Redes de Computadores

Encaminhamento pelo Caminho mais Curto

(2) *Belman-Ford Routing*

Departamento de Informática da

FCT/UNL

Objetivos

- Em alternativa à técnica ***Link-State Routing*** é possível usar outro algoritmo de encaminhamento distribuído. O mesmo baseia-se na troca de informação entre vizinhos diretos sobre os **destinos** a que cada um é capaz de chegar
- A partir da visão dos destinos vistos pelos vizinhos, é possível um nó construir a sua própria visão e passá-la também aos seus vizinhos
- Se os nós forem passando esta informação uns aos outros, acabarão todos por ficar a saber a melhor maneira de chegar a cada destino
- Este tipo de aproximação permitiu construir um algoritmo de encaminhamento, designado ***Vetor de Distâncias***

Make everything as simple as possible, but not simpler.

– Autor: ***Albert Einstein (1879-1955)***

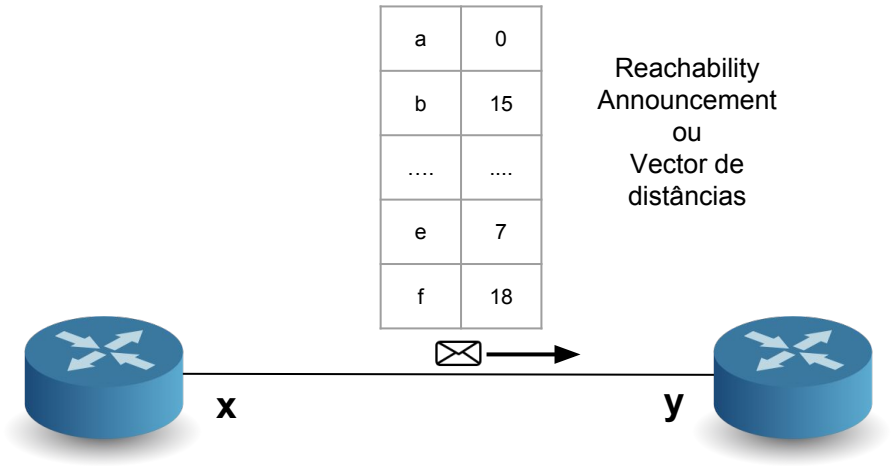
Ideia Base: Anúncios para a Vizinhança

- Filosofia de base do algoritmo
  - Se um nó dá acesso direto ao destino D
  - pode dizer aos vizinhos: "D é acessível por mim à distância 0"
  - e os vizinhos podem dizer aos vizinhos: "D é acessível por mim à distância …"
  - até que todos os nós sabem que existe o destino D e podem selecionar o melhor vizinho para lá chegar
- Os anúncios de vizinhança também se designam *Reachability Annoucements, Anúncios de visibilidade* ou Vetores de Distâncias, daí o nome do algoritmo
- Às vezes o algoritmo também é conhecido pelo nome dos seus primeiros inventores: Bellman-Ford

Algoritmo Bellman-Ford

- Filosofia de base do algoritmo
  - Se os vizinhos de um nó lhe comunicarem os destinos a que dão acesso, então:
  - esse nó fica a saber as diferentes maneiras de chegar até esses destinos
  - se sempre que vão alargando os seus horizontes, os nós passarem de novo essa informação aos seus vizinhos, então todos os nós passarão a conhecer formas de chegar a todos os destinos existentes
- Mas como selecionar o melhor caminho para um destino D que não é local ?
  - Cada nó anuncia o custo que acha que tem para chegar a cada destino D que conhece
  - Para cada destino D, um nó seleciona o vizinho V que minimiza: custo para chegar a V mais o custo que V anuncia para chegar D

Vetor de Distâncias



Processamento de um Anúncio

x — nó que enviou o anúncio
d —distance ou custo para chegar a x
reachability [1..k] — anúncio

