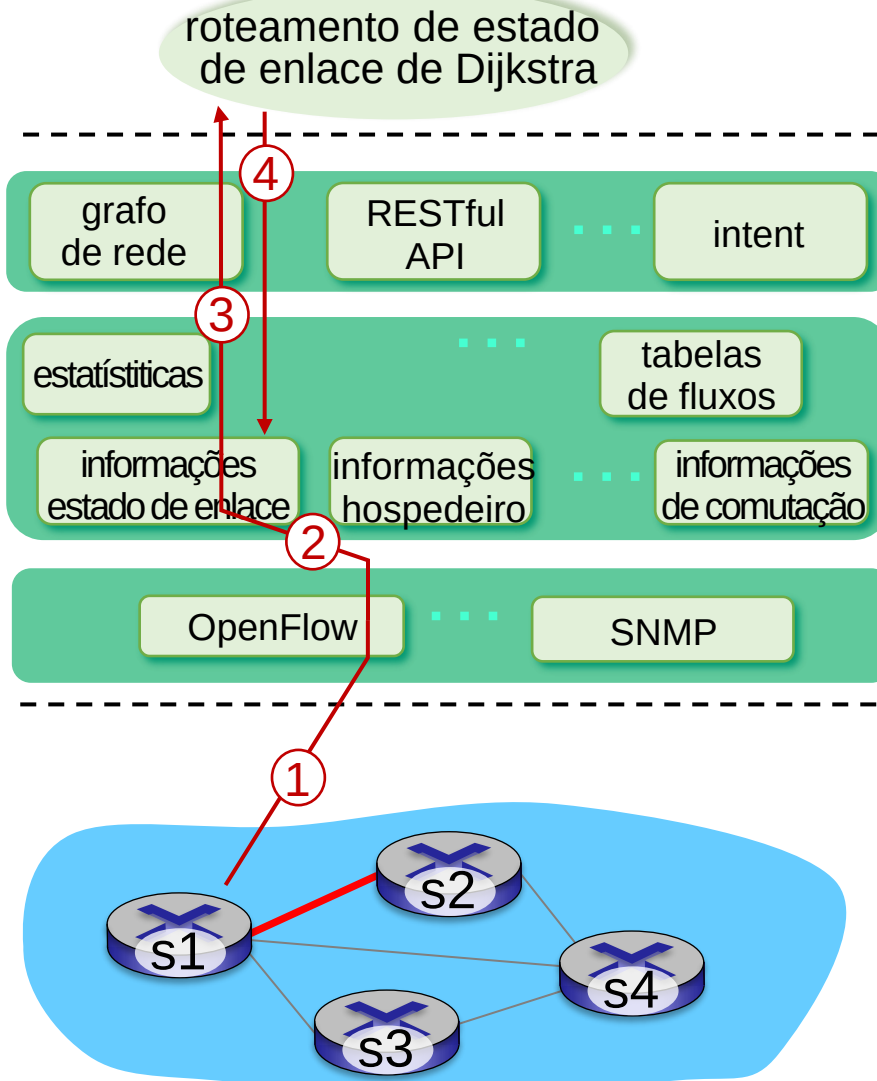
- *packet-in*: transfere pacote (e seu controle) para o controlador. Veja a mensagem *packet-out* do controlador
- *flow-removed*: entrada da tabela de fluxo excluída no comutador
- *port status*: informa o controlador de uma mudança em uma porta.

Felizmente, os operadores de rede não “programam” comutadores criando/enviando mensagens OpenFlow diretamente. Em vez disso, usam abstrações de nível superior no controlador

Controlador OpenFlow

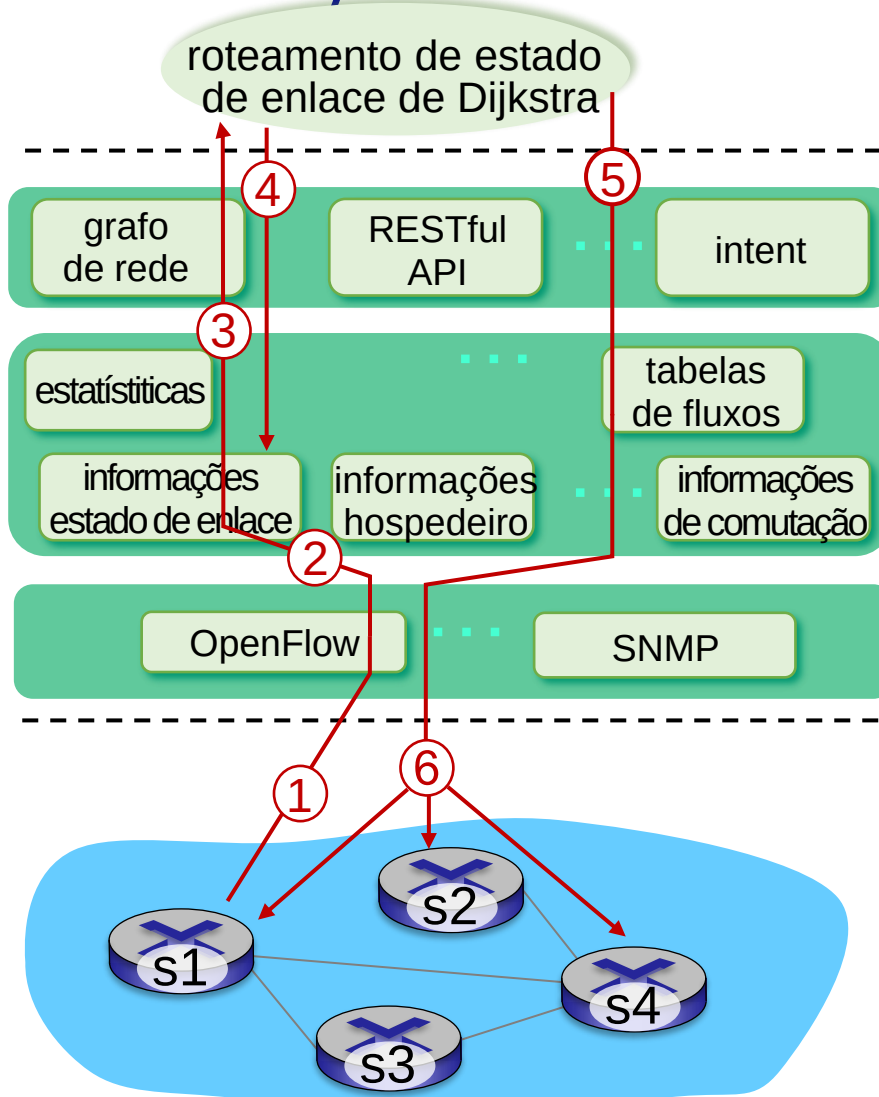


SDN: exemplo de interação plano de controle/dados



- ① S1, com falha de enlace, usa a mensagem port status do OpenFlow para notificar o controlador
- ② O controlador SDN recebe a mensagem OpenFlow e atualiza as informações de status do enlace
- ③ O aplicativo de algoritmo de roteamento de Dijkstra se registrou anteriormente para ser chamado sempre que o status do enlace mudar. Ele é chamado.
- ④ O algoritmo de roteamento de Dijkstra acessa informações do grafo da rede, informações do estado do enlace no controlador, e calcula novas rotas

