Camada de Rede - Roteamento

Sumário

- Introdução;
- Algoritmos de Roteamento:
  - Roteamento de estado de enlace (link-state LS)
  - Roteamento de Vetor de Distância (distance-vector DV)
  - Comparação entre os algoritmos LS e DV
  - Roteamento Hierárquico
- Roteamento na Internet:
  - RIP

Sumário

- Roteamento na Internet:
  - OSPF
  - BGP
- Roteamento broadcast e multicast

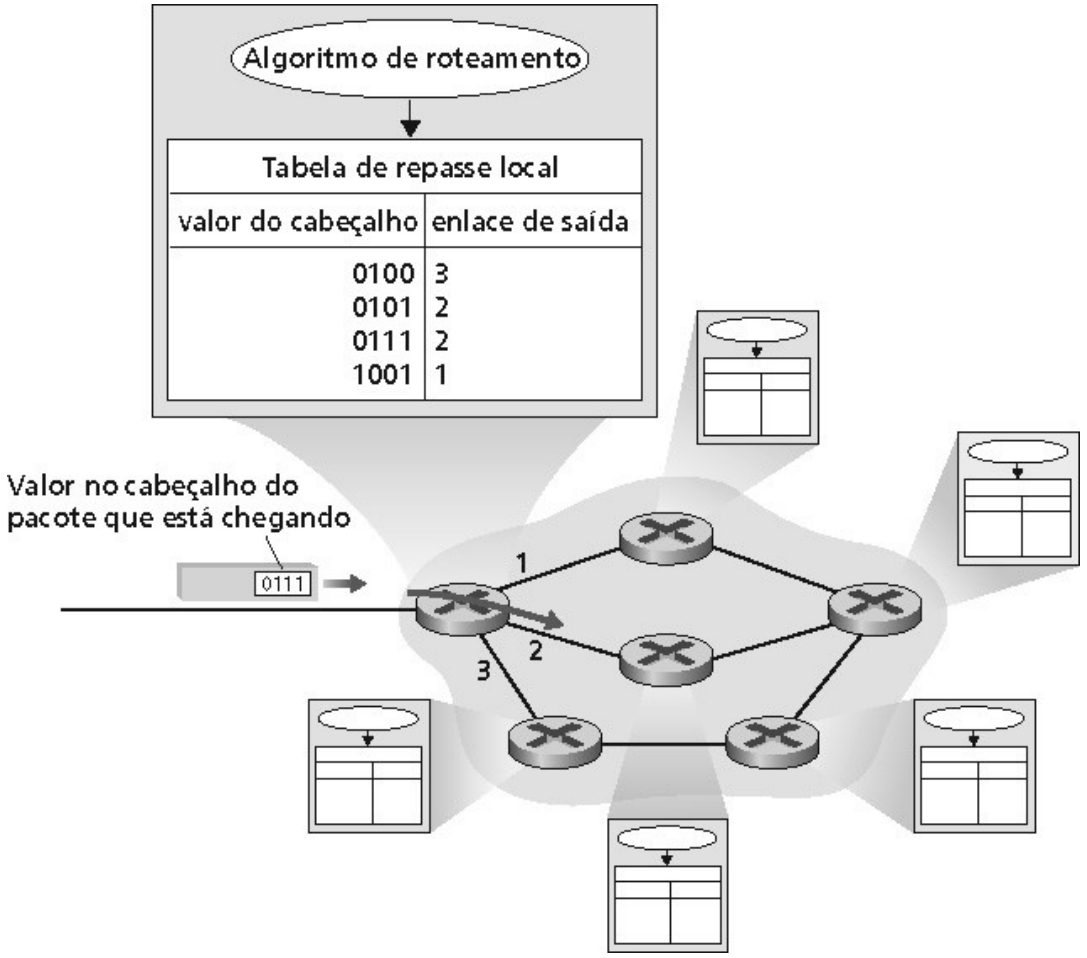
Introdução

- Vimos na aula anterior uma distinção importante entre as funções de repasse e roteamento:
  - Repasse: envolve a transferência de um pacote de um enlace de entrada para um enlace de saída dentro de um único roteador;
  - Roteamento: envolve todos os roteadores (ou rotas) que os pacotes percorrem em suas viagens do nó de origem ao nó de destino;

- Agora iremos aprofundar em um importante tópico da camada de rede, os algoritmos de roteamento;

Introdução

- O algoritmo de roteamento é a parte do software da camada de rede responspavel pela decisão sobre a linha de saída a ser usada na transmissão do pacote de entrada:



Introdução

- Normalmente um hospedeiro está ligado diretamente a um roteador, o roteador default;
- O problema de rotear um pacote do hospedeiro de origem até o hospedeiro destinatário se reduz, claramente, ao problema de rotear o pacote da fonte ao roteador de destino;
- Portanto, dado um conjunto de roteadores conectados por enlaces, um algoritmo de roteamento descobre um "bom" caminho entre o roteador de fonte e o roteador de destino;

Introdução

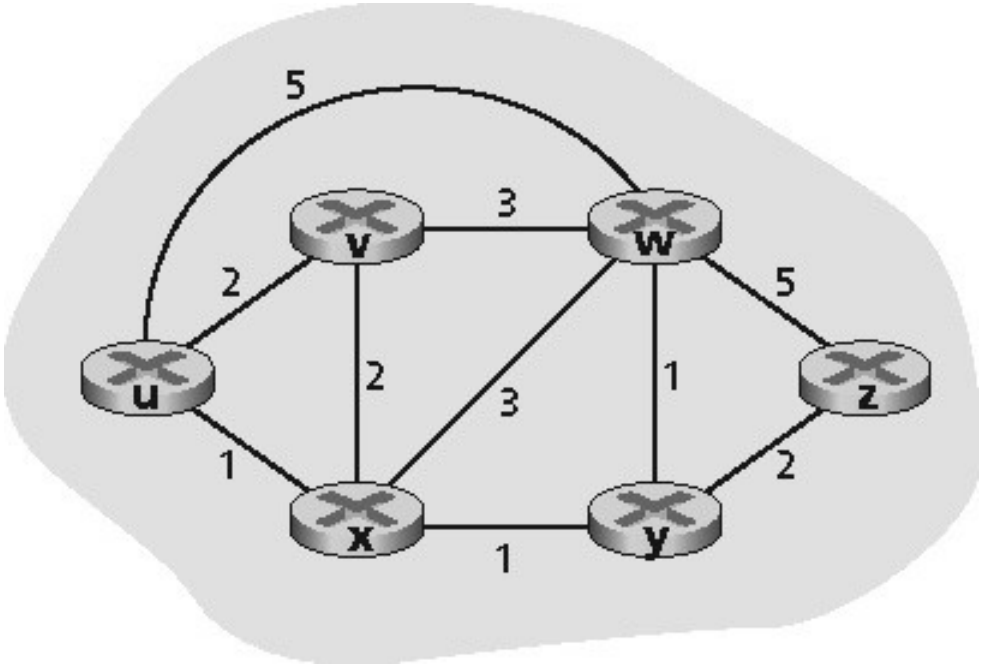
- Um “bom” caminho é aquele que tem o “menor custo”, entretanto, teremos inúmeras questões do mundo real que influenciarão na decisão do “bom” caminho:

Calçada

Atalho???

Introdução

- Qual seria o caminho de menor custo entre os nós u e z?
  - Você verificou todos os 17 possíveis caminho?
  - Exemplo de algoritmo de roteamento centralizado (o seu cérebro)



Introdução

- Classificação dos algoritmos de roteamento:
  - Algoritmo de roteamento global: calcula o caminho de menor custo entre uma fonte e um destino usando conhecimento completo e global sobre a rede. O cálculo pode ser rodado em um local ou duplicado em vários locais;
  - Algoritmo de roteamento descentralizado: calcula o caminho de menor custo de modo interativo e distribuído. Cada nó começa sabendo apenas os custos dos enlaces diretamente ligados a ele;

Introdução

- Uma segunda maneira geral de classificar algoritmos de roteamente é como estáticos e dinâmicos:
  - Estáticos: as rotas mudam muito lentamente ao longo do tempo, muitas vezes como resultado de intervenção humana (editando a tabela de repasse do roteador);
  - Dinâmico: mudam os caminhos de roteamento à medida que mudam as cargas de tráfego ou a topologia da rede. Mais suscetíveis a problemas como loops de roteamento e oscilação em rotas;

Introdução

- Uma terceira maneira de classificar algoritmos de roteamento é como sensíveis à carga ou insensíveis à carga:
  - Algoritmos sensíveis à carga: atribuem custos ao enlace para refletir o nível corrente de congestionamento no enlace subjacente.
  - Algoritmos insensíveis à carga: consideram que o custo de um enlace não reflete explicitamente seu nível de congestionamento corrente (algoritmos utilizados na Internet)

Roteamento de estado de enlace (link-state LS)

- Nesse algoritmo lembre-se que a topologia da rede e todos os custos de enlace são conhecidos;
  - Na prática, cada nó transmite pacotes de estado de enlace a todos os outros nós da rede;

- O algoritmo de roteamento de estado de enlace á conhecido como algoritmo de Dijkstra, o nome do seu inventor:
  - O algoritmo de Dijkstra calcula o caminho de menor custo entre um nó e todos os outros da rede

Roteamento de estado de enlace (link-state LS)