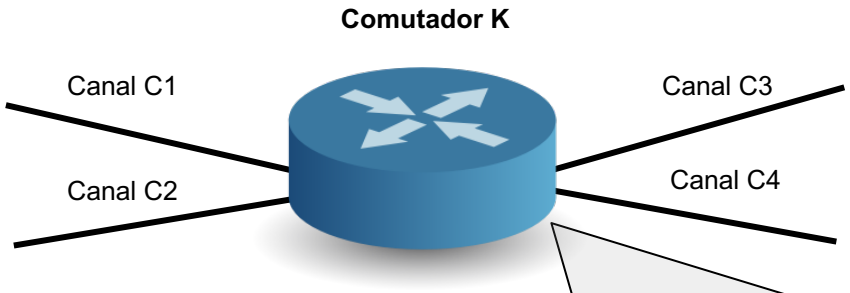
distance[1..k] —distance to each node
nexthop[1..k] — vizinho seleccionado para chegar a cada nó

```
for i  in 1..k do {
        if ( reachability[i] + d < distance[i] )   {
                distance[i] = reachability[i] + d
                nexthop[i] = x
        }
```

Vetores de Distâncias

- Cada nó X tem uma tabela de vizinhos diretos e o custo para lhes chegar e uma tabela com os destinos que conhece, os custos para lá chegar e as interfaces que correspondem ao início desse caminho
- Cada nó X anuncia aos vizinhos um anúncio com um vetor de distâncias de si até aos outros nós da rede que conhece
- Sempre que recebe um vetor de distancias dos vizinhos, o nó X guarda-o e atualiza a sua maneira de alcançar os diferentes destinos
- Assim, cada nó atualiza a sua tabela de encaminhamento e o vetor de distâncias que passará aos seus vizinhos

Como Funciona um Comutador VD



Tabelas de encaminhamento, anúncios, vizinhos (RIB - Routing Information Base)

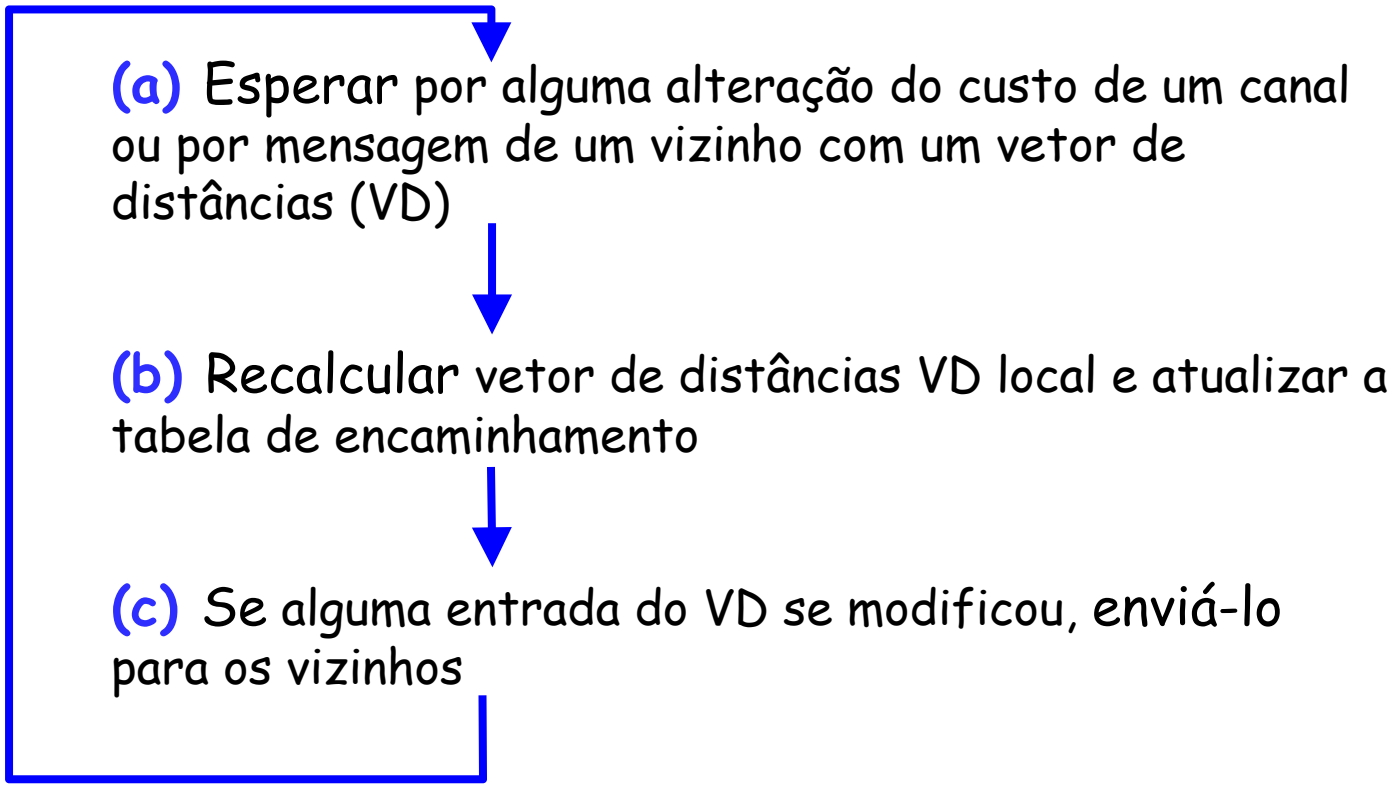
Destino	Tipo	Interface	Custo
A	...	–	0
B	...	C2	9
C	...	C3	10
D	...	C2	5
....	...	...	...
M		C4	9