SDN: exemplo de interação plano de controle/dados

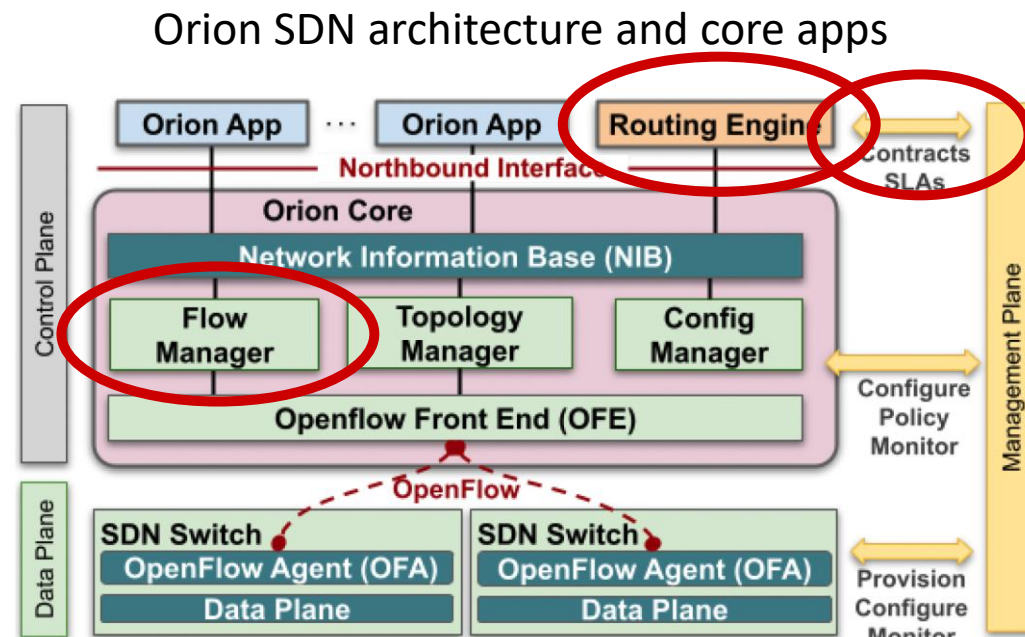


- ⑤ o aplicativo de roteamento de estado de enlace interage com o componente de computação de tabela de fluxo no controlador SDN, que calcula as novas tabelas de fluxo necessárias
- ⑥ controlador usa OpenFlow para instalar as novas tabelas em comutadores que precisam de atualização

Plano de controle SDN da Google ORION

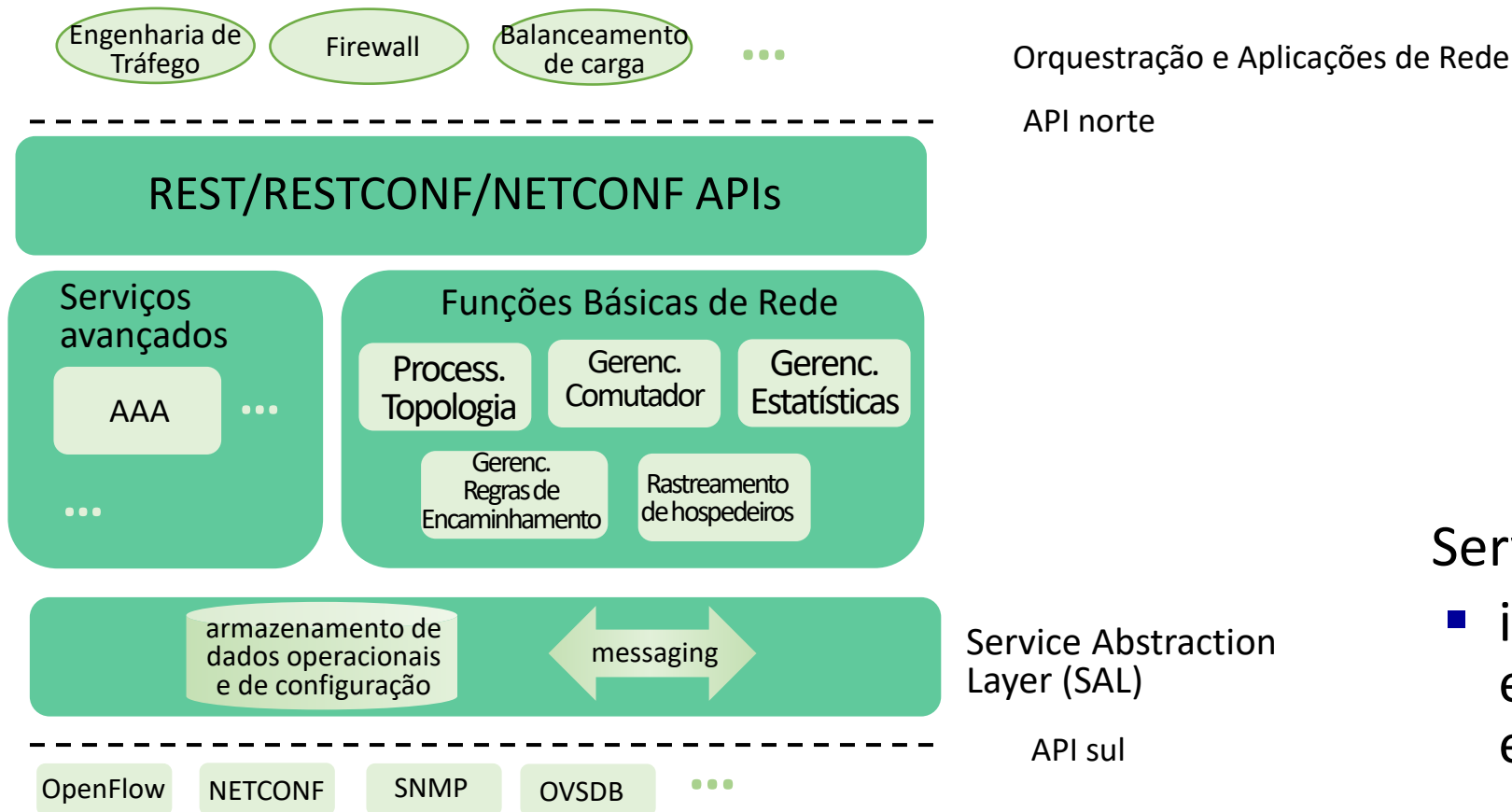
ORION: Plano de controle SDN da Google (*NSDI'21*): plano de controle para o datacenter da Google (Jupiter) e WANs (B4)