- Inicialmente, vamos definir a seguinte notação:
  - C(i, j): custo do enlace do nó i ao nó j. Custo é infinito se não houver ligação entre i e j;
  - D(v): custo do caminho de menor custo entre o nó da fonte e o destino v até que essa iteração do algoritmo;
  - p(v): nó anterior (vizinho de v) ao longo do caminho de menor custo corrente desde a fonte até v;
  - N': subconjunto de nós; v pertence a N' se o caminho de menor custo entre a fonte e v for inequivocamente conhecido;

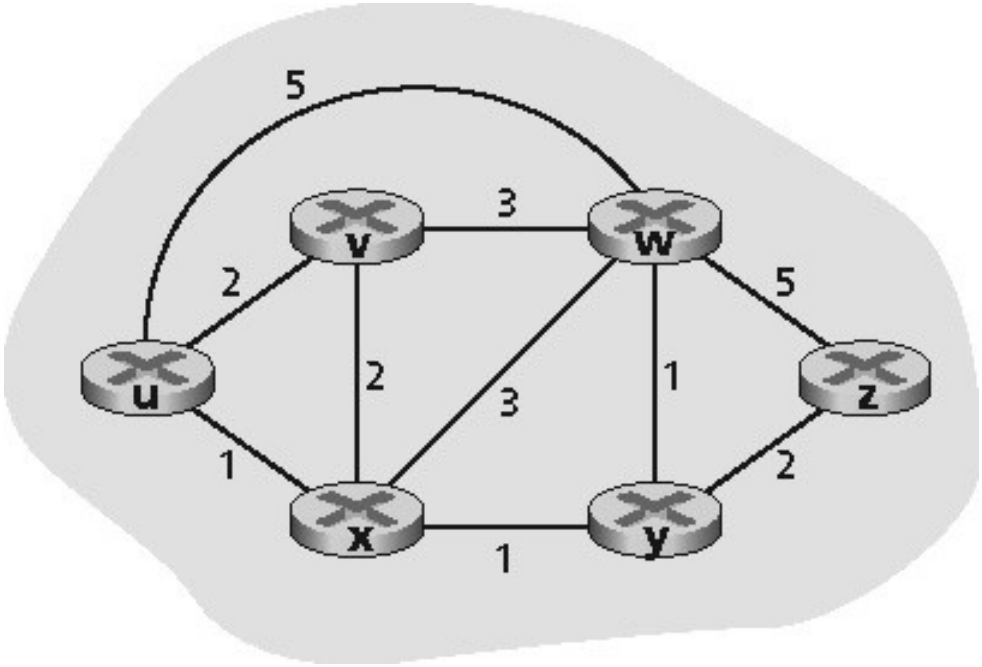
Roteamento de estado de enlace (link-state LS)

```
1  Inicialização:
2   N' = {u}
3   para todos os nós v
4     se v é adjacente a u
5       então D(v) = c(u,v)
6       senão D(v) = ∞
7
8  Loop
9    ache w não em N' tal que D(w) é um mínimo
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 ou 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'
```

Roteamento de estado de enlace (link-state LS)

- Tomemos como exemplo a rede da figura abaixo:

  - O algoritmo consiste em uma etapa de inicialização seguida de um loop;
  - O número de vezes que o loop é rodado é igual ao número de nós na rede;



Roteamento de estado de enlace (link-state LS)

- Vamos detalhar alguns dos primeiros estágios:
  - Inicialização: os caminhos de menor custo correntemente conhecidos de u até os vizinhos diretamente ligados a ele (v, w e x) são inicializados para 2, 1 e 5, respectivamente. Os custos até y e z são estabelecidos como infinito;

Passo	Início N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
→ 0	u	2,u	5,u	1,u	∞	∞

  - Na 1ª iteração: examinamos os nós que ainda não foram adicionados ao conjunto N' e descobrimos o nó de menor custo ao final da interação anterior (nó x), então, x é adicionado ao conjunto N'.

Roteamento de estado de enlace (link-state LS)

- Continuação:
  - A linha 12 do algoritmo é então rodada para atualizar D(v) para todos os nós v, produzindo o seguinte resultado:

Passo	Início N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
→ 0	u	2,u	5,u	1,u	∞	∞
→ 1	ux	2,u	4,x		2,x	∞

  - O custo do caminho até v não muda. O custo do caminho até w através do nó x é 4, portanto, a tabela é atualizada. De maneira semelhante, o custo até y (através de x) é computado como 2.

Roteamento de estado de enlace (link-state LS)

- Continuação:
  - Segunda iteração: verificamos que os nós v e y são os que têm os caminhos de menor custo (2); decidimos o empate arbitrariamente e adicionamos o y ao conjunto N';
  - O custo dos nós remanescentes que ainda não estão em N' (isto é, nós v, w e z) são atualizados pela linha 12 do algoritmo Ls, produzinho o seguinte resultado:

Passo	Início N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
→ 0	u	2,u	5,u	1,u	∞	∞
→ 1	ux	2,u	4,x		2,x	∞
→ 2	uxy	2,u	3,y			4,y

Roteamento de estado de enlace (link-state LS)

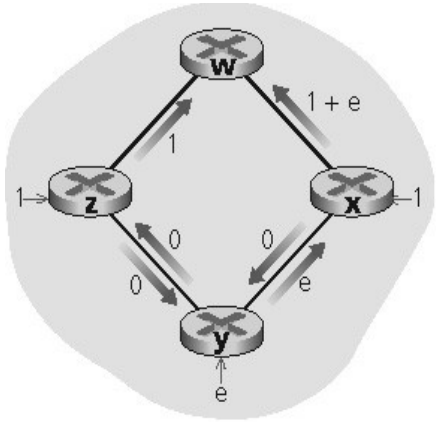
- Ao final do algoritmo LS, temos, para cada nó, seu predecessor ao longo do caminho de menor custo a partir do nó da fonte:
  - Temos também o predecessor para cada um desses predecessores, veja:

Passo	Início N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	u	2,u	5,u	1,u	∞	∞
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					

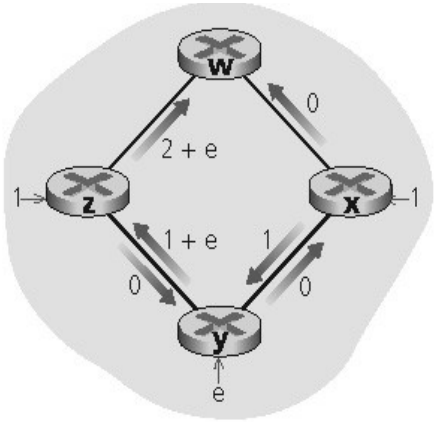
Roteamento de estado de enlace (link-state LS)

- Qual é a complexidade do cálculo desse algoritmo?
  - Cada iteração precisa verificar todos os nós w, que não estão em N
  - Portanto: n(n+1)/2 comparações; complexidade $O(n^2)$
  - Implementações mais eficientes: O(nlogn)

- Agora, consideremos a seguinte patologia:
  - Enlaces não simétricos, portanto, c(u,v) pode ser diferente de c(v,u).



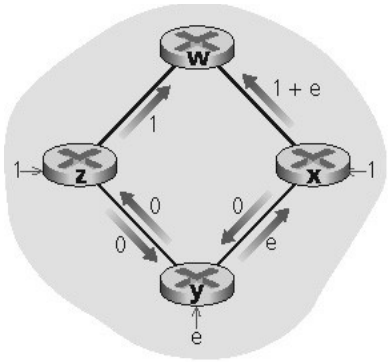
a. Roteamento inicial



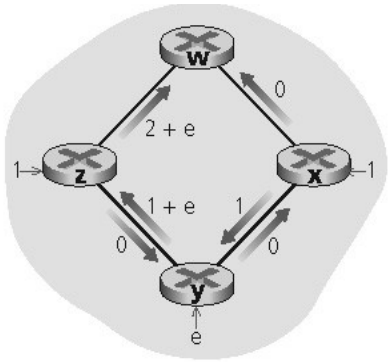
b. *x*, *y* detectam melhor caminho até *w* em sentido horário

Roteamento de estado de enlace (link-state LS)

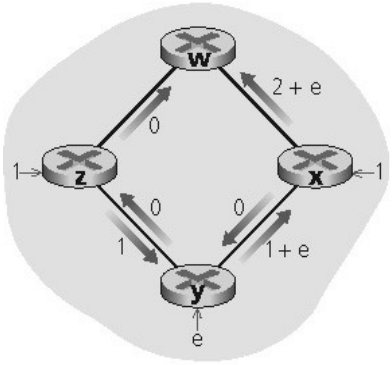
- Nesse exemplo, o nó z origina uma unidade de tráfego destinada a w:
  - Resultado: Oscilações entre o menor caminho;



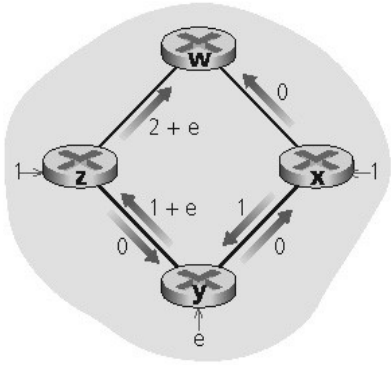
a. Roteamento inicial



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



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



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

  - Solução: variar o instante em que cada rodeador roda o algoritmo;

Roteamento de Vetor de Distância (distance-vector DV)

- Enquanto o algoritmo LS usa informação global, o algoritmo de vetor de distâncias é:
  - Distribuído: cada nó recebe alguma informação de um ou mais vizinhos diretamente ligado a ele;
  - Iterativo: porque esse processo continua até que mais nenhuma informação seja trocada entre vizinhos;
  - Assíncrono: não requer que todos os nós rodem simultaneamente;