Tabela de comutação (FIB - Forwarding Information Base)

Destino	Interface	Custo
A	–	0
B	C2	9
C	C3	10
D	C2	5
....	...	...
M	C4	9

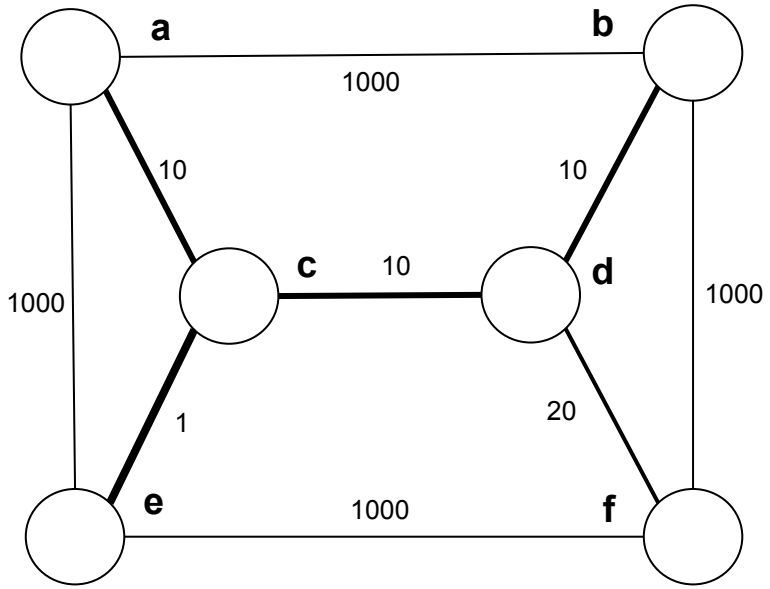
Funcionamento Distribuído

Em cada nó:



O tempo que as etapas (b) e (c) levam a executar é diferente em cada nó e desconhecido a priori, apenas se sabe que há progresso em cada nó

Exemplo de Execução



(a) - Rede

Destino	À distância	Via o vizinho (*next hop*)
a	0	___
b	1000	b
c	10	c
e	1000	e

(b) - FIB inicial do comutador a

Processamento do Primeiro Anúncio

Destino	À distância	Via o vizinho (*next hop*)
a	0	____
b	1000	b
c	10	c
e	1000	e

(a) - FIB inicial do comutador a

1 - vector distância enviado por b

Destino	a	b	d	f
Distância	1000	0	10	1000

2 - vector distância enviado por c

Destino	a	c	d	e
Distância	10	0	10	1

3 - vector distância enviado por e

Destino	a	c	e	f
Distância	1000	1	0	1000

(b) - Vectores distância recebidos pelo comutador a

Destino	À distância	Via o vizinho (*next hop*)
a	0	____
b	1000	b
c	10	c
d	1010 / 20	b / c
e	1000 / 11	e / c
f	2000	b