- **roteamento** (intradomínio, iBGP) e engenharia de tráfego: implementados em *aplicações* no topo do núcleo ORION
- controles **baseados em fluxo de ponta a ponta** (por exemplo, agendamento CoFlow) para atender aos SLAs do contrato
- **gerenciamento**: microsserviços distribuídos no núcleo Orion e OpenFlow para sinalização/monitoramento



Nota: ORION fornece serviços *intradomínio* dentro da rede da Google

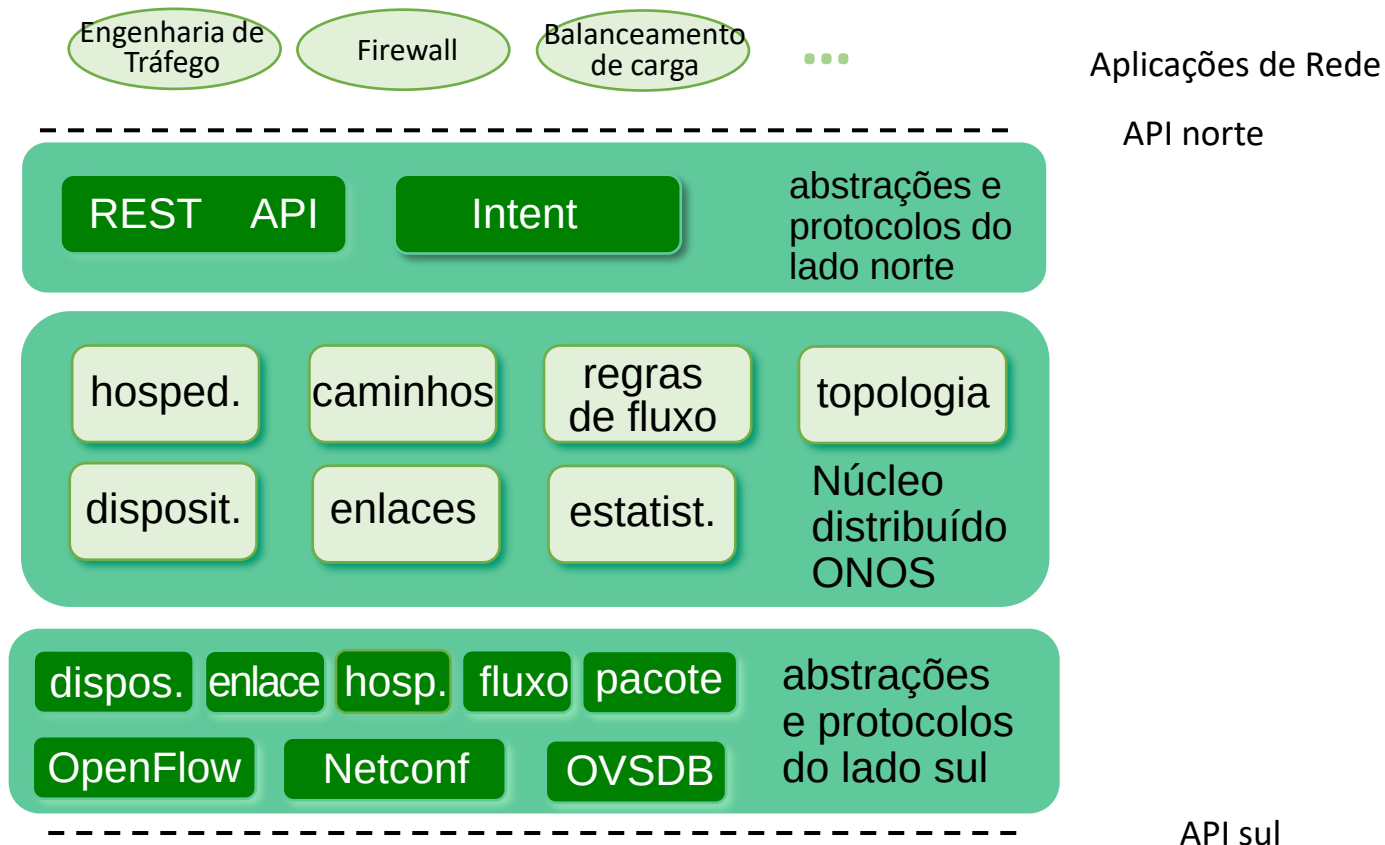
Controlador OpenDaylight (ODL)



Service Abstraction Layer:

- interconecta aplicativos e serviços internos e externos

Controlador ONOS



- aplicativos de controle separados do controlador
- intent framework: especificação de serviço de alto nível: o que em vez de como
- ênfase considerável no núcleo distribuído: confiabilidade de serviço, escalabilidade com desempenho

SDN: desafios selecionados

- fortalecendo o plano de controle: sistema distribuído confiável, escalável com desempenho e seguro
 - robustez a falhas: aproveita a forte teoria de sistema distribuído confiável para o plano de controle
 - confiabilidade e segurança: “incorporados” desde o primeiro dia?
- redes e protocolos que atendem aos requisitos específicos da missão
 - por exemplo, em tempo real, ultra-confiável, ultra-seguro
- escala na Internet: além de um único AS
- SDN é crítica em redes celulares 5G

SDN e o futuro dos protocolos de rede tradicionais

- Tabelas de encaminhamento calculadas por SDN versus calculadas por roteador:
 - apenas um exemplo de cálculo lógico-centralizado versus cálculo de protocolo
- pode-se imaginar o controle de congestionamento computado por SDN:
 - o controlador define as taxas do remetente com base nos níveis de congestionamento relatados 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 das SDN
- **Internet Control Message Protocol**