- Primeiramente, compreendemos a equação de Bellman-Ford que relaciona os menores custos:

Roteamento de Vetor de Distância (distance-vector DV)

- Equação de Bellman-Ford:

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

  - **$d_x(y)$:** custo do caminho de menor custo de x para y
  - **$\min_v$:**calculado sobre todos os vizinhos de x

- Realmente, se após transitarmos de x para v tomarmos o caminho de menor custo de v a y, o custo do caminho será $c(x,v) + d_v(y)$:

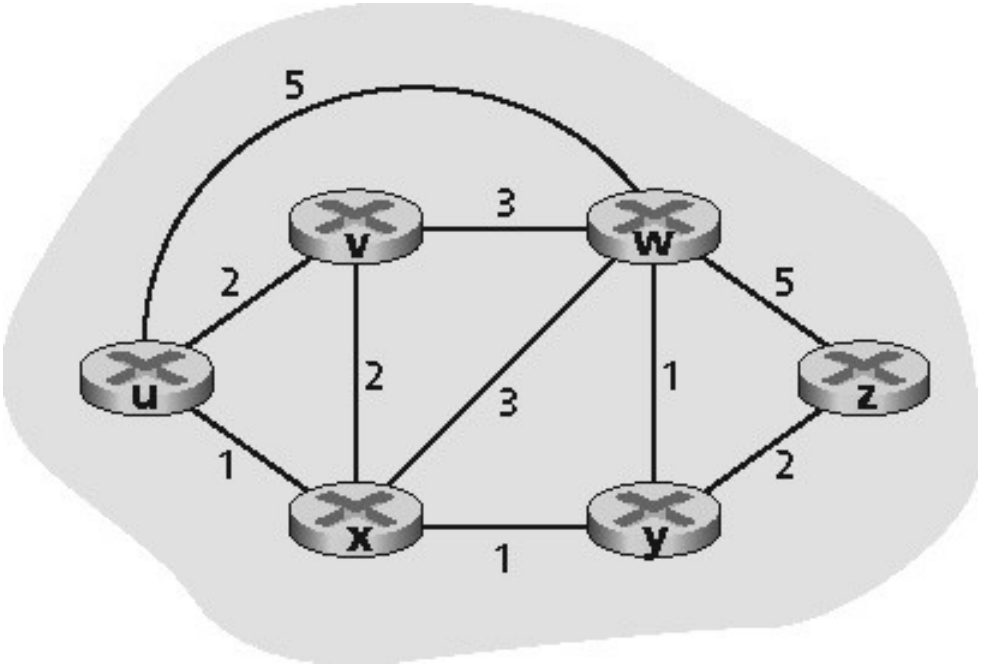
Roteamento de Vetor de Distância (distance-vector DV)

- Vejamos a aplicação do algoritmo de Bellman-Ford na rede da figura abaixo:

$d_u(z) = \min \{ c(u,a) + d_a(z) \}$

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

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



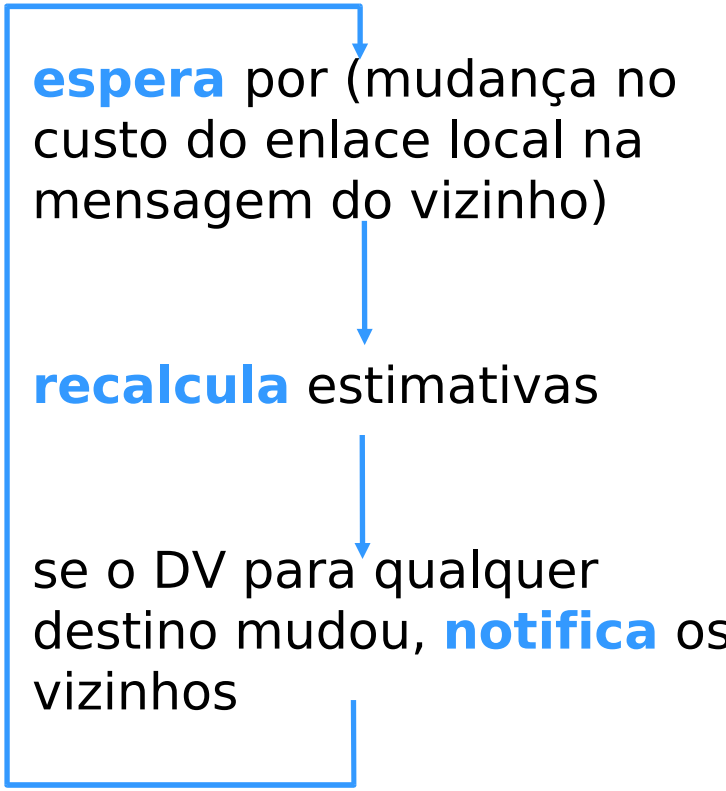
Roteamento de Vetor de Distância (distance-vector DV)

- Com o algoritmo de Bellman-Ford o algoritmo DV mantém em cada nó x os seguintes dados de roteamento:
  - Para cada vizinho v, o custo c(x,y) de x até o vizinho diretamente ligado a ele, v
  - O vetor de distância do nó x ($\mathbf{D_x = [D_x(y): y \epsilon N]}$) contendo a estimativa de x para seus custos até todos os destinos y, em N
  - Os vetores de distância ($\mathbf{D_v = [D_v(y): y \epsilon N]}$)de seus vizinhos para cada vizinho v de x;

Roteamento de Vetor de Distância (distance-vector DV)

- No algoritmo distribuído, assíncrono, cada nó envia, a intervalos regulares, uma cópia do seu vetor de distância a cada um de seus vizinhos:

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



Roteamento de Vetor de Distância (distance-vector DV)

- Vejamos a aplicação do algoritmo DV na rede da figura abaixo:
  - Inicialmente as três tabelas de roteamento inclui seu próprio vetor de distância e os vetores de cada um de seus vizinho

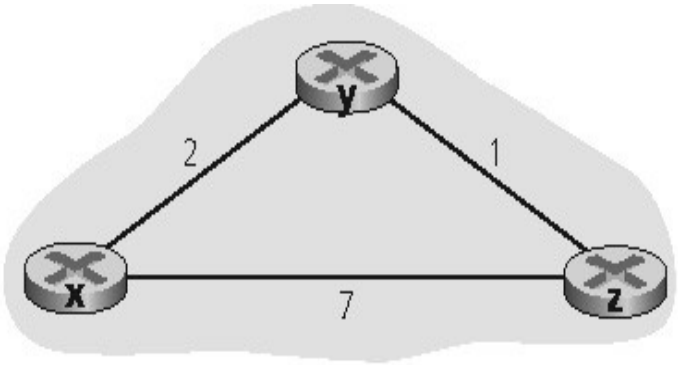


Tabela do nó x

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

Tabela do nó y

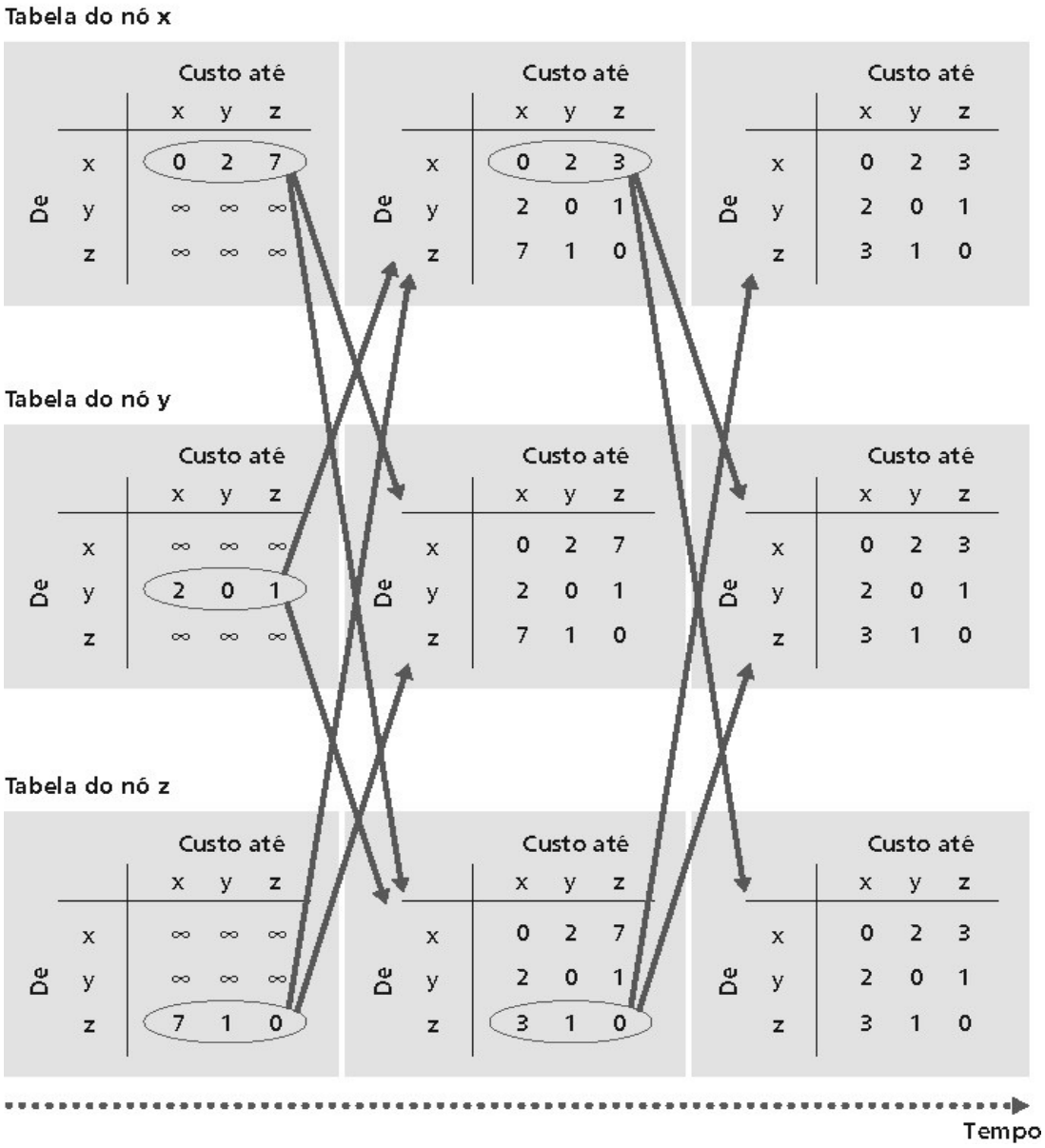
De	Custo até x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