(c) - FIB final do comutador a

Estado das Tabelas Após uma Iteração

Dst	d.	*next hop*
a	1000	a
b	0	__
c	20	d
d	10	d
e	2000	f
f	30	d

FIB do comutador b

Dst	d.	*next hop*
a	10	a
b	20	d
c	0	__
d	10	d
e	1	e
f	30	d

FIB do comutador c

Dst	d.	*next hop*
a	20	c
b	10	b
c	10	c
d	0	__
e	11	c
f	20	f

FIB do comutador d

Dst	d.	*next hop*
a	11	c
b	2000	a
c	1	c
d	11	c
e	0	__
f	1000	f

FIB do comutador e

Dst	d.	*next hop*
a	2000	b
b	30	d
c	30	d
d	20	d
e	1000	e
f	0	__

FIB do comutador f

Processamento do Segundo Anúncio

Destino	À distância	Via o vizinho (*next hop*)
a	0	___
b	1000	b
c	10	c
d	20	c
e	11	c
f	2000	b

(a) - FIB inicial do comutador a

1 - vector distância enviado por b

Destino	a	b	c	d	e	f
Distância	1000	0	20	10	2000	30

2 - vector distância enviado por c

Destino	a	b	c	d	e	f
Distância	10	20	0	10	1	30

3 - vector distância enviado por e

Destino	a	b	c	d	e	f
Distância	11	2000	1	11	0	1000

(b) - Vectores distância recebidos pelo comutador a

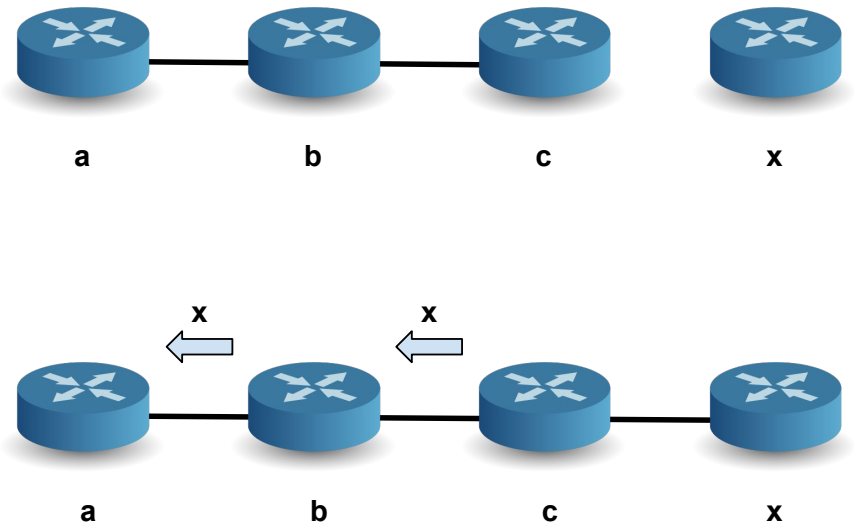
Destino	À distância	Via o vizinho (*next hop*)
a	0	___
b	30	c
c	10	c
d	20	c
e	11	c
f	1030 / 40	b / c