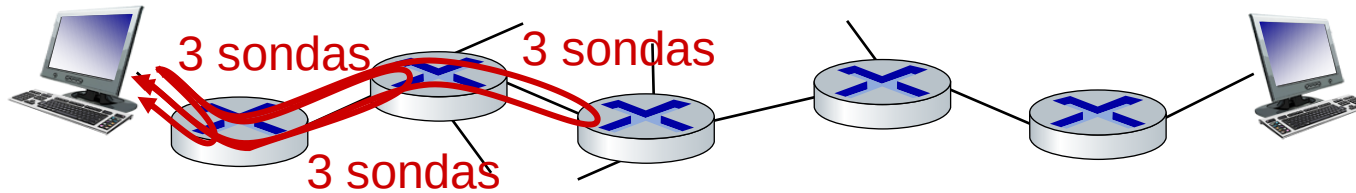
- gerenciamento e configuração de rede
 - SNMP
 - NETCONF/YANG

ICMP: internet control message protocol

- usado por hospedeiros e roteadores para comunicar informações em nível de rede
 - relatório de erros: hospedeiro, rede, porta, ou protocolo inacessível
 - solicitação/resposta de eco (usado por ping)
- camada de rede “acima” do IP:
 - mensagens ICMP são transportadas em datagramas IP
- *mensagem ICMP*: tipo, código mais os primeiros 8 bytes do datagrama IP causando erro

<u>Tipo</u>	<u>Código</u>	<u>Descrição</u>
0	0	resposta de eco (ping)
3	0	rede de destino inalcançável
3	1	hospedeiro de destino inalcançável
3	2	protocolo de destino inalcançável
3	3	porta de destino inalcançável
3	6	rede de destino desconhecida
3	7	hospedeiro de destino desconhecido
4	0	supressão de fonte (controle de congestionamento – não usado)
8	0	requisição de eco (ping)
9	0	anúncio de rota
10	0	descoberta de rota
11	0	TTL expirado
12	0	cabeçalho IP inválido

Traceroute e ICMP



- origem envia conjuntos de segmentos UDP para destino
 - 1º conjunto tem TTL =1, 2º conjunto tem TTL=2, etc.
- datagrama no n -ésimo conjunto chega ao n -ésimo roteador:
 - roteador descarta datagrama e envia mensagem ICMP para a 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: grava RTTs

critérios de parada :

- o segmento UDP eventualmente chega ao hospedeiro de destino
- destino retorna mensagem ICMP “porta inacessível” (tipo 3, código 3)
- origem para

Camada de rede: roteiro do “plano de controle”

- introdução
- protocolos de roteamento
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- plano de controle das SDN
- Internet Control Message Protocol



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

O que é gerenciamento de rede?

- sistemas autônomos (também conhecidos como “redes”): milhares de componentes de hardware/software interagindo
- outros sistemas complexos que requerem monitoramento, configuração, controle :
 - avião a jato, usina nuclear, outros?

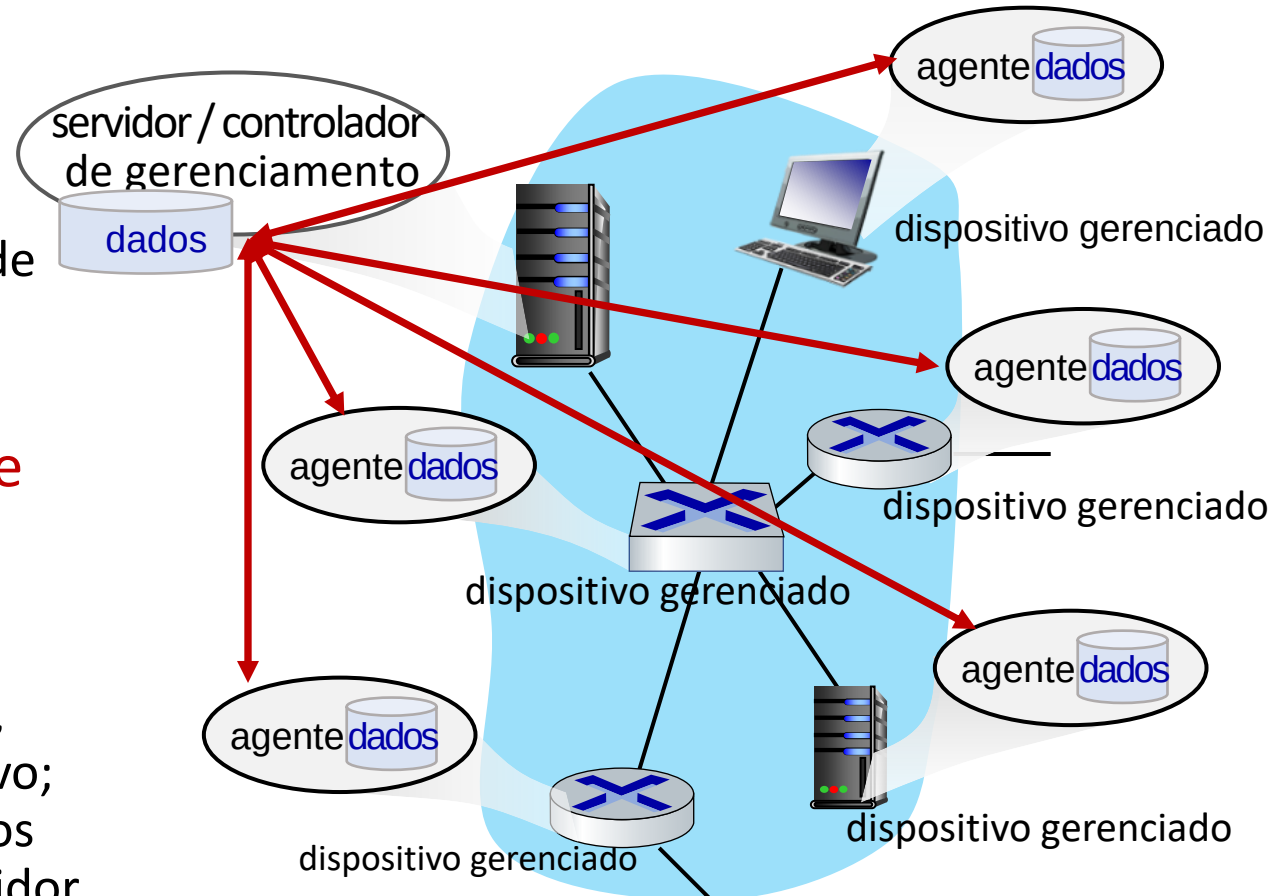


"Gerenciamento de rede inclui a implantação, integração e coordenação do hardware, software e elementos humanos para monitorar, testar, sondar, configurar, analisar, avaliar e controlar a rede e os recursos dos elementos para atender aos requisitos de tempo real, desempenho operacional, e Qualidade de Serviço a um custo razoável."