Tabela do nó z

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

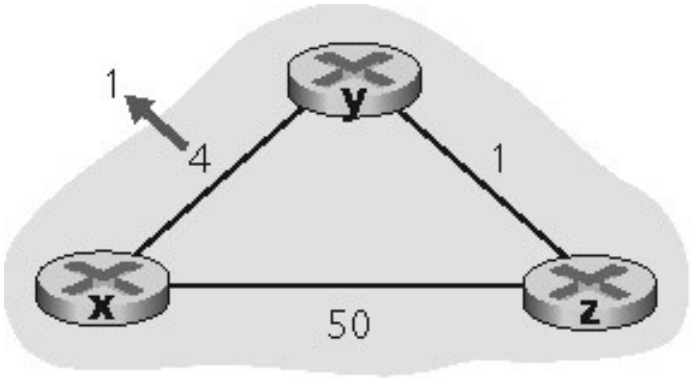
Roteamento DV

- Após a inicialização, cada nó envia seu vetor de distância a cada um de seus dois vizinhos:
  - Depois que recalculam seus vetores de distâncias, os nós enviam novamente seus vetores



Roteamento de Vetor de Distância (distance-vector DV)

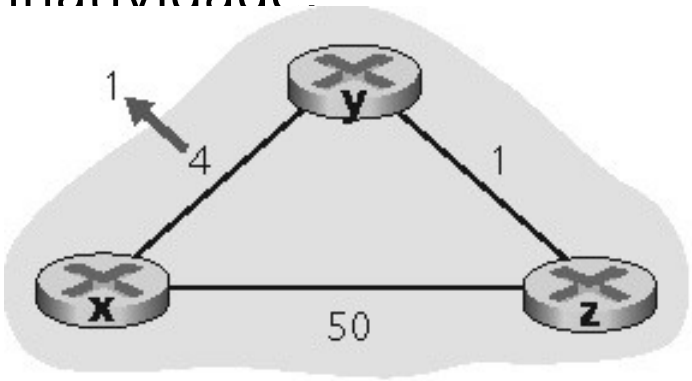
- Quando um nó que está rodando o algoritmo DV detecta uma mudança no custo do enlace, ele atualiza seu vetor de distâncias:
  - No tempo t_0, y detecta a mudança no custo do enlace, atualiza seu vetor de distâncias e informa essa mudança a seus vizinhos;
  - No tempo t_1, z recebe a atualização de y e atualiza sua tabela sua própria tabela. Calcula um novo menor custo para x e envia seu vetor a seus vizinhos;



a.

Roteamento de Vetor de Distância (distance-vector DV)

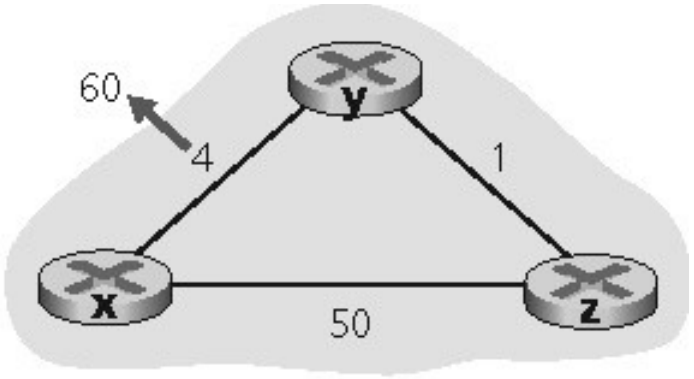
- Continuação:
  - No tempo t_2, y recebe a atualização de z e atualiza sua tabela de distâncias. Os menores custos de y não mudaram, portanto, y não envia nenhuma mensagem a z
  - O algoritmo entra em estado de inatividade:



a.

Roteamento de Vetor de Distância (distance-vector DV)

- O que aconteceria se o custo de um enlace aumentar?



b.

  - Boas notícias viajam rápido, mas notícias viajam devagar;
  - Problema da "contagem ao infinito" (Batata quente entre y e z)

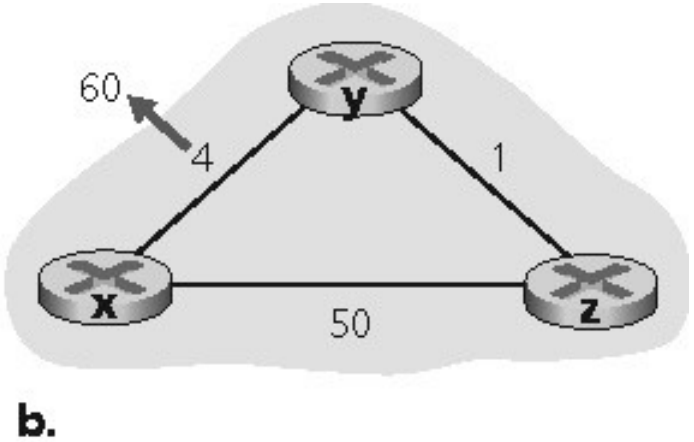
Roteamento de Vetor de Distância (distance-vector DV)

- Antes da mudança do curso do enlace, $D_y(x) = 4$, $D_y(z) = 1$, $D_z(y) = 1$ e $D_z(x) = 5$

  - No tempo t_0, y detecta a mudança no custo do enlace ($D_y(x) = 60$). y calcula seu novo caminho de custo mínimo até x;

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

$$\min\{60 + 0, 1+5\} = 6$$



b.

Roteamento de Vetor de Distância (distance-vector DV)

- Loop de roteamento:
  - Para chegar a x, y faz a rota através de z, que por sua vez faz a rota através de y.
  - Dessa forma, um pacote destinado a x que chegar a y ou a z a partir do momento t_1 vai ricochetear entre um e outro desses dois nós para sempre (ou até que as tabelas de repasse sejam mudadas)

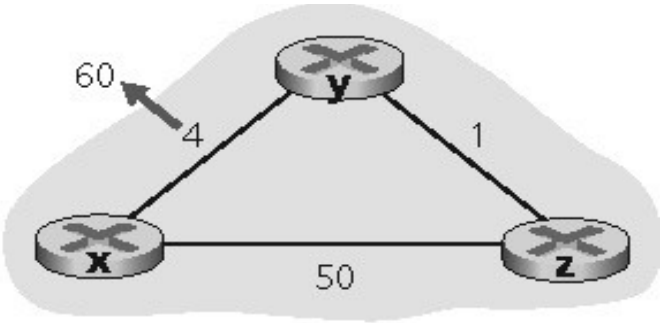
- Tão logo o nó y tenha calculado um novo custo mínimo até x, ele informará a z esse novo vetor de distâncias no tempo t_1;

Roteamento de Vetor de Distância (distance-vector DV)

- Algum tempo depois de t_1, z recebeo novo vetor de distâncias de y, que indica que o custo mínimo de y até x é 6.

- Z por sua vez sabe que pode chegar até y com um custo de 1 e, por conseguinte, calcula um novo menor custo até x:

$$D_z(x) = \min\{ c(z,x) + D_x(x),$$
$$c(y,z) + D_y(x)\}$$
$$\min\{50 + 0, 1+6\} = 7$$



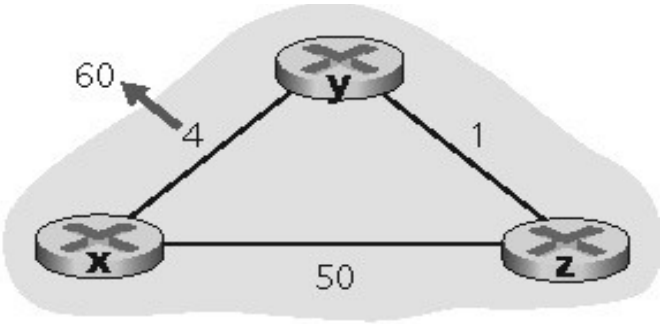
b.

