Protocolos de roteamento

- O roteamento é a principal função da camada de rede
  - Pacotes passam por múltiplos nós intermediários até chegar ao destino
  - Objetivo de um *protocolo de roteamento*:
    Achar o "melhor" dentre vários caminhos alternativos (de acordo com algum critério)
    - ✓ Caminho: sequência de roteadores que os pacotes atravessarão da origem até o destino
    - ✓ "Melhor": menor custo, menos congestionado, mais rápido

Algoritmo de Roteamento

Parte do software da camada de rede responsável por decidir o caminho fim-a-fim a ser seguido pelos pacotes

Obs: Pacotes de uma mesma conexão podem seguir caminhos diferentes

Métricas para se calcular o custo de um caminho

- Número de saltos (*hops*) ou enlaces intermediários
- Distância geográfica
- Tempo médio de permanência na fila e de transmissão nos roteadores (tempo de serviço)
- Largura de banda dos enlaces
- Quantidade de tráfego nos enlaces
- Custo monetário da comunicação
- Tamanho médio da fila

Ou uma função que combina várias destas métricas

Roteamento por caminho "mais curto"

- Constrói-se um grafo da sub-rede
  - nós representam roteadores
  - arestas representam enlaces de comunicação

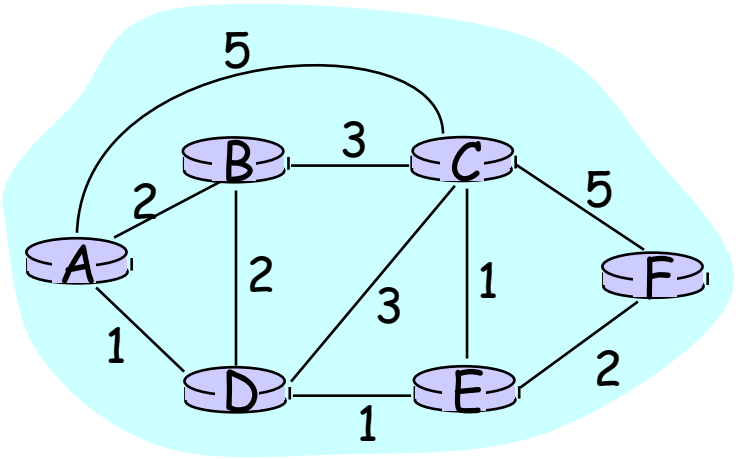
Melhor rota entre um dado par de roteadores:

Rota "mais curta" (de menor custo) entre os dois nós do grafo correspondentes ao par de roteadores

Algoritmos de roteamento

Algoritmo de Roteamento

Objetivo: determinar "bons" caminhos (caminhos de menor custo) através da rede, da fonte ao destino.



Algoritmos de roteamento são descritos por grafos:

- Nós do grafo representam roteadores
- Arestas do grafo representam enlaces
  - ✓ Custo do enlace: atraso, preço, nível de congestionamento, inverso da largura de banda, etc

Custo do caminho (a, d, e) = c(a,d) + c(d,e)

- "bons" caminhos:
  - ✓ tipicamente correspondem aos caminhos de menor custo
  - ✓ Podem existir caminhos redundantes ou alternativos (c/ mesmo custo)

Classificação dos Algoritmos de Roteamento

Informação global ou descentralizada

Global:

- Todos os roteadores têm informações completas da topologia da rede e dos custos dos enlaces
- algoritmos "*Link State*" (estado de enlace)

Descentralizada:

- Roteadores só conhecem informações sobre seus vizinhos e os enlaces para eles
- Processo de computação iterativo: troca de informações com os vizinhos
- algoritmos "*Distance Vector*" (vetor de distâncias)

Classificação dos Algoritmos de Roteamento

Estático ou Dinâmico

Estático:

- Caminhos são computados previamente (off-line) e então descarregados nos roteadores
- As rotas mudam lentamente ao longo do tempo

Classificação dos Algoritmos de Roteamento

Estático ou Dinâmico

Dinâmico:

- Rotas são computadas dinamicamente
- Cálculo de rotas é feito periodicamente, baseado nas condições atuais do tráfego e da topologia da rede
- Reflete mudanças na topologia e nas características de tráfego
- As rotas mudam mais rapidamente
  - ✓ Atualizações periódicas e automáticas
  - ✓ Podem responder mais rapidamente a mudanças no custo dos enlaces

Classificação dos Algoritmos de Roteamento

- Estáticos:
  - Rota mais "curta"
  - Roteamento baseado na carga do enlace
- Dinâmicos:
  - Baseado no vetor de distâncias

Algoritmo de estado de enlace (LS)

Algoritmo de Dijkstra

- **Centralizado:** topologia de rede e custo dos enlaces são conhecidos por todos os nós.
  - ✓ Implementado via "*link state broadcast*"
  - ✓ Todos os nós têm a mesma informação
- Computa caminhos de menor custo de um nó (fonte) para todos os outros nós
  - ✓ Permite obter uma tabela de roteamento para aquele nó
- Convergência: após *k* iterações, conhece-se o caminho de menor custo para *k* destinos.