Componentes de gerenciamento de rede

Servidor de gerenciamento :
aplicação,
normalmente com
gerenciadores de rede
(humanos) no ciclo

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



Dispositivo gerenciado :
equipamento com hardware
gerenciável e configurável,
componentes de software

Dados: dados de
configuração do
“estado” do
dispositivo, dados
operacionais,
estatísticas do
dispositivo

Abordagens do operador de rede para gerenciamento

CLI (Command Line Interface)

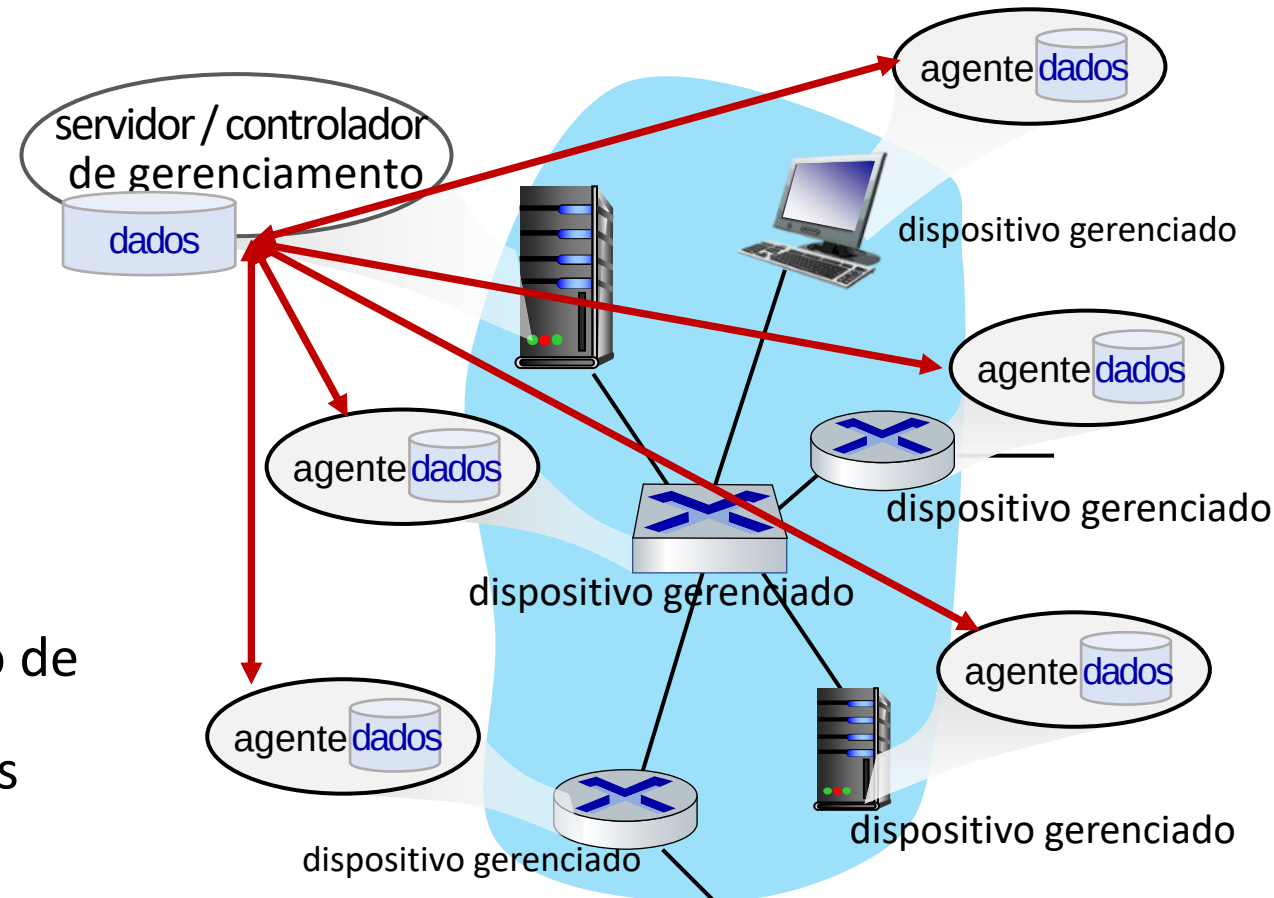
- operador envia (digitação, scripts) diretamente para dispositivos individuais (ex.: vis ssh)