Roteamento de Vetor de Distância (distance-vector DV)

- De maneira semelhante, após receber o novo vetor de distâncias de z, y determina $D_y(x) = 8$ e envia a z seu vetor de distâncias;

- Então z determina $D_z(x) = 9$ e envia a y seu vetor de distâncias e assim por diante:

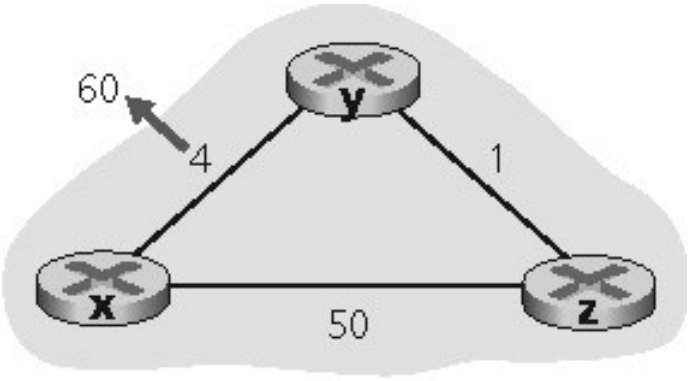
$$D_z(x) = \min\{ c(z,x) + D_x(x),$$
$$c(y,z) + D_y(x)\}$$
$$\min\{50 + 0, 1+8\} = 9$$



b.

Roteamento de Vetor de Distância (distance-vector DV)

- Quantas interações entre y e z serão realizadas?
  - 44 interações



b.

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

Comparação entre os algoritmos LS e DV

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

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

Comparação entre os algoritmos LS e DV

- **Robustez:** o que acontece se um roteador funciona mal?
  - Ls:
    - Nós podem informar custos de **link** incorretos
    - Cada nó calcula sua própria tabela de roteamento
  - Dv:
    - Nó DV pode informar custo de **caminho** incorreto
    - Tabela de cada nó é usada por outros
    - Propagação de erros pela rede

Roteamento Hierárquico

- Nos algoritmos LS e DC, consideramos a rede simplesmente como uma coleção de roteadores interconectados, todos rodando o mesmo algoritmo;

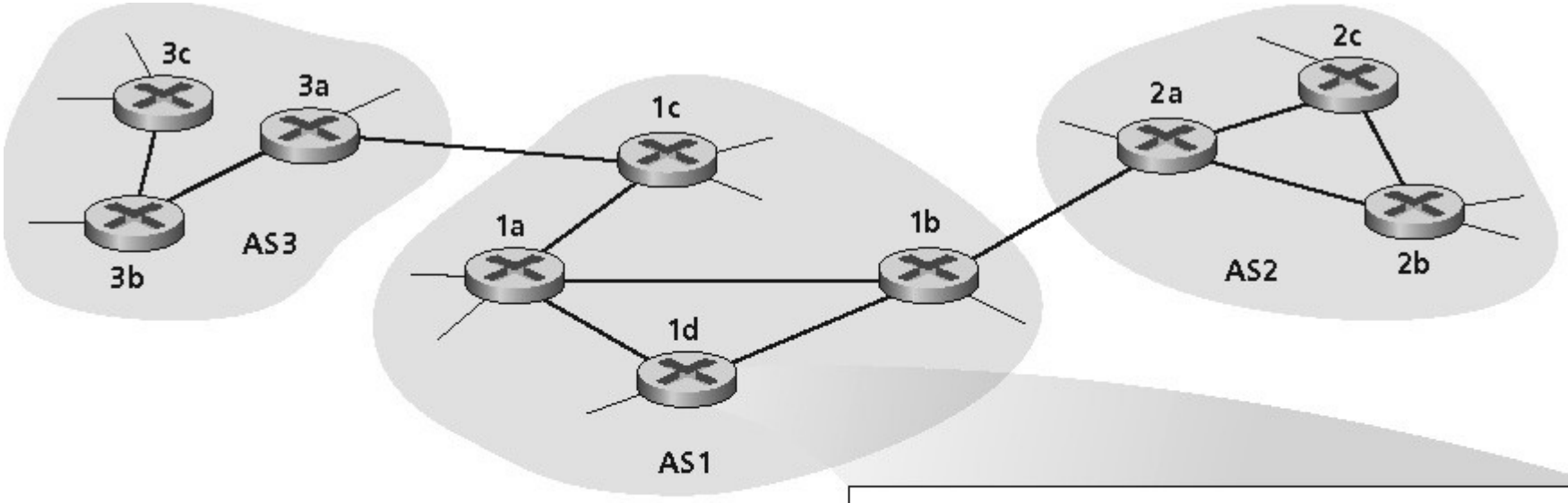
- Na prática, temos:
  - Escalabilidade: aumento no número de roteadores, sobrecarga relativa ao cálculo, ao armazenamento e à comunicação da tabela de roteamento;
  - Autonomia administrativa: empresas desejam controlar seus roteadores como bem entendem;

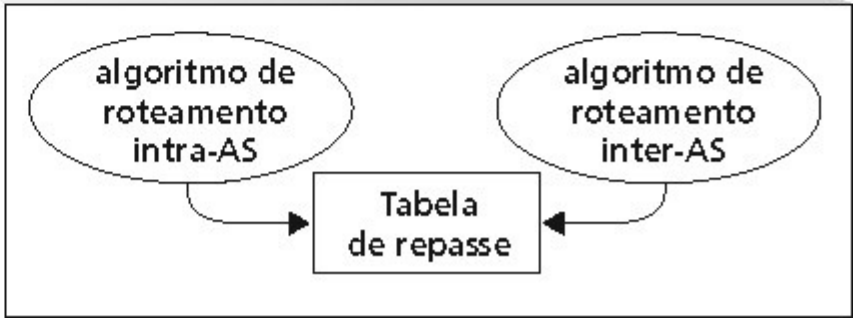
Roteamento Hierárquico

- Agrupamos, portanto, os roteadores em:
  - Sistemas autônomos (autonomous systems – Ass): roteadores sob o mesmo controle administrativo rodando o mesmo algoritmo de roteamento (roteamento intra-sistema autônomo);
  - Roteadores de borda (gateway routers): roteadores responsáveis em conectar os Ass entre si;

Roteamento Hierárquico

- Um exemplo de sistemas autônomos interligados:





Roteamento Hierárquico

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

- AS1 precisa:
  - Aprender quais destinos são alcancáveis através de AS2 e através de AS3.
  - Propagar suas informações de alcance para todos os roteadores em AS1.
  - Tarefa para o roteamento inter-AS routing!

Roteamento Hierárquico

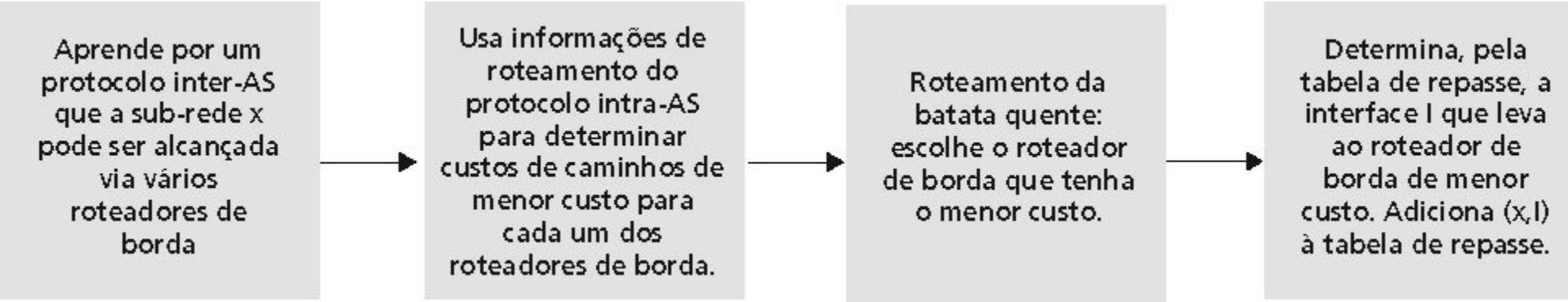
- Suponha que AS1 aprende pelo protocolo inter-AS protocol que a sub-rede **x** é alcancável através de AS3 (gateway 1c) mas não através de AS2
  - O protocolo inter-AS propaga informações de alcance para todos os roteadores internos
  - Baseado nas informações de roteamento intra-AS, o roteador 1d determina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

Roteamento Hierárquico

- Agora suponha que AS1 aprende pelo protocolo inter-AS que a sub-rede **x** é alcançavel através de AS3 *e* através de AS2.
  - Para configurar a tabela de roteamento, o roteador 1d deve determinar por qual gateway ele deve encaminhar os pacotes para o destino x.
  - Isso também é tarefa para o protocolo de roteamento inter-AS.
  - Roteamento de “batata quente”: envia o pacote para o mais próximo de dois roteadores

Roteamento Hierárquico

- Resumindo:



Roteamento na Internet

- Veremos que os protocolos de roteamento da Internet incorporam muitos dos princípios que aprendemos anteriormente (LS, DV e Intra-AS);
  - Roteamento Intra-AS também são conhecidos como Internet Gateway Protocols - IGP

- Protocolos de roteamento intra-AS mais comuns:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (proprietário da Cisco)

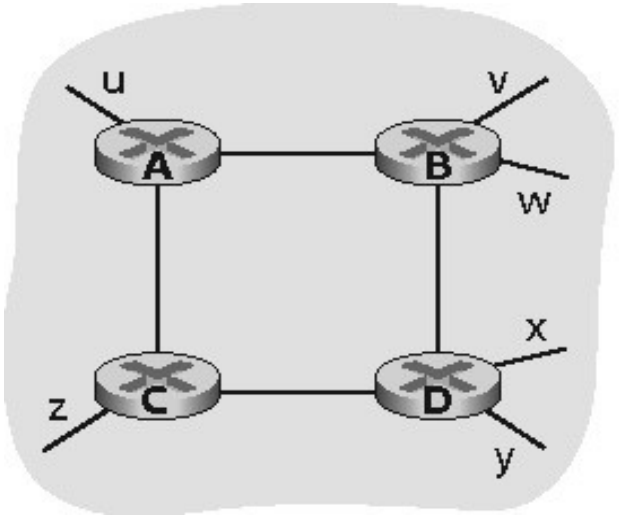
Routing Information Protocol - RIP

- O RIP foi um dos primeiros protocolos de roteamento intra-AS da Internet, e seu uso é ainda amplamente disseminado;
  - Sobretudo, pela inclusão, em 1982, na versão do BSD-UNIX;
  - A versão 1 do RIP está definida no RFC 1058 e a versão 2, compatível com a versão 1, no RFC 1723