Notação:

C(i,j): custo do enlace do nó i ao nó j. Custo é infinito se não houver enlace entre i e j

D(v): valor atual do custo do caminho da fonte ao destino **v**

P(v): nó predecessor ao longo do caminho da fonte ao nó **v**, isto é, antes do nó **v**

N: subconjunto de nós para os quais o caminho de menor custo é definitivamente conhecido

Algoritmo de Dijsktra

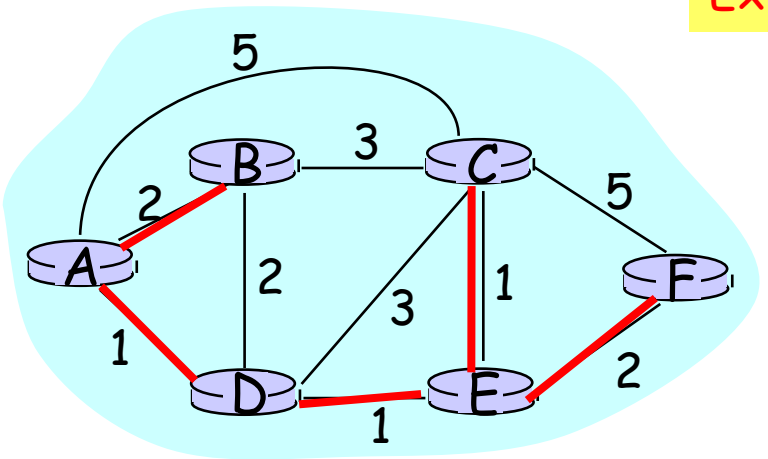
Executando no nó *A*

1 ***Inicialização:***
2 N = {A}
3 para todos os nós v
4 se v é adjacente a A
5 então D(v) = c(A,v)
6 senão D(v) = infinito
7
8 ***Loop***
9 ache w não em N tal que D(w) é mínimo (entre os demais)
10 acrescente w a N
11 atualize D(v) para todo v adjacente a w e não em N:
12 D(v) = min(D(v), D(w) + c(w,v))
13 /* novo custo para v é o custo anterior para v ou o menor
14 custo de caminho conhecido para w mais o custo de w a v */
15 ***até que todos os nós estejam em N***

Algoritmo de Dijkstra: exemplo

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

Executando no nó *A*



Algoritmo de Dijkstra: exemplo (2)

Menor caminho resultante a partir de A:

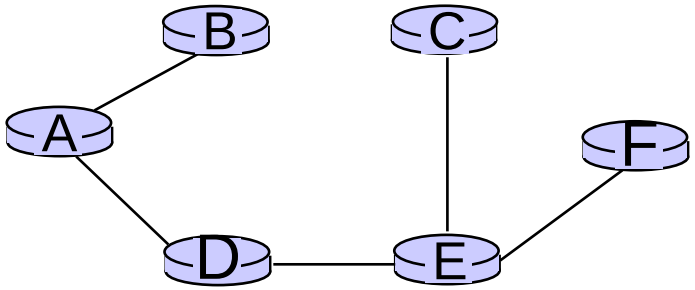


Tabela de repasse resultante em A:

destino	enlace
B	(A,B)
D	(A,D)
E	(A,D)
C	(A,D)
F	(A,D)

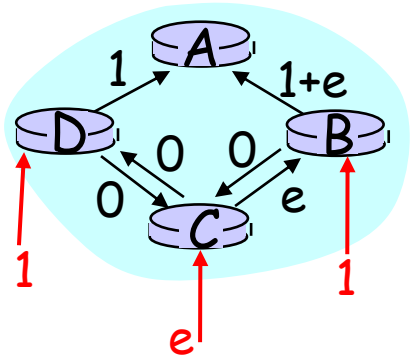
Discussão do Algoritmo de Dijkstra

Complexidade do Algoritmo para n nós:

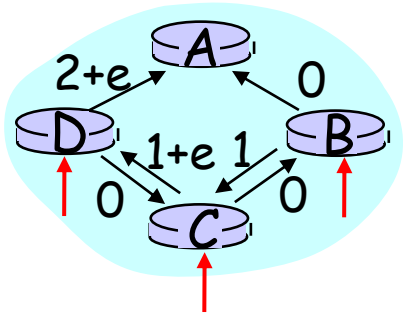
- Cada iteração: precisa verificar todos os nós w, que não estão em N
- [(n)+(n-1)+(n-2)+,…,+1] = n*(n+1)/2 comparações: complexidade $o(n^2)$
- Implementações mais eficientes: o(n*log n)

Oscilações possíveis:

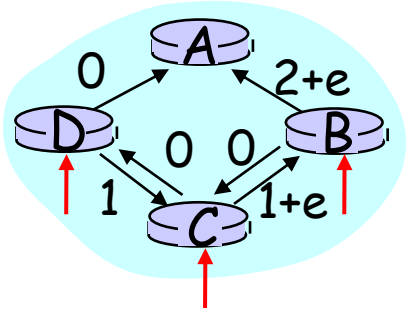
- Por exemplo, custo do enlace é igual ao total de tráfego transportado no mesmo, onde c(u,v) pode ser diferente de c(v,u)
- Nó D origina uma unidade de tráfego destinada a A. Idem para o nó B. Nó C injeta "e", destinada a A.



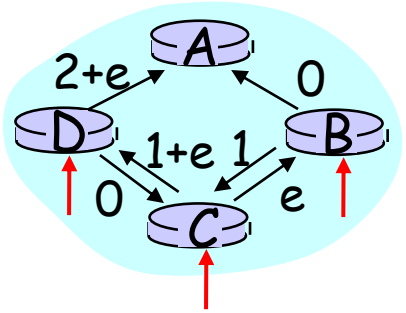
Roteamento inicial



… recalcula roteamento



… recalcula



… recalcula

Algoritmo Distance Vector

Iterativo:

- Continua até que os nós não troquem mais informações.
- *Finito: n*ão há sinal de parada

Assíncrono:

- Os nós não precisam trocar informações simultaneamente!