SNMP/MIB

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

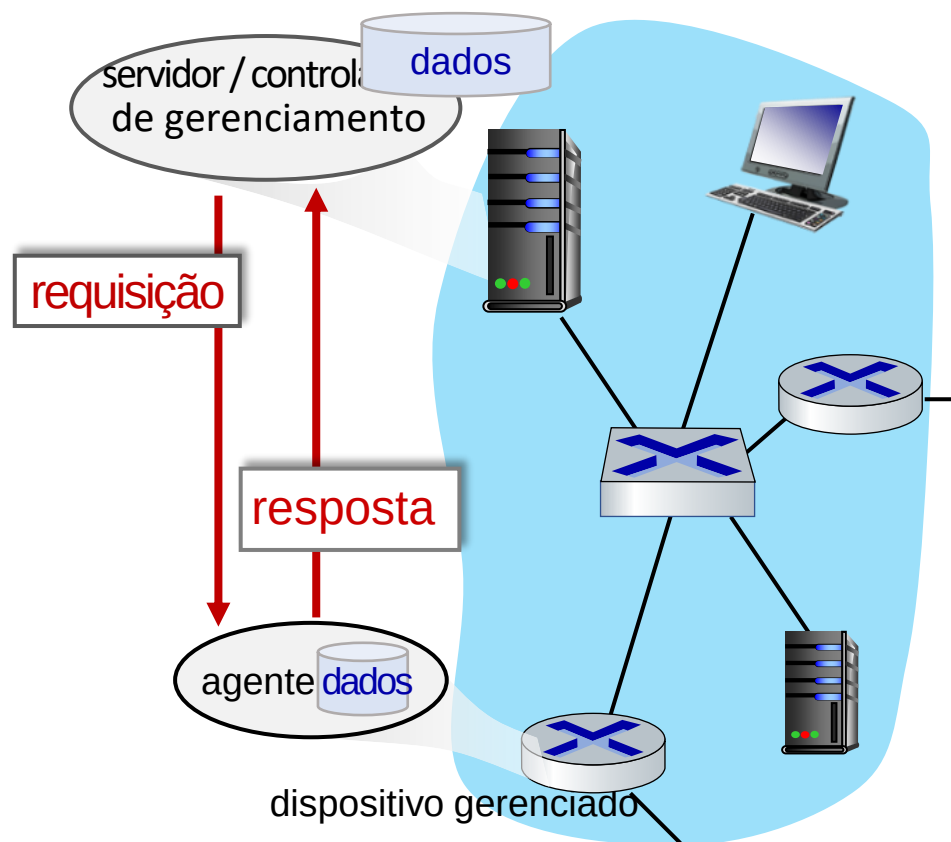
NETCONF/YANG

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

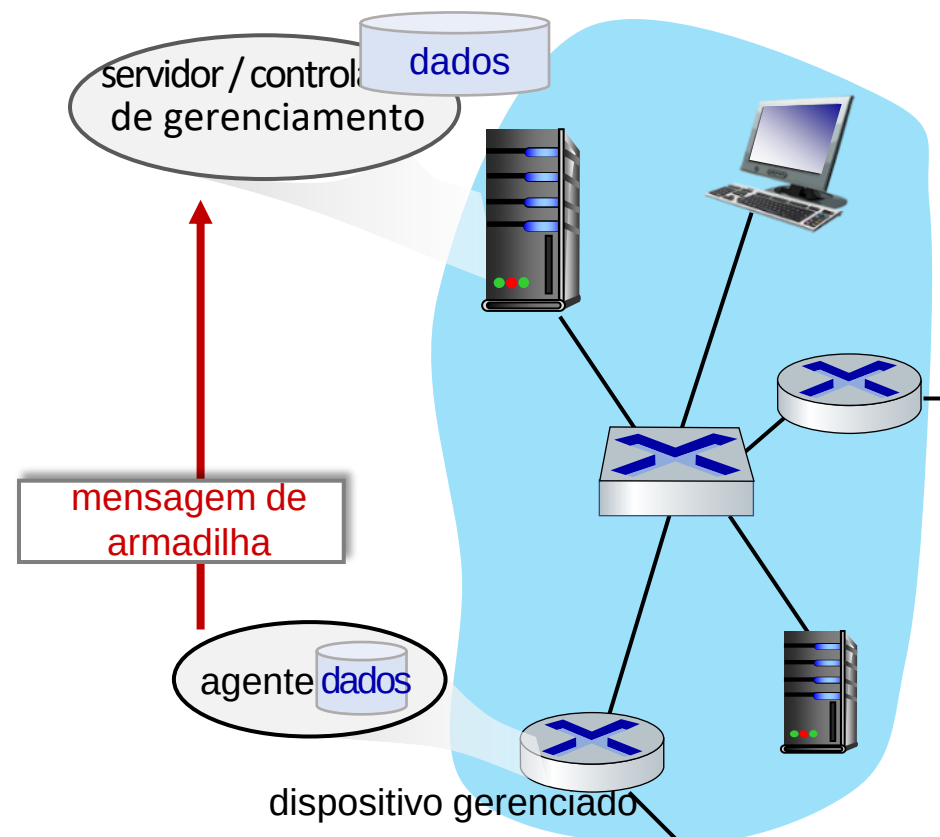


Protocolo SNMP

Duas maneiras de transmitir informações e comandos MIB:



modo requisição/resposta (request/response)

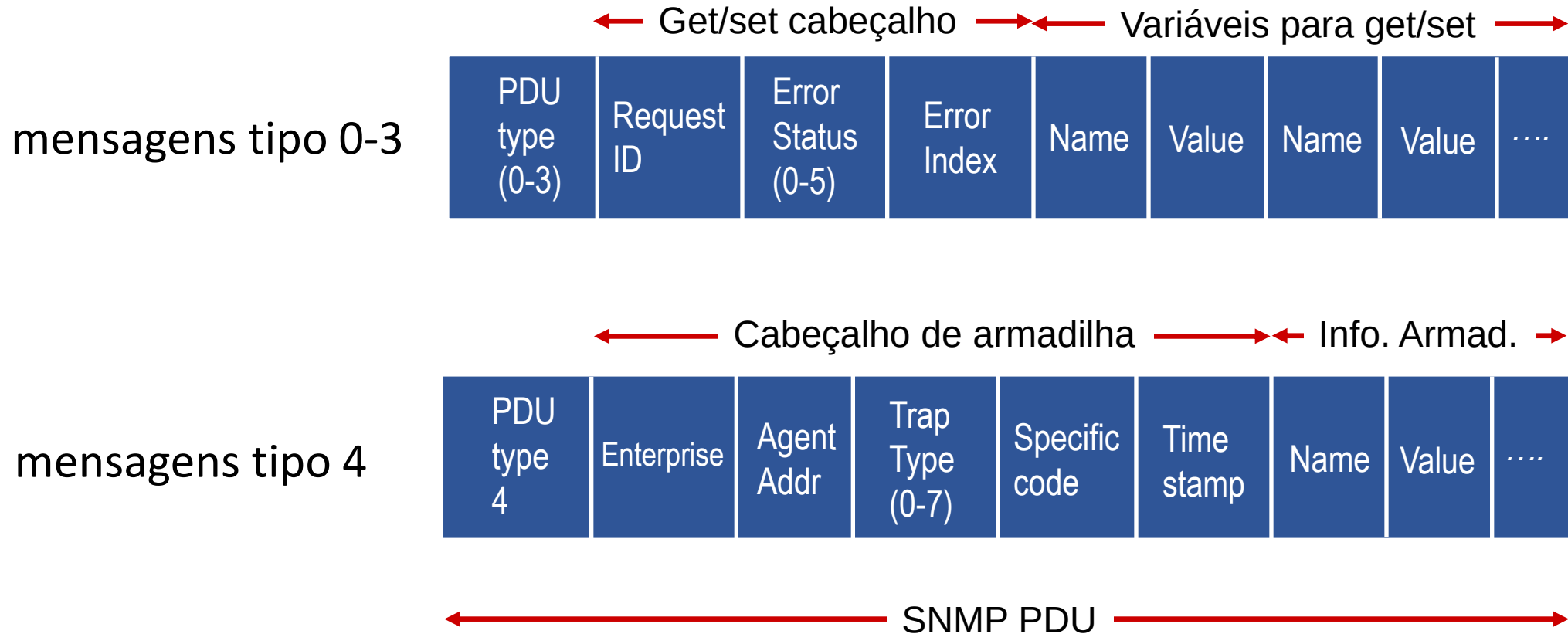


modo armadilha (trap)