(c) - FIB final do comutador a

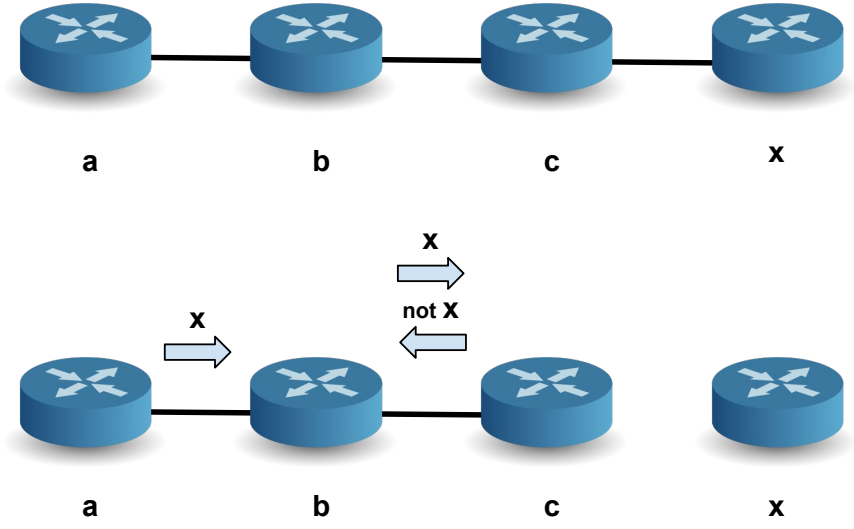
Convergência do Protocolo

- O protocolo Vetor de Distâncias requer em cada nó um processamento bastante simples
- Quando a rede é ativada, os anúncios propagam-se rapidamente e todos os nós passam a conhecer o melhor caminho para cada destino
- A convergência apenas está dependente do diâmetro da rede e é rápida
- Infelizmente, quando o custo de um canal aumenta, pode desencadear um fenómeno chamado contagem para o infinito
- Esta característica também se designa por "As boas notícias andam rápido mas as más devagar"

Exemplos



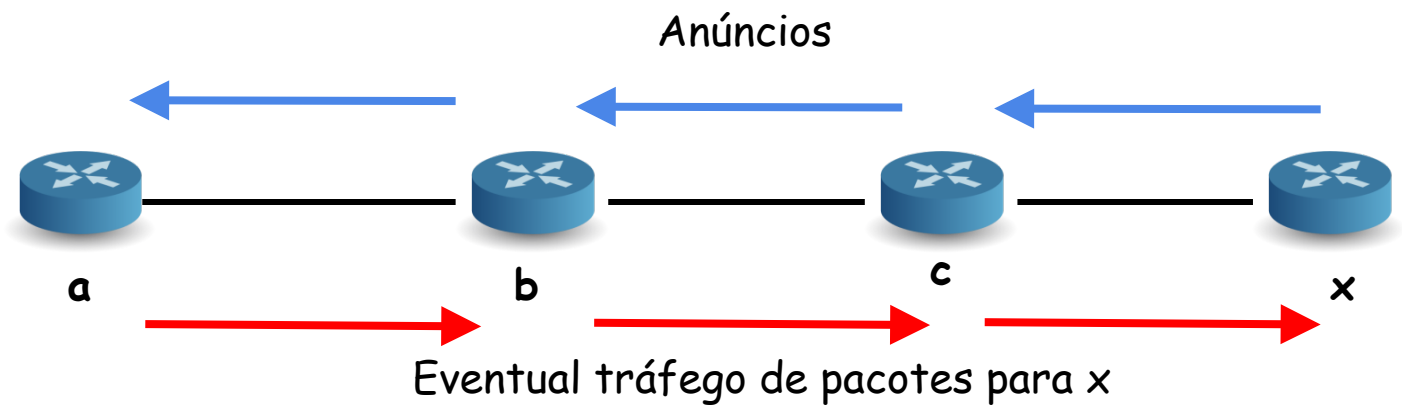
(a) - Good News Travel Fast



(b) - Bad News Travel Slowly

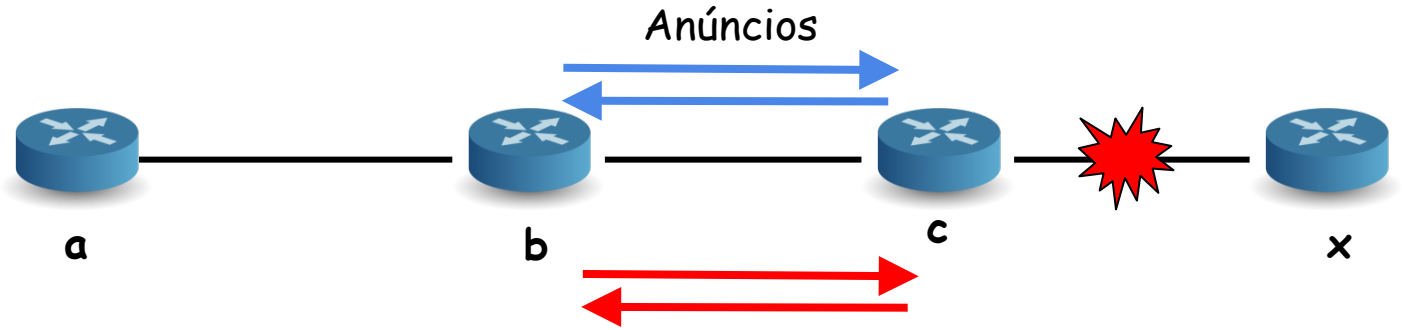
As Boas Notícias Correm Depressa

- Basta que c anuncie a b e que este anuncie a a que o destino x está acessível para que a distância e o caminho para x seja conhecido por toda a rede
  - Este comportamento chama-se "As boas notícias correm depressa" (*good news travel fast*)