Distribuído:

- Cada nó se comunica apenas com os seus vizinhos diretamente conectados

Estrutura de dados da tabela de distância

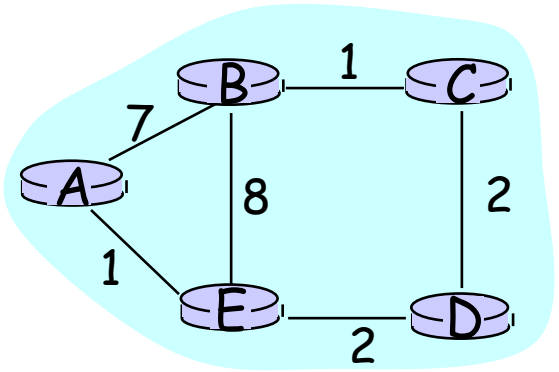
- Cada nó tem sua própria tabela
- Linha para cada possível destino
- Coluna para cada roteador vizinho

Exemplo: no nó X, para destino Y via vizinho Z:

$$D^{X}(Y,Z) = \text{distância } \textit{de } X \textit{ para } Y, \textit{via } Z \text{ como próx. salto}$$

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

Exemplo de Tabela de Distância



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

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

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

custo via nó vizinho

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

A Tabela de Distâncias Gera a Tabela de Roteamento

custo através de

$D^E()$

destino	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

Enlace de saída, custo

destino	
A	A,1
B	D,5
C	D,4
D	D,2

Tabela de distância → Tabela de Roteamento (para o nó E)

Algoritmo Distance Vector

Ideia chave:

- De tempos em tempos, cada nó (roteador) envia sua própria tabela (DV) para os vizinhos

- Quando x recebe um novo DV de qualquer vizinho, x atualiza sua tabela (DV) usando a equação de Bellman-Ford:

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

  - Sob condições normais $D_x(y)$ *converge para o menor custo* $d_x(y)$

Roteamento Vetor-Distância: resumo

Iterativo, assíncrono:

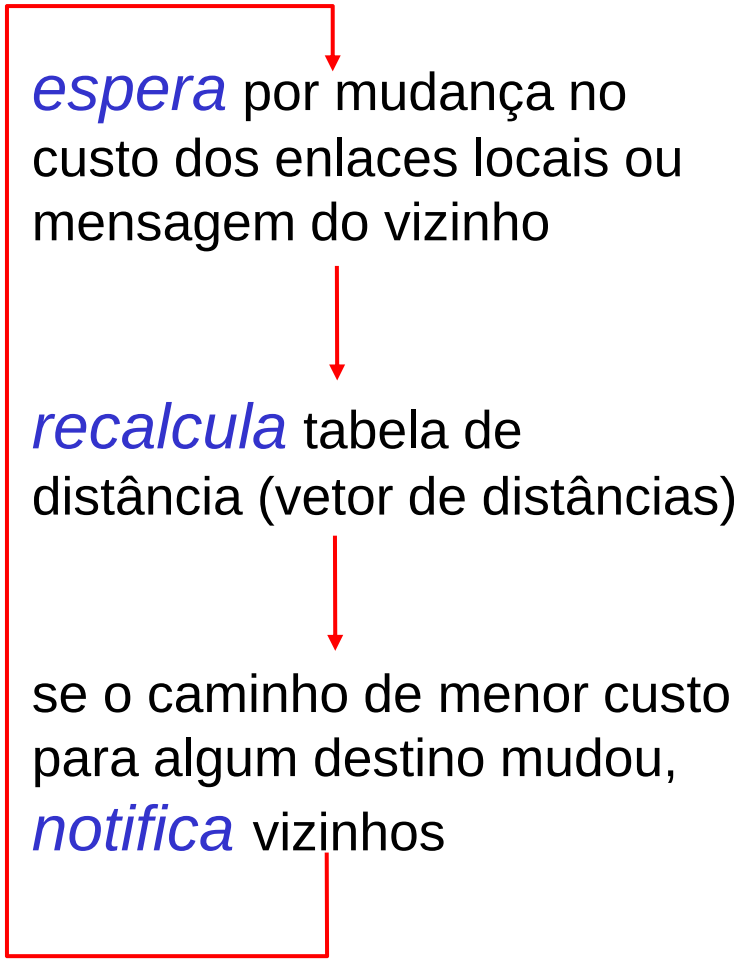
cada iteração local é causada por:

- Mudança de custo dos enlaces locais
- Mensagem do vizinho: seu caminho de menor custo para o algum destino mudou