Protocolo SNMP: tipos de mensagens

Tipo de mensagem		Função
Requisições	GetRequest GetNextRequest GetBulkRequest	gerenciador-para-agente: “me dê dados” (instância de dados, próximo dado na lista, bloco de dados).
	SetRequest	gerenciador-para-agente: ajuste valores MIB
Resposta	Response	Agente-para-gerenciador: valor, resposta para Requisição
Armadilha	Trap	Agente-para-gerenciador: informa gerenciador sobre um evento excepcional

Protocolo SNMP: formato de mensagens



SNMP: Management Information Base (MIB)

- dados operacionais (e algumas configurações) do dispositivo gerenciado
- reunidos no **modulo MIB** do dispositivo
 - 400 módulos MIB definidos em RFC's; muitos mais MIBs específicos de fornecedores
- **Structure of Management Information (SMI)**: linguagem de definição de dados
- exemplo de variáveis MIB para o protocolo UDP :

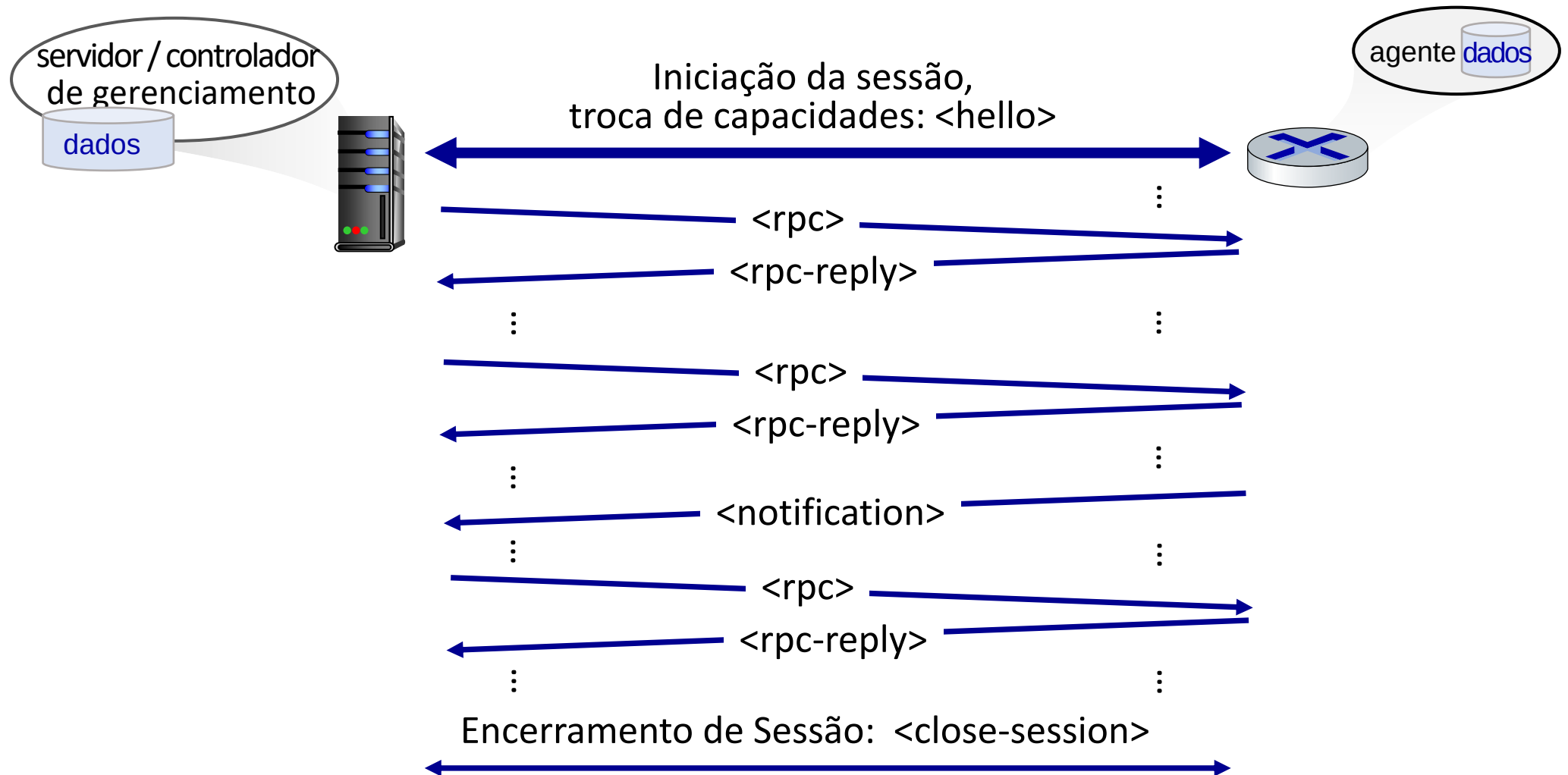


ID de Objeto	Nome	Tipo	Comentários
1.3.6.1.2.1.7.1	UDPInDatagrams	contador 32 bits	número total de datagramas entregues
1.3.6.1.2.1.7.2	UDPNoPorts	contador 32 bits	número de datagramas não entregues (sem aplicação na porta)
1.3.6.1.2.1.7.3	UDInErrors	contador 32 bits	número de datagramas não entregues (todos os outros motivos)
1.3.6.1.2.1.7.4	UDPOutDatagrams	contador 32 bits	número total de datagramas enviados
1.3.6.1.2.1.7.5	udpTable	SEQUÊNCIA	uma entrada para cada porta atualmente em uso

Visão geral do NETCONF

- **objetivo:** gerenciar/**configurar** ativamente 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 (**atomic-commit**) em múltiplos dispositivos
 - consultar dados operacionais e estatísticas
 - subscrever notificações de dispositivos
- paradigma de chamada de procedimento remoto (remote procedure call - RPC)
 - mensagens do protocolo NETCONF codificadas em XML
 - trocados por protocolo de transporte seguro e confiável (por exemplo, TLS)