- O RIP é um protocolo de vetor de distâncias que funciona de um modo muito parecido com o algoritmo DV;

Routing Information Protocol - RIP

- A versão 1 do RIP usa contagem de saltos como métrica de custo, isto é, cada enlace tem um custo 1;

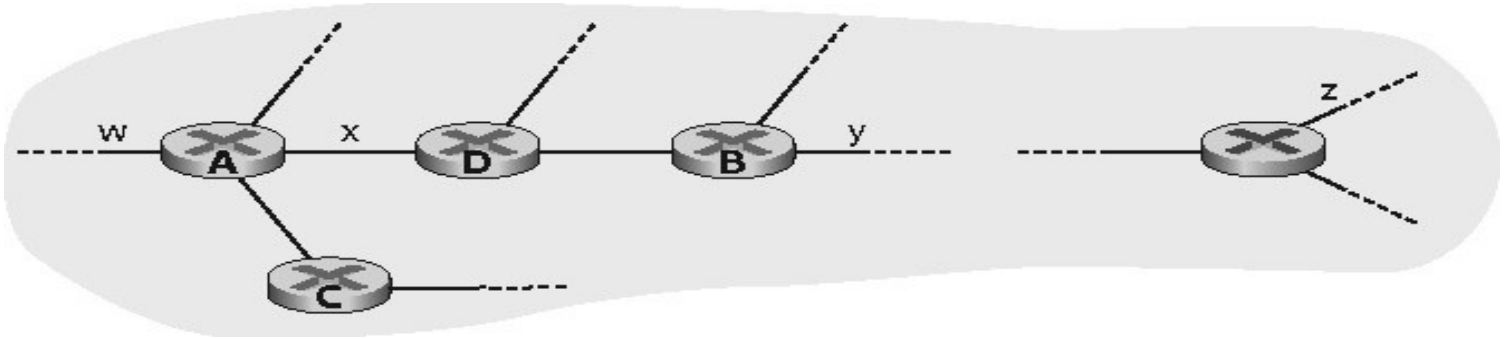


Destino	Saltos
u	1
v	2
w	2
x	3
y	3
z	2

  - O custo máximo de um caminho é limitado a 15
  - Tabelas de roteamento são trocadas entre vizinhos a cada 30 segundos (anúncios RIP), no máximo 25 sub-redes de destino.

Routing Information Protocol - RIP

- Vejamos um exemplo onde temos uma parte de um sistema autônomo e a tabela de roteamento no roteador D:



Sub-rede de destino	Roteador seguinte	Nº de saltos até o destino
w	A	2
y	B	2
z	B	7
x	-	1
. . .	. . .	. . .

Routing Information Protocol - RIP

- Suponha que 30 segundos mais tarde o roteador D receba do roteador A o anúncio mostrado na tabela abaixo:

Sub-rede de destino	Roteador seguinte	Nº de saltos até o destino
z	C	4
w	-	1
x	-	1
. . .	. . .	. . .

  - Dessa forma, o roteador D atualiza sua tabela de roteamento para levar em conta o mais curto dos caminhos mais curtos;
  - Como ficaria a nova tabela de roteamento de D?

Routing Information Protocol - RIP

- Vejamos alguns aspectos da implementação do RIP:
  - Se não há um aviso depois de 180 segundos, o vizinho e o enlace são declarados mortos;
  - Então, novos anúncios são enviados aos vizinhos;
  - Os vizinhos por sua vez devem enviar novos anúncios (se suas tabelas de rotas foram alteradas);
  - Reversão envenenada é usada para previnir loops (distância infinita = 16 saltos);

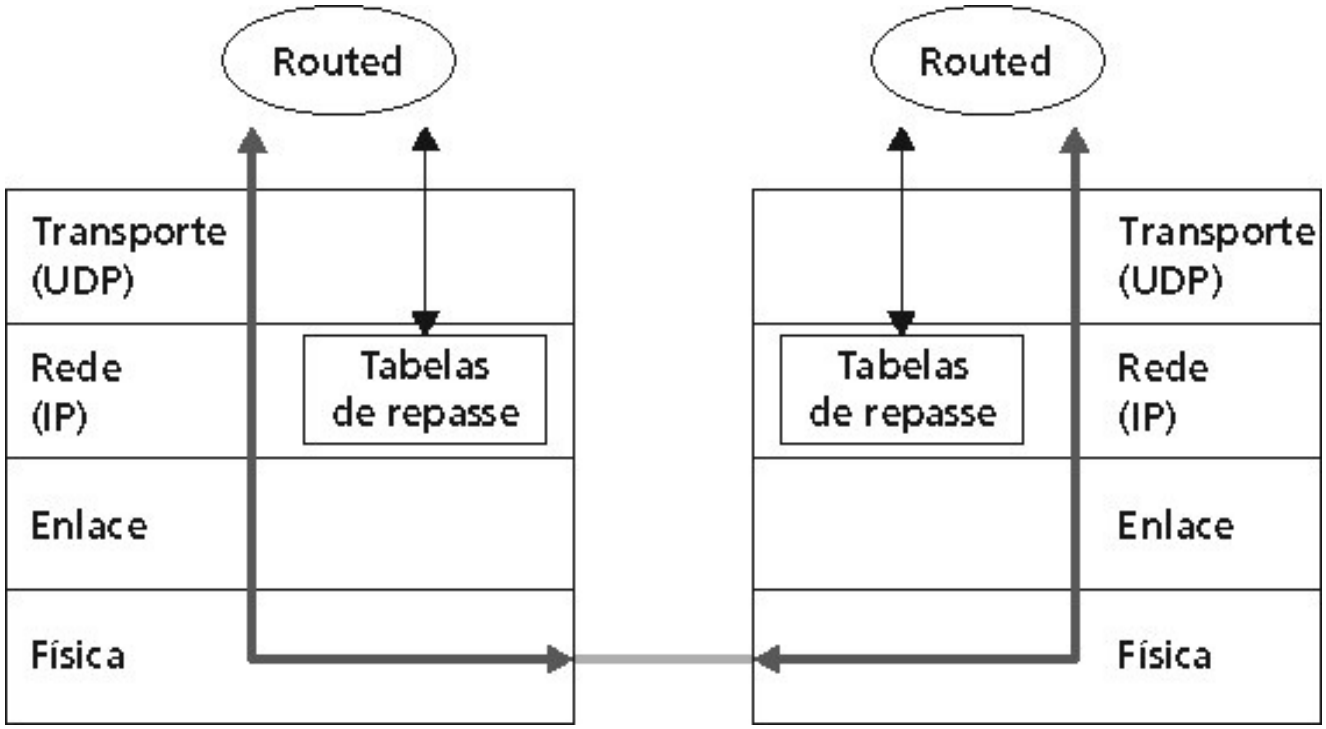
Routing Information Protocol - RIP

- Roteadores enviam mensagens RIP de requisição e de resposta pelo protocolo UDP (Camada de Transporte) usando a porta 520 sobre o protocolo IP (Camada de Rede);

- Um processo denominado *routed* roda o RIP e troca mensagens com processos *routed* dos roteadores vizinhos;
  - Assim, o RIP é um protocolo de camada de aplicação que roda sobre UDP;
  - Processo manipula as tabelas de roteamento dentro do núcleo do UNIX;

Routing Information Protocol - RIP

- A figura abaixo ilustra esquematicamente como o RIP é comumente implementado em sistemas UNIX:



Open Shortest Path First - OSPF

- O OSPF foi concebido como sucessor do RIP e como tal tem uma série de características avançadas:
  - Protocolo de estado de enlace que usa broadcasting;
  - Algoritmo de caminho de menor custo de Dijkstra;

- Os custos dos enlaces são configurados pelo administrador da rede:
  - Custo de enlace em 1 ou
  - Pesos inversamente proporcionais à capacidade do enlace (menor tráfego em banda baixa);

Open Shortest Path First - OSPF

- A versão mais recente do OSPF, versão 2, está definida no RFC 2178. Nela é especificado que:
  - Um roteador transmite informações do estado de enlace sempre que houver uma mudança no estado de um enlace;
  - O roteador transmite o estado do enlace periodicamente (pelo menos a cada 30 minutos), mesmo que o estado não tenha mudado;
  - Anúncios OSPF são carregados diretamente por IP;
  - O protocolo OSPF verifica se os enlaces estão operacionais (via uma mensagem HELLO).