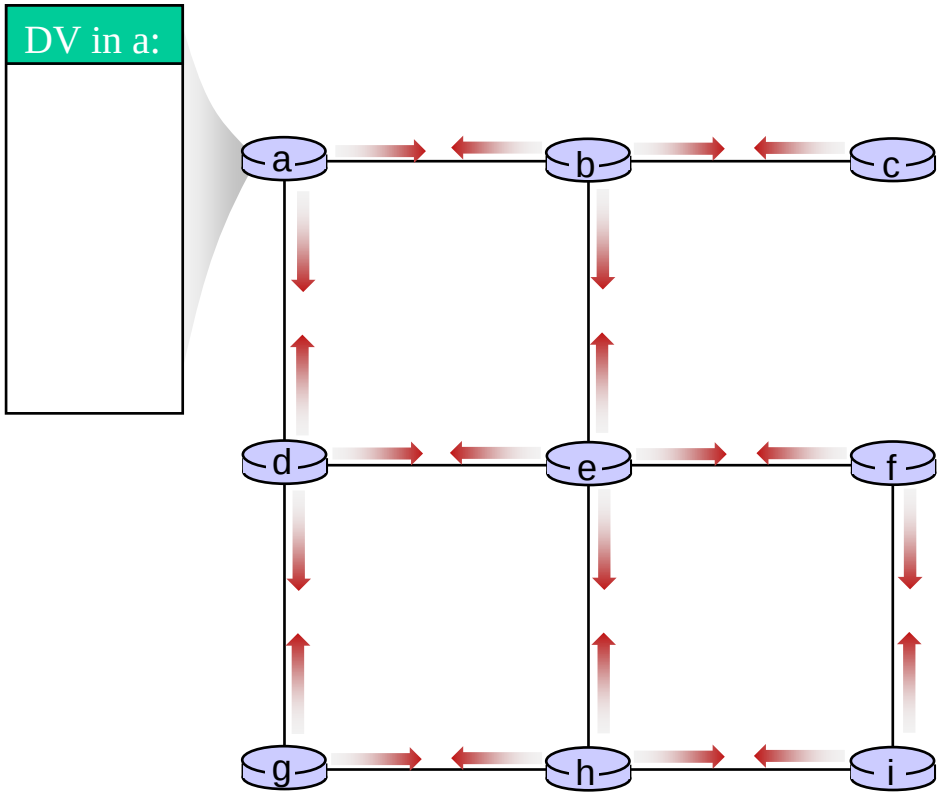
Distribuído:

- Cada nó notifica seus vizinhos apenas quando seu menor custo para algum destino muda (DV)
  - ✓ Vizinhos notificam seus vizinhos, e assim por diante…

Cada nó:



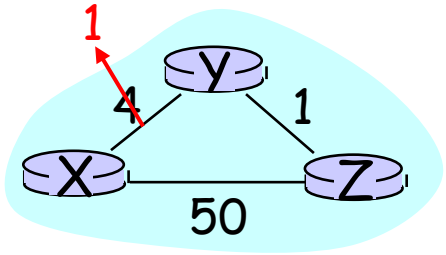
Exemplo do Distance Vector



Vetor-Distância: mudança no custo do enlace

Mudança no custo do enlace (para menos):

- nó detecta que o custo do enlace local mudou
- atualiza tabela de distâncias
- se o custo do caminho de menor custo mudou, notifica vizinhos



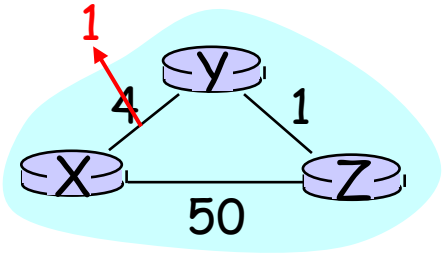
"boas notícias viajam depressa"

t_0 : y detecta mudança no custo do enlace, atualiza seu DV, informa seus vizinhos.

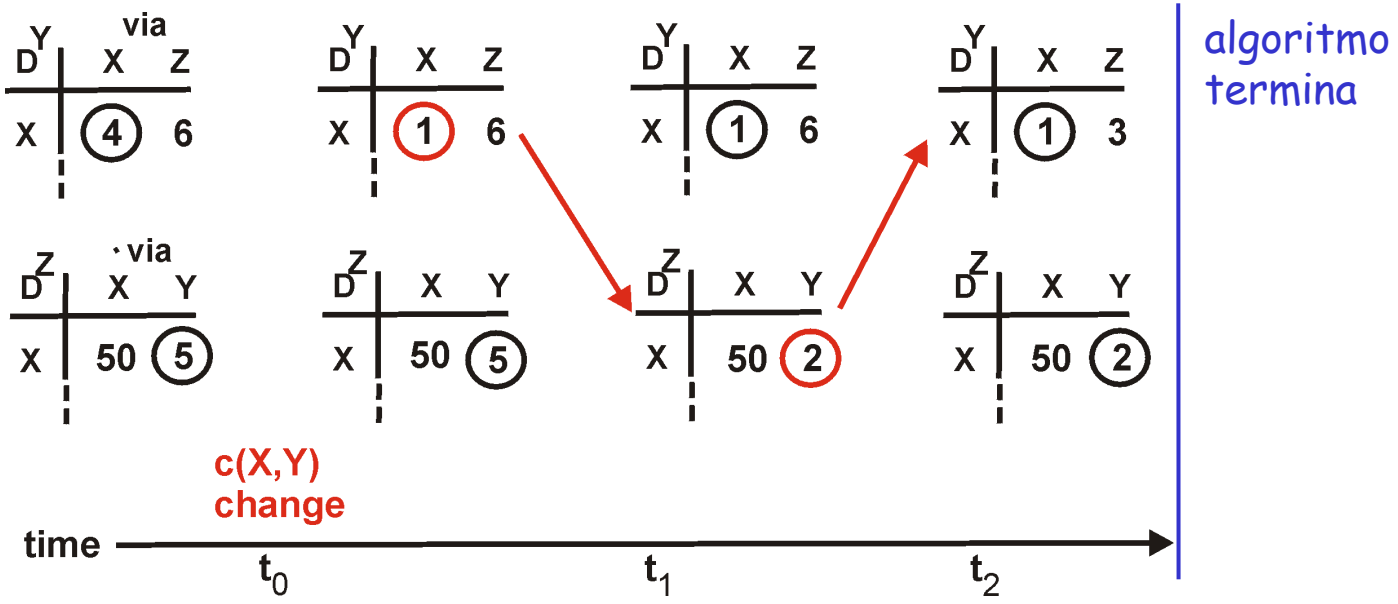
t_1 : z recebe a atualização de y, atualiza sua tabela, calcula o novo menor custo para x, envia aos vizinhos seu DV.

t_2 : y recebe a atualização de z, atualiza sua tabela de distâncias. Os menores custos de y não mudam, portanto y não envia nova mensagem aos vizinhos.