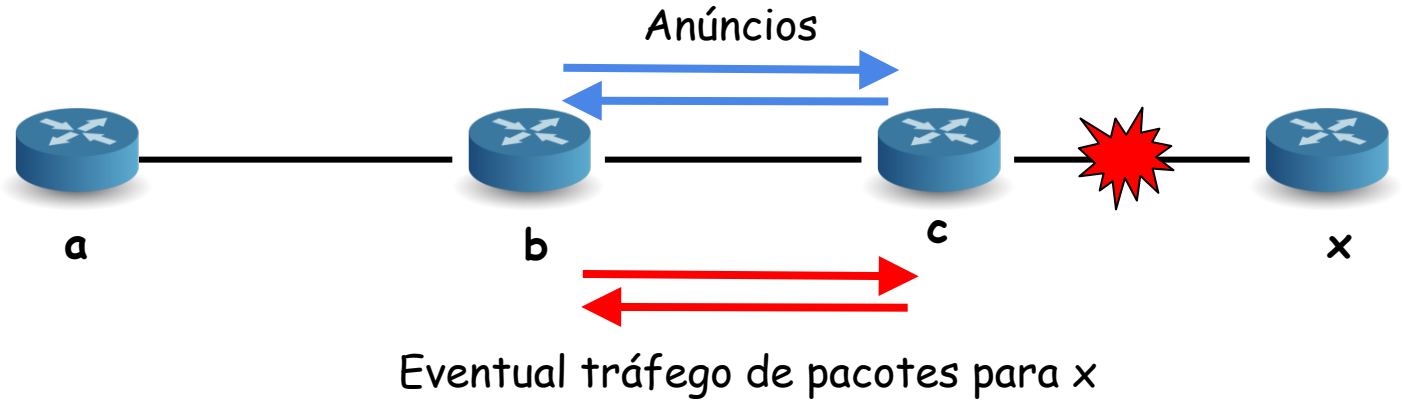
Mas as Más Notícias Andam Devagar

- Quando o custo de um canal aumenta, o algoritmo propaga a nova informação lentamente e às vezes de forma errada
- Por exemplo, se o canal entre x e c passar de custo 1 a infinito, mas c usar o fato de que b também lhe anuncia que conhece um caminho para x, ou se c guardou o vetor de distancias que b lhe enviou antes, c pode, erradamente, pensar que pode chegar a x via b
- Se isto acontecer aparecerá um *ping pong* de anúncios entre b e c e um ciclo no encaminhamento (para um buraco negro), que se costuma chamar um *routing loop / routing black hole*



Contagem para o Infinito

- Com efeito b pode acreditar que há um melhor caminho para x via c, mas c acha que é via b que chega a x
- Por cada vetor de distancias trocado entre ambos, o custo para chegar a x vai aumentando de uma unidade (admitindo que o custo de cada canal é 1)
- E o ciclo só vai parar quando o custo chegar a "infinito"
- Este comportamento chama-se "As más notícias andam devagar" (*bad news travel slowly*) ou contagem para infinito (*count to infinity*)