NETCONF inicialização, troca, encerramento



Operações NETCONF selecionadas

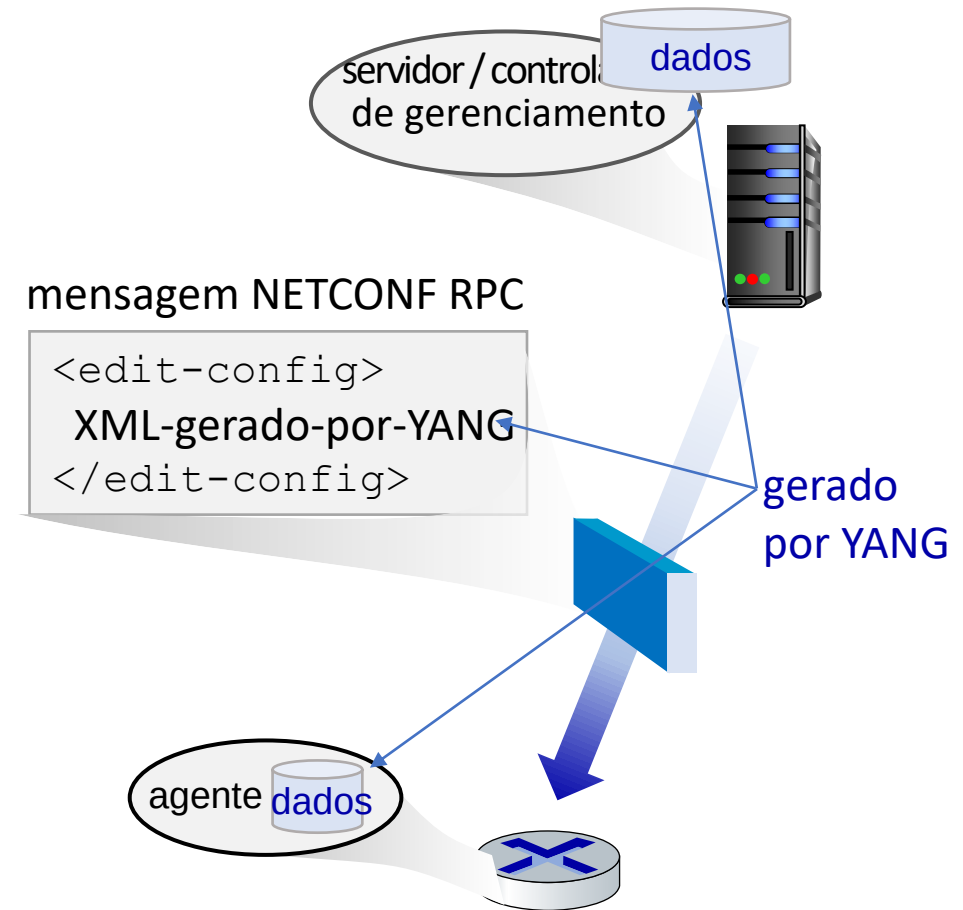
NETCONF	Descrição da Operação
<get-config>	Recupera toda ou parte de uma determinada configuração. Um dispositivo pode ter várias configurações.
<get>	Recupera todo ou parte do estado de configuração e dos dados do estado operacional.
<edit-config>	Altera a configuração especificada (possivelmente em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

Exemplo de mensagem NETCONF RPC

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <rpc message-id="101"  note message id
03   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
04   <edit-config>      altera uma configuração
05     <target>
06       <running/> altera 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> muda 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 estrutura, sintaxe, semântica de dados de gerenciamento de rede NETCONF
 - tipos de dados integrados, como SMI
- documento XML descrevendo dispositivo e capacidades pode ser gerado a partir da descrição YANG
- pode expressar restrições entre dados que devem ser satisfeitos por uma configuração NETCONF válida
 - garante que as configurações NETCONF satisfaçam as restrições de correção e consistência



Camada de rede: resumo

aprendemos muito!

- abordagens ao plano de controle de 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
- Internet Control Message Protocol
- gerenciamento de rede

próxima parada: camada de enlace!

Camada de rede, plano de controle: **Concluído!**

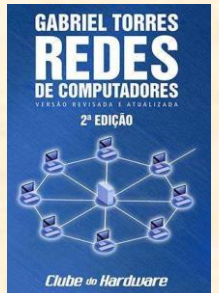
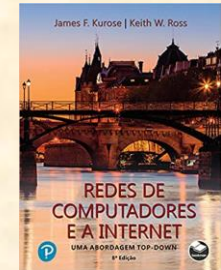
- introdução
- protocolos de roteamento
 - estado de enlace
 - vetor de distância
- roteamento intra-ISP: OSPF
- roteamento entre ISPs: BGP
- plano de controle das SDN
- Internet Control Message Protocol



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

Leitura Recomendada e Complementar

- Leitura Recomendada
 - [KUROSE, James F. e ROSS, Keith W. Redes de computadores e a Internet: Uma abordagem top-down. 8ª Edição. Bookman, 2021.](#)
 - Capítulo 5 – Camada de Rede: O Plano de Controle
 - [TANENBAUM, Andrew S., FEAMSTER, Nick e WETHERALL, David. Redes de Computadores. 5ª Edição. São Paulo: Bookman, 2021.](#)
 - Capítulo 5 – A Camada de Rede.
- Leitura Complementar
 - [FOUROUZAN, Behrouz A. e FIROUZ, Mosharraf. Redes de Computadores: uma abordagem top-down. Porto Alegre: AMGH, 2013.](#)
 - Capítulo 4 – Camada de Rede.
 - [TORRES, Gabriel. Redes de Computadores: Curso Completo. Axcel Books, 2001.](#)
 - Capítulo 3 – TCP/IP e Capítulo 18 – Roteadores.
 - [COMER, Douglas E. Interligação de Redes com TCP/IP. Volume 1: Princípios, protocolos e arquitetura. 6ª Edição. Rio de Janeiro: Elsevier, 2015.](#)
 - Capítulos 6 a 9



Slides adicionais do Capítulo 5

Vetor de distância: outro exemplo

$$D_x()$$

	custo para		
	X	y	z
de	0	2	7
y	∞	∞	∞
z	∞	∞	∞

$$D_y()$$

	custo para		
	X	y	z
de	∞	∞	∞
y	2	0	1
z	∞	∞	∞

$$D_z()$$

	custo para		
	X	y	z
de	∞	∞	∞
y	∞	∞	∞
z	7	1	0

$$D_x()$$

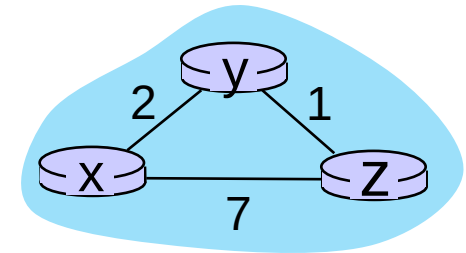
	custo para		
	X	y	z
de	0	2	3
y	2	0	1
z	7	1	0

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

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

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

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



..... tempo

Vetor de distância: outro exemplo