Vetor-Distância: mudança no custo do enlace

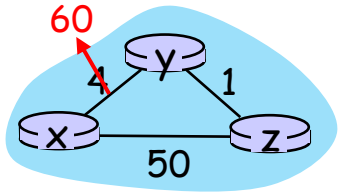


"boas notícias viajam depressa"



Vetor-Distância: mudança no custo do enlace

Mudança de custo (para mais!):



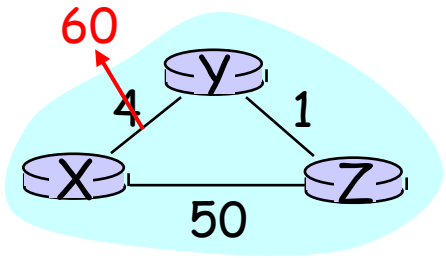
- nó detecta que o custo do enlace local mudou
- "notícias ruins viajam lentamente" – ***problema da contagem ao infinito:***

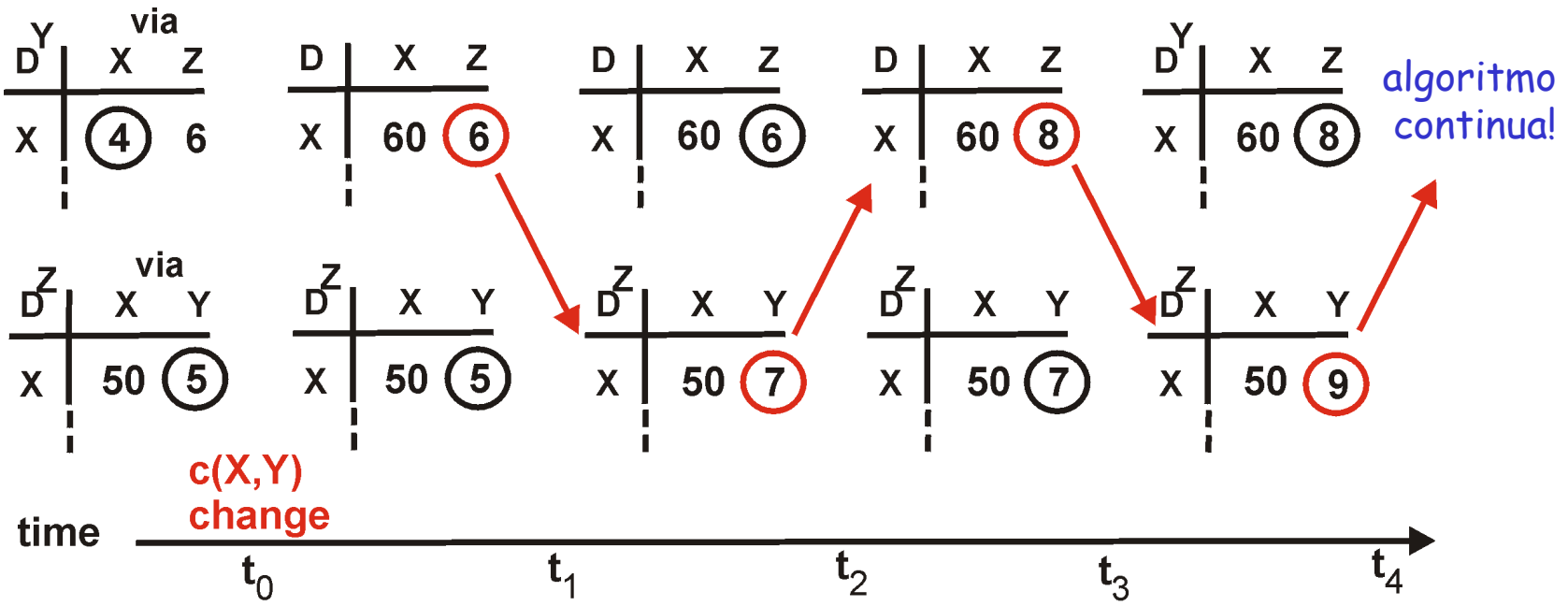
- *y vê que o custo do enlace para x* mudou p/ 60, mas z disse a ele que seu custo para x era 5. Logo *y* recalcula o menor custo p/ x como 6, via z; notifica z do novo custo (6) p/ *x*.
- z *aprende* que caminho p/ x via *y* tem um novo custo (6), logo z recalcula o "novo custo *x* será 7 via y e notifica *y* do seu novo custo (7) p/ *x*.
- *y aprende* que caminho p/ x via z tem um novo custo 7, logo *y* recalcula o "novo custo *x* será 8, via z, notifica z do seu novo custo (8) p/ *x*.
- z *aprende* que caminho p/ x via *y* tem um novo custo 8, so z recalcula o "novo custo *x* será 9 via y, notifica *y* do seu novo custo (9) p/ *x*....

Vetor-Distância: mudança no custo do enlace

Mudança no custo do enlace (para mais):

- más notícias viajam devagar - problema da "contagem ao infinito"
  - ✓ a rede demora para aprender os novos custos (44 iterações neste exemplo!)