Soluções

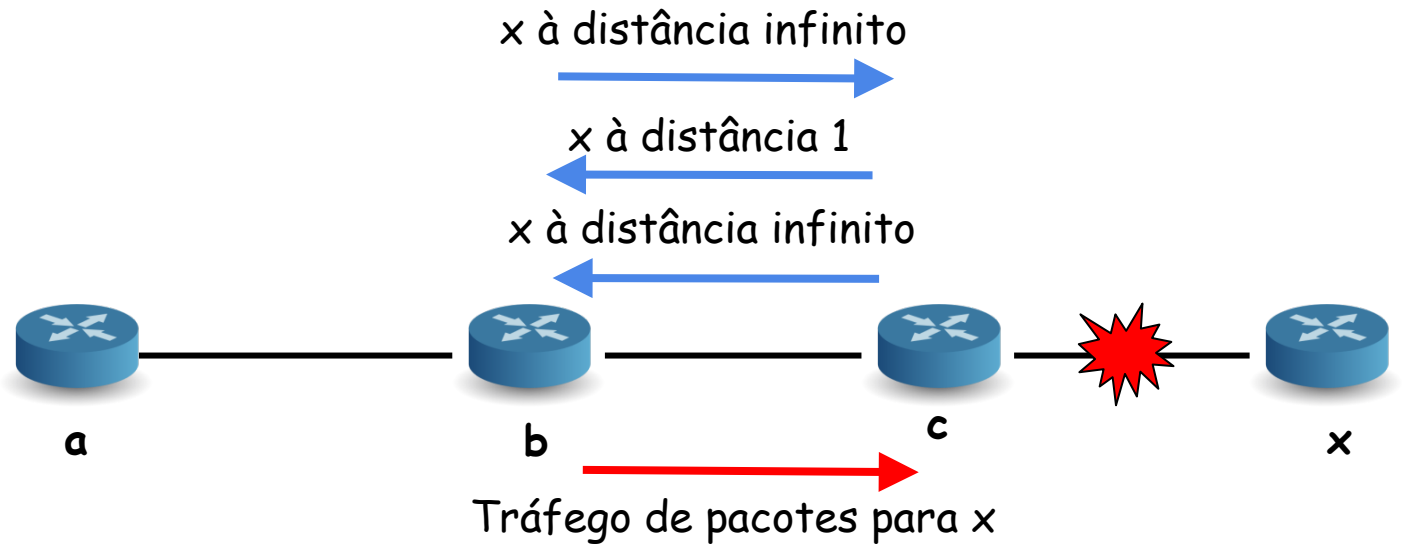
- **"Infinito" é modelizado por um número relativamente baixo** para aumentar a velocidade de convergência (tal solução implica que este algoritmo não pode ser aplicado a situações em que o diâmetro da rede é muito grande ou os custos representam o débito)
  - Por exemplo, no protocolo RIP, que utiliza o algoritmo vector de distancias e todos os canais têm custo 1, "infinito" = 16

- **"Split horizon with poisoned reverse"** – um nó R1 anuncia ao nó R2 todos os destinos para os quais R1 usa R2 como melhor caminho à distância infinito; ou seja, R1 anuncia a R2 com distância infinito tudo aquilo que R1 acha que está por detrás de R2. Por outras palavras, não tenho nada de novo a anunciar aos nós que uso para enviar pacotes para um dado destino
- Outras soluções baseadas em temporizadores (não tratadas nesta disciplina)

Anúncios Envenenados no Sentido Inverso

Se b vai para x via c, então anuncia a c que vê x à distância infinito.

Desta forma c nunca usará o caminho via b para alcançar x.



Detalhes de Implementação

- Os anúncios podem ser passados entre nós de comutação de forma fiável ou não fiável
- A solução mais simples é usar UDP, mas nesse caso podem perder-se anúncios; uma solução simples consiste em repetir os anúncios periodicamente
- Quando os anúncios são passados de forma fiável (e.g. TCP) podem transmitir-se apenas as alterações e não todo o vetor
- Sempre que um nó concluí que há alterações pode desencadear imediatamente um anúncio para que o algoritmo convirja mais rapidamente

Tempos e Contagem para o Infinito

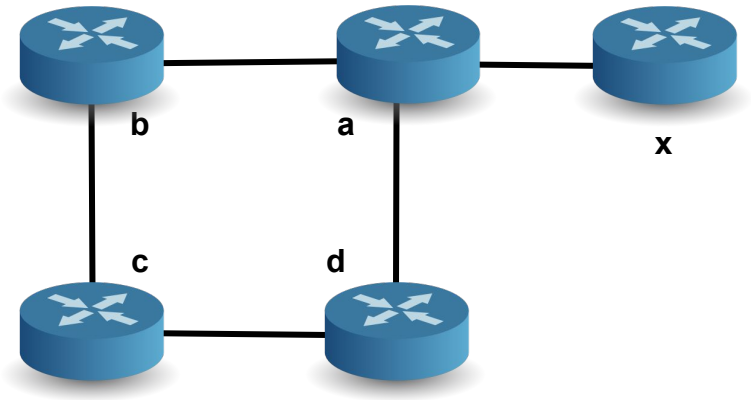
- Não está definida uma ordem de processamento: um comutador processa primeiro todos os anúncios recebidos ou logo que um anúncio é processado, e há alterações, envia imediatamente novos anúncios?