Open Shortest Path First - OSPF

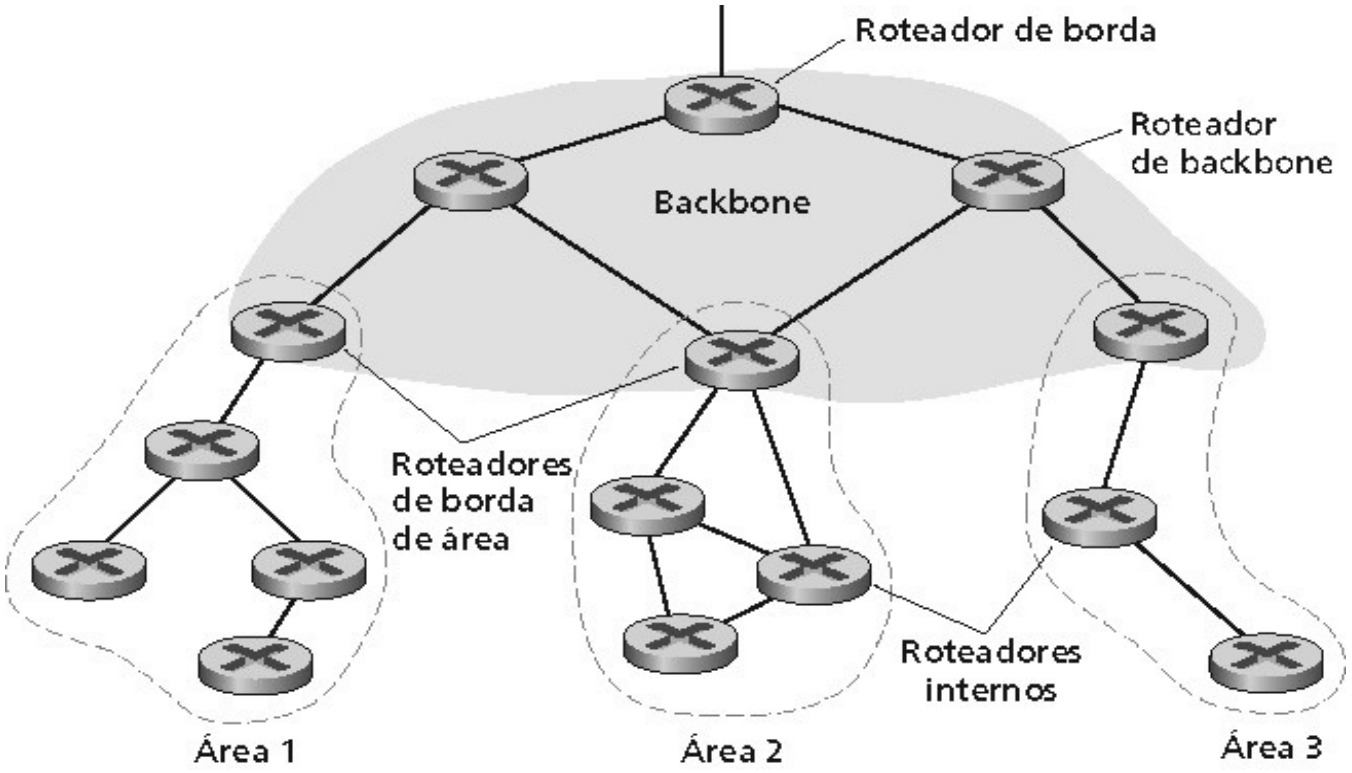
- Alguns avanços incorporados ao OSPF são:
  - Segurança: Todas as mensagens são autenticadas para previnir intrusões maliciosas;
  - Múltiplos caminhos de mesmo custo são permitidos (o RIP só permite um caminho);
  - Para cada link, múltiplas métricas de custo, por exemplo, custo de enlace por satélite definido baixo para tráfego de "melhor esforço" e alto para serviços de tempo real;
  - Suporte intergrado para roteamento *unicast e multicast* [RFC 1584];
  - Suporte para hierarquia dentro de um único domínio de roteadores (OSPF hierárquico);

Open Shortest Path First - OSPF

- Um sistema autônomo OSPF pode ser configurado em áreas:
  - Cada área roda seu próprio algoritmo de roteamento de estado de enlace OSPFe transmite seu estado de enlace a todos os outros roteadores daquela área;
  - Assim, os detalhes internos de uma área permanecem invisíveis para todos os roteadores externos a ela;
  - Dentro de cado área, um ou mais roteadores de borda são responsáveis pelo roteamento de pacotes fora da área;
  - Exatamente uma área OSPF no AS é configurada para ser área de backbone;

Open Shortest Path First - OSPF

- Vejamos um diagrama de rede OSPF hierarquicamente estruturada:



Open Shortest Path First - OSPF

- Podemos identificar quatro tipos de roteadores OSPF:
  - Roteadores Internos: estão em áreas que não são de backbone e realizam apenas roteamento intra-AS;
  - Roteadores de borda de área: pertencem tanto a uma área quanto ao backbone;
  - Roteadores de backbone: realizam o roteamento dento do backbone, mas não são roteadores de borda de área;
  - Roteadores de borda: traca informações de roteamento com roteadores pertencentes a outros sistemas autônomos. Esse roteador poderia, por exemplo, usar BGP para realizar roteamento inter-AS;

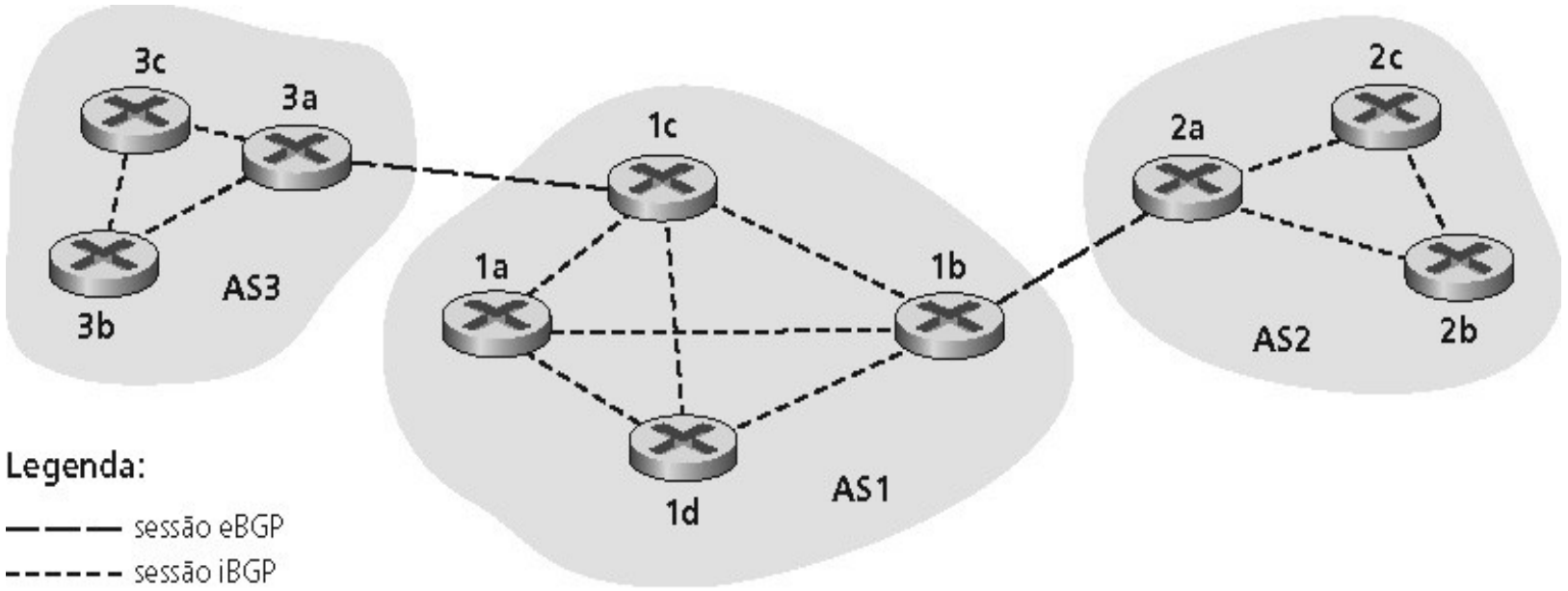
Border Gateway Protocol - BGP

- A versão 4 do Protocolo de Roteamento de Borda (BGP), especificada na RFC 1771, é o padrão, de fato, para roteamento entre sistemas atônomos na Internet de hoje;

- O BGP permite que cada sub-rede anuncie sua existência ao restante da Internet e como chegar até lá:
  - Extremamente complexo;
  - Protocolo que agrea TUDO;

Border Gateway Protocol - BGP

- No BGP, pares de roteadores trocam informações de roteamento por conexões TCP semipermanentes usando a porta 179:



Border Gateway Protocol - BGP

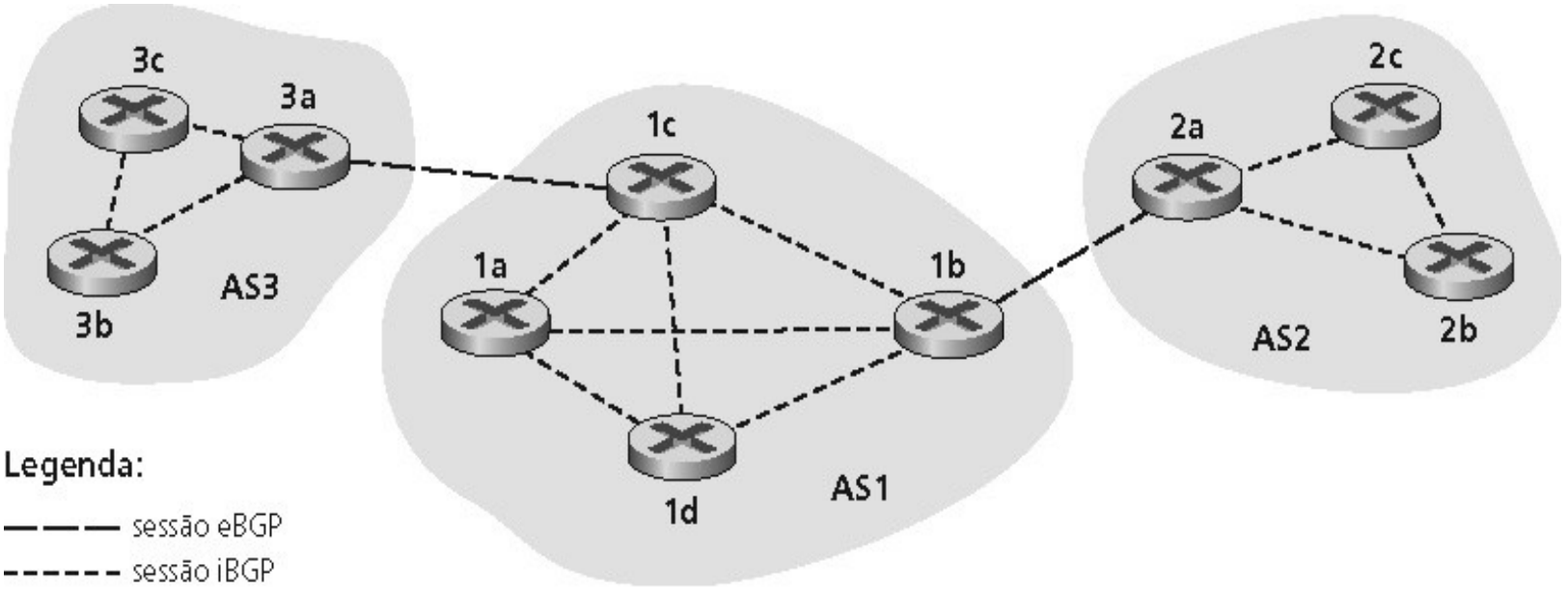
- Os dois roteadores nas extremidades de cada conexão TCP são denominados pares BGP, e a conexão TCP é denominada de sessão BGP;
- Uma sessão BGP externa (eBGP) é uma sessão que abranja dois ASs;
- Uma sessão BGP interna (iBGP) é uma sessão entre dois roteadores no mesmo AS;

Border Gateway Protocol - BGP

- Em BGP, os destinos não são hospedeiros, mas prefixos ciderizados, sendo que cada prefixo representa uma sub-rede ou um conjunto de sub-redes.

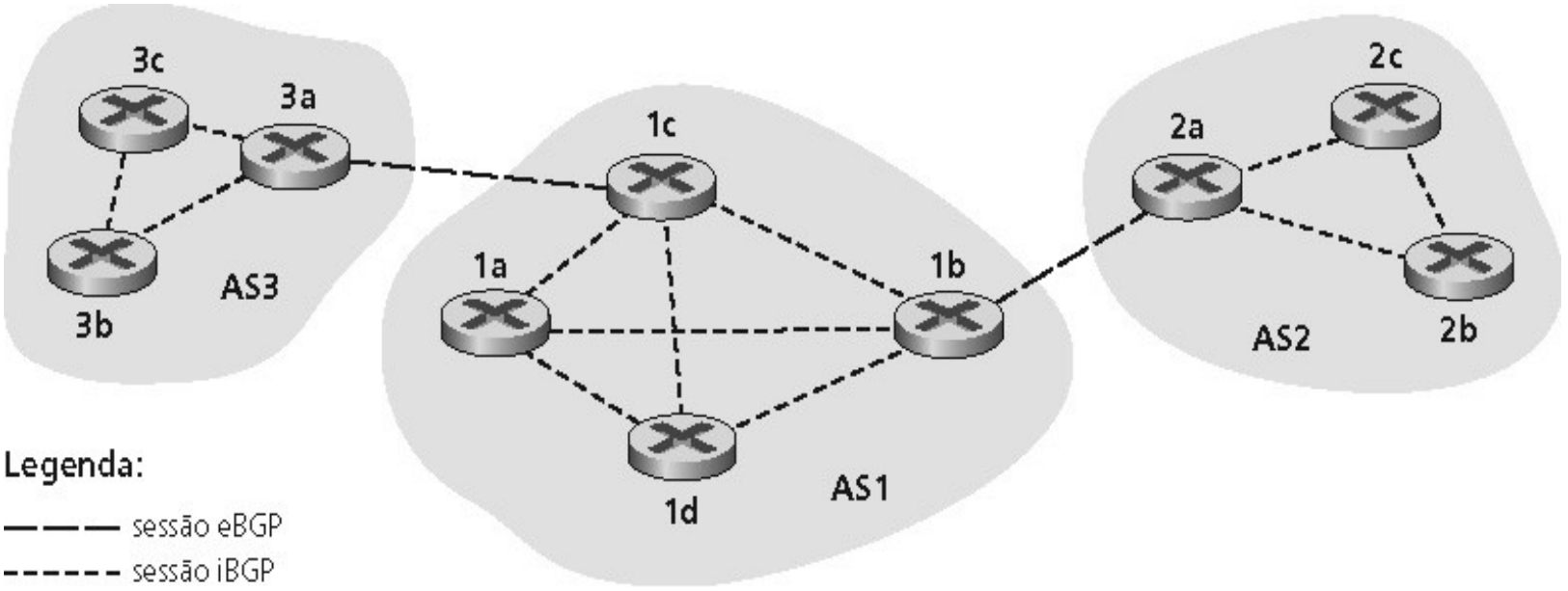
- Quando AS2 comunica um prefixo ao AS1, AS2 está prometendo que encaminhará todos os datagramas destinados a esse prefixo em direção ao prefixo;
  - AS2 pode agregar prefixos em seu comunicado;

Border Gateway Protocol - BGP



- Em cada sessão eBGP entre 3a e 1c, AS3 envia informações de alcance de prefixo para AS1;
- 1c pode então usar iBGP para distribuir essa nova informação de alcance de prefixo para todos os roteadores em AS1

Border Gateway Protocol - BGP



- 1b pode recomunicar essa nova informação para AS2 por meio de sessão eBGP 1b-para2a;
- Quando um roteador aprende um novo prefixo, ele cria uma entrada para o prefixo em sua tabela de roteamento.

Border Gateway Protocol - BGP

- Quando se comunica um prefixo, o comunicado inclui os atributos BGP:
  - Prefixo + atributos = “rota”

- Vejamos dois atributos importantes:
  - AS-PATH: Contém os Ass pelos quais o comunicado para o prefixo passou AS 67 AS 17
  - NEXT-HOP: Indica o roteador específico interno ao AS para o AS do próximo salto (next-hop). (Pode haver múltiplos links do AS atual para o AS do próximo salto.)

Border Gateway Protocol - BGP

- Quando um roteador gateway recebe um comunicado de rota, ele usa política de importação para aceitar/rejeitar;

- Um roteador pode aprender mais do que uma rota para o mesmo prefixo, para isso ele deve selecionar uma rota, vejamos a seleção:
  - Atributo de valor preferência local: decisão de política
  - AS-PATH (caminho) mais curto
  - Roteador do NEXT-HOP (próximo salto) mais próximo: roteamento da “batata quente”
  - Critérios adicionais

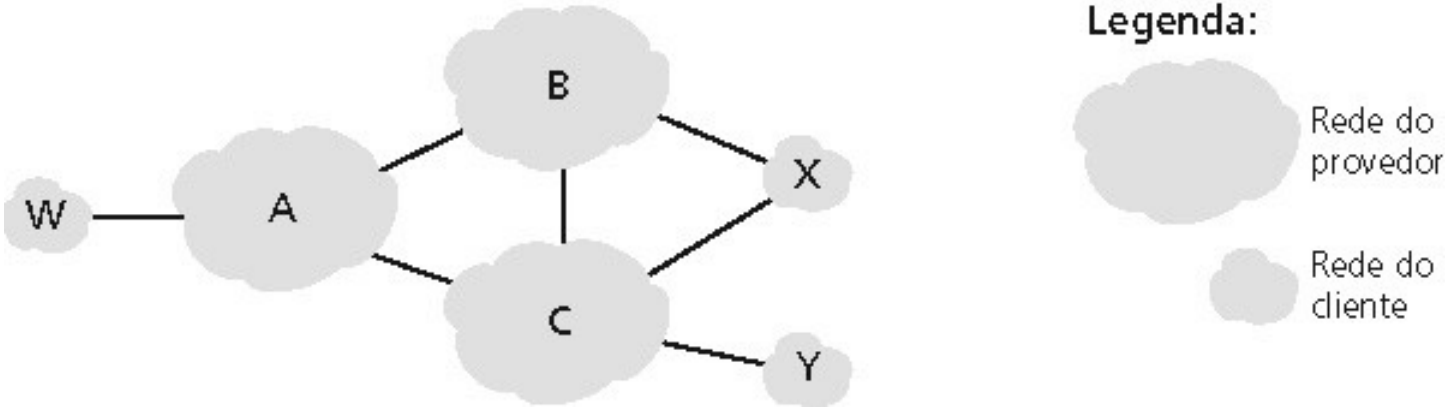
Border Gateway Protocol - BGP

- As mensagens BGP são trocadas usando o TCP:

- Vejamos os principais tipos de mensagens BGP:
  - OPEN: abre conexão TCP para o peer e autentica o transmissor
  - UPDATE: comunica novo caminho (ou retira um antigo)
  - KEEPALIVE mantém a conexão ativa na ausência de atualizações (updates); também ACKs OPEN request
  - NOTIFICATION: reporta erros em mensagens anteriores; também usado para fechar a conexão

Border Gateway Protocol - BGP