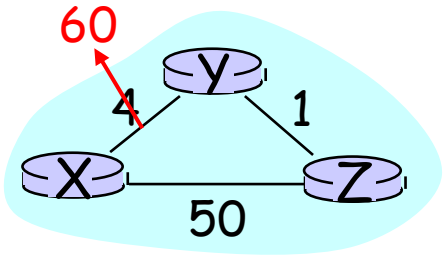


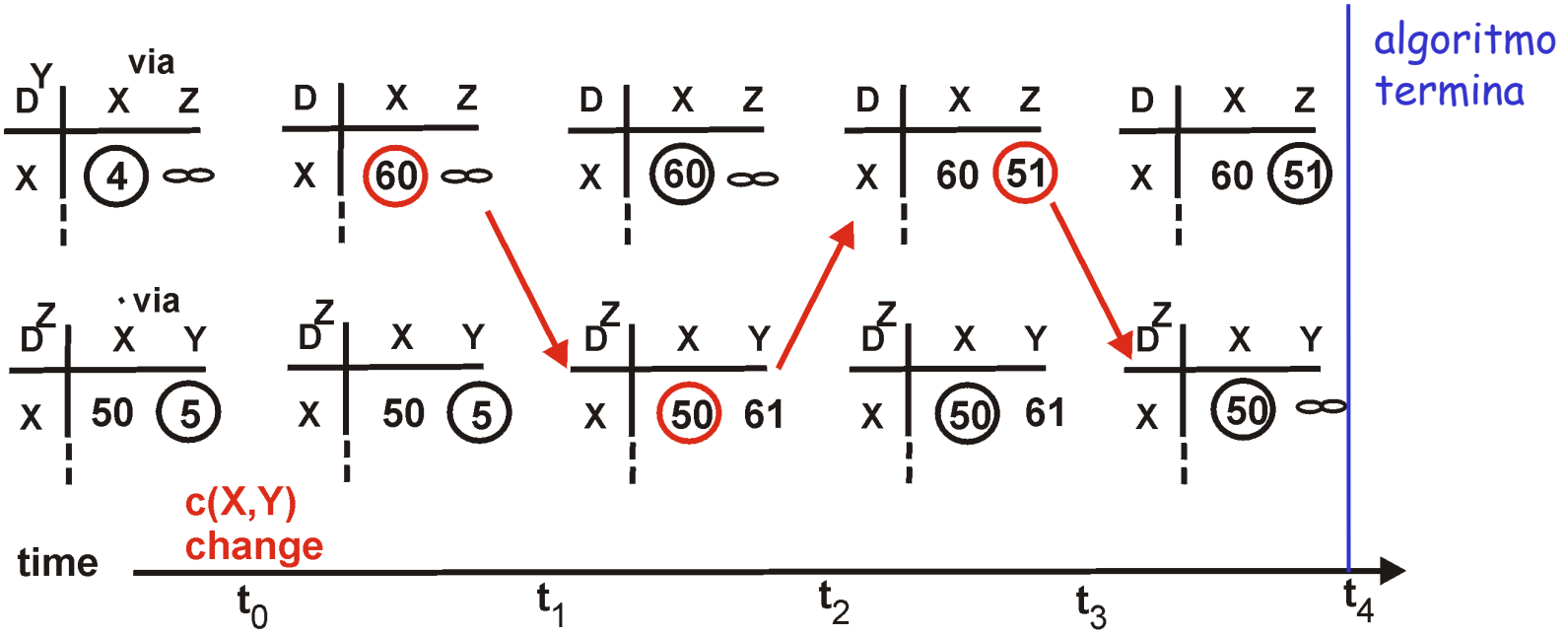


Vetor Distância: Poisoned Reverse

Se Z roteia através de Y para chegar a X :

- Z diz a Y que sua (de Z) distância para X é infinita (assim Y não roteará para X via Z)





Comparação entre os Algoritmos LS e DV

Complexidade

- LS: com n roteadores e E enlaces: o(n^2) mensagens enviadas
- DV: trocas somente entre vizinhos, tempo de convergência varia

Velocidade de convergência

- LS: algoritmo o(n^2) exige o(n^2) mensagens
  - ✓ Pode ter oscilações
- DV: tempo de convergência varia
  - ✓ Podem haver loops de roteamento
  - ✓ Problema da contagem ao infinito

Robustez: o que acontece se um roteador funciona mal?

- LS:
  - ✓ roteadores podem informar custos incorretos para os enlaces.
  - ✓ Cada roteador calcula somente sua própria tabela de roteamento
    - independente dos demais
- DV:
  - ✓ Nó pode informar caminhos com custo incorreto
    - custo errado se propaga
  - ✓ Tabela de cada roteador é usada por outros
    - Propagação de erros pela rede

Tornando o roteamento escalável

Problemas do mundo real

- roteadores não são todos idênticos
- na prática, a Internet é heterogênea: as redes possuem diferentes topologias e algoritmos de roteamento

Escala: bilhões de destinos:

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

Autonomia Administrativa

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

Abordagem da Internet p/ roteamento escalável

- Agrega roteadores em regiões: "sistemas autônomos " (**AS**)
- Roteadores no mesmo AS rodam o mesmo protocolo de roteamento
  - ✓ Protocolo de roteamento "Intra-AS"
- Roteadores em diferentes AS's podem rodar diferentes protocolos de roteamento