- A velocidade de propagação das novidades depende dos nós, da rede e das características da rede

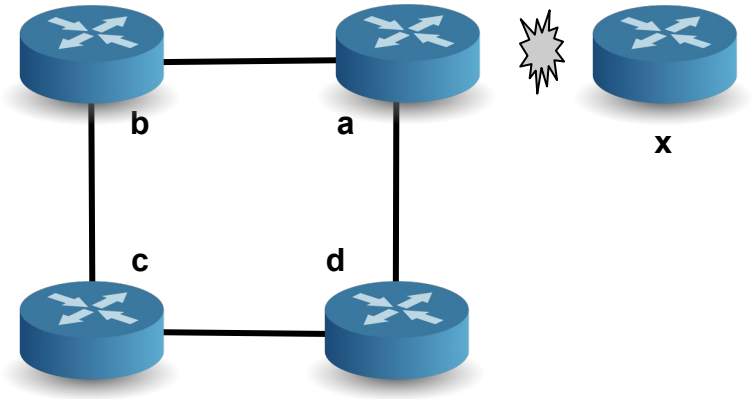
- Se a transmissão de anúncios for por meios não fiáveis, o tempo para que alguma novidade seja propagada aumenta, pois a mesma só será recebida da próxima vez que houver um anúncio periódico. Por outro lado, se um pacote se perder esse tempo também aumenta

Poison Reverse é Suficiente ?

Poison reverse não é suficiente para garantir que não aparecem episódios de contagem para o infinito. Ver o seguinte exemplo.



(a) - Rede antes da avaria



(b) - Rede depois da avaria

c recebe de d anúncios de acessibilidade de x (*split horizon* não bloqueia este anúncio pois c usa d para chegar a x)

Caso a ligação ax vá abaixo, b pode receber essa informação de a, mas antes de c também a receber de d, c pode anunciar a b a acessibilidade de x através de si (isto é, o caminho via d que b ainda não sabe que também está inacessível).

Resumo

- Os nós passam aos seus vizinhos a sua visão da rede em anúncios que lhes permitem aprender que existem novos destinos ou melhores caminhos do que os que já conheciam

- Quando o processo estabiliza, o algoritmo permite a cada nó conhecer um caminho mais curto para cada destino
- Cada nó passa alguma informação que conhece sobre todos os destinos, mas apenas aos seus vizinhos, e o cálculo para atualização das tabelas é trivial.
- O algoritmo caracteriza-se pela simplicidade, baixa ocupação de memória, e poucos tipos de mensagens
- Cada nó processa o algoritmo de forma assíncrona e autónoma
- A menos do problema da convergência ("durante a propagação de más notícias"), é um algoritmo muito simples

Conclusões Sobre o Algoritmo

- É um algoritmo muito simples do ponto de vista computacional
  - Converge rapidamente face a "boas notícias"
  - É computacionalmente trivial e requer poucos recursos de memória
  - Mas tem o problema da "contagem para infinito"
- Solução adequada em redes simples e de pequeno diâmetro, mas
  - Em redes com muitos destinos a dimensão dos anúncios cresce
  - Em redes com muitos canais e instabilidade, a convergência (de más notícias) é lenta.

Conclusões

- O encaminhamento baseia-se em algoritmos distribuídos
  - Para reagir a alterações da configuração da rede e
  - Computar os caminhos mais curtos
- Dois algoritmos principais
  - *Dijkstra* → *link-state routing* (e.g., OSPF e IS-IS)
  - *Bellman-Ford* → *distance vector routing* (e.g., RIP)
- Convergência – quando há uma alteração na rede, quanto tempo levam todos os nós a atuar de acordo com a mesma?
  - É necessário transitar de uma topologia para outra
  - Durante a transição podem surgir inconsistências e perdem-se pacotes pelo que a transição deve ser rápida
  - Trata-se de uma questão crítica para o funcionamento de uma rede de grande dimensão