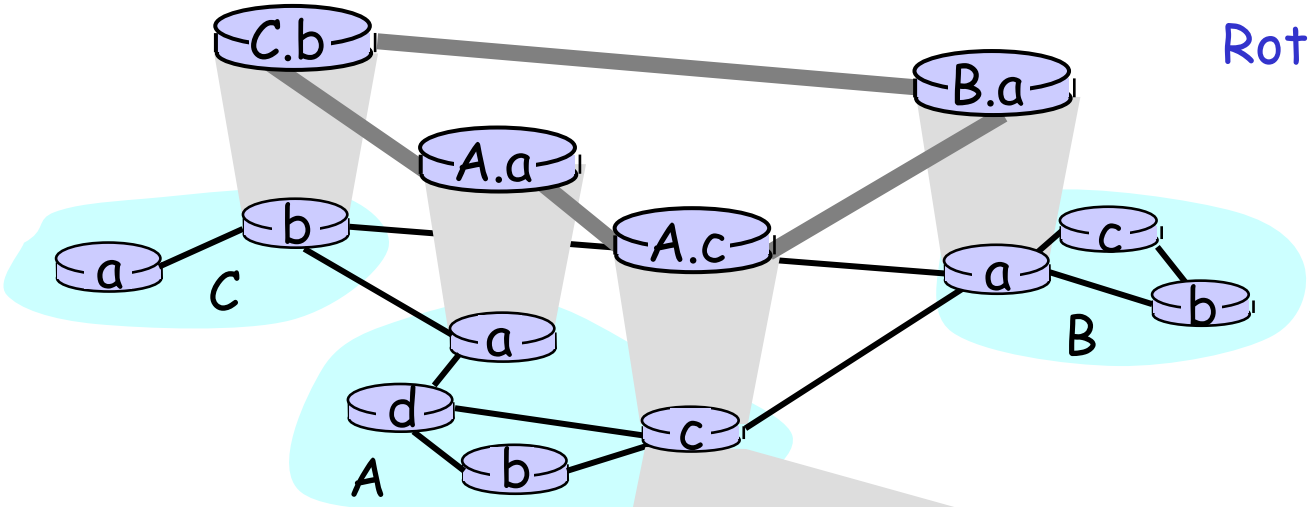
roteadores de borda

- Roteadores de interface de um AS com outros AS's
- Rodam protocolos de roteamento intra-AS com os outros roteadores do mesmo AS
- *Também são* responsáveis por enviar mensagens para fora do AS

↓

- Rodam *protocolo de roteamento inter-AS* com outros roteadores de borda (de outros AS's)

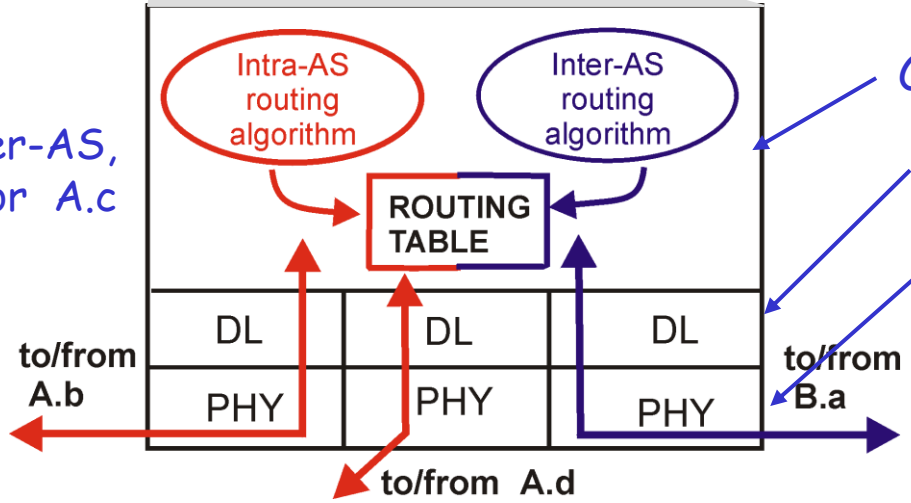
Roteamento Intra-AS e Inter-AS



Roteadores de Borda

- realizam roteamento inter-AS entre si
- realizam roteamento intra-AS com outros roteadores do mesmo AS

Roteamento inter-AS, intra-AS no roteador A.c



Camada de rede

Camada de enlace

Camada física

AS´s interconectadas

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

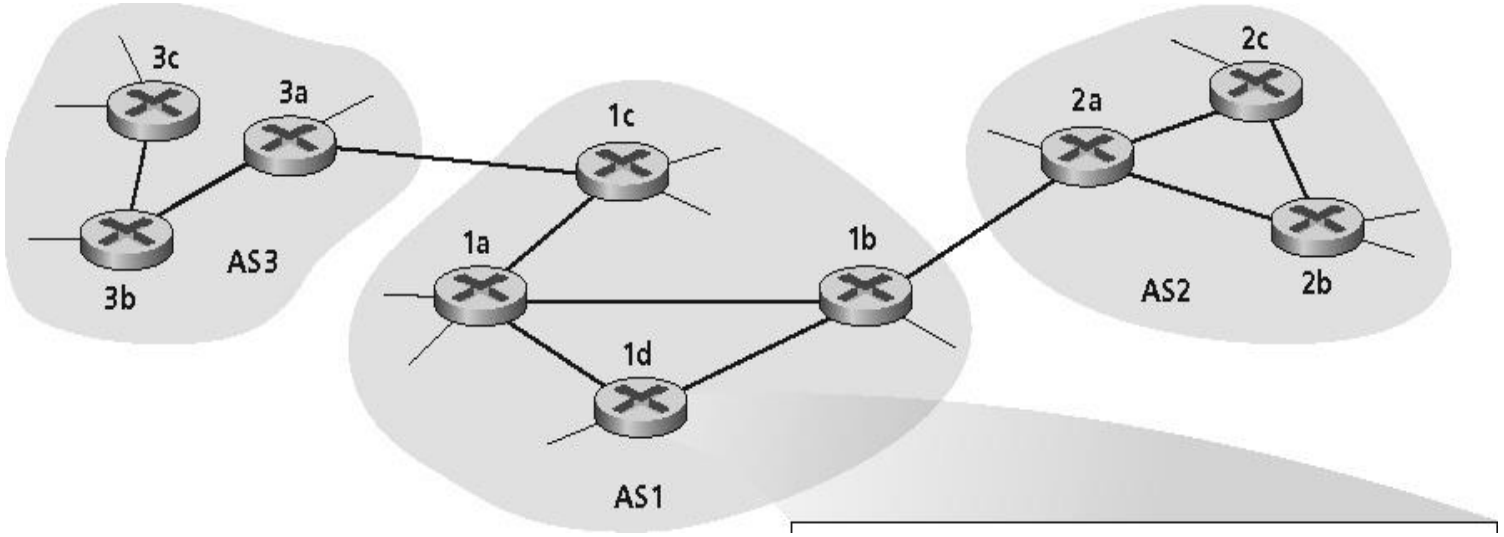
Tarefas Inter-AS

Q: Suponha que um roteador no AS1 receba um datagrama cujo destino seja fora do AS1. O roteador deveria encaminhar o pacote para os roteadores de borda, mas qual deles?

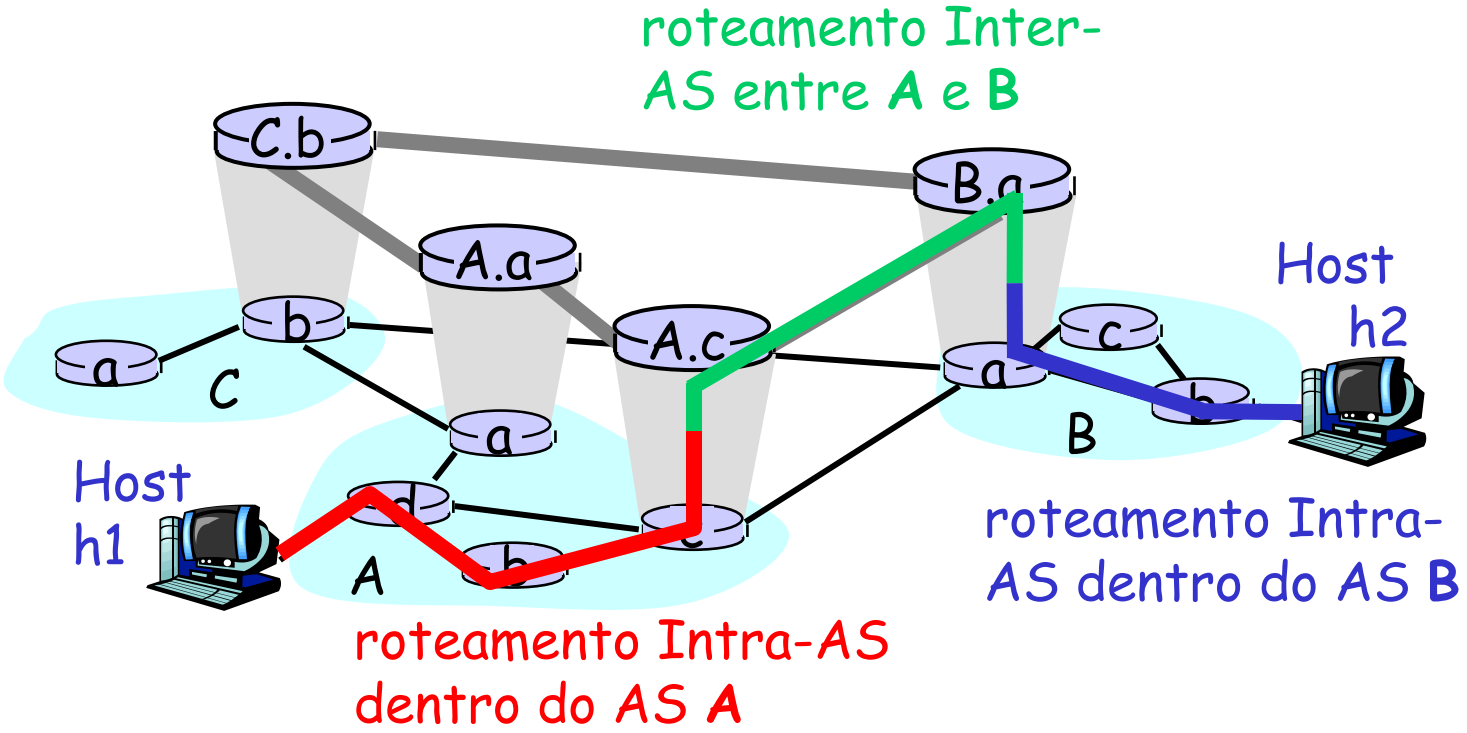
Roteador de borda em AS1 precisa:

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

Tarefa para o protocolo de roteamento inter-AS!



Roteamento Intra-AS e Inter-AS



- Rotas fim-a-fim (h1 p/ h2) são obtidas por meio da concatenação de rotas internas em vários AS´s e através das rotas inter-AS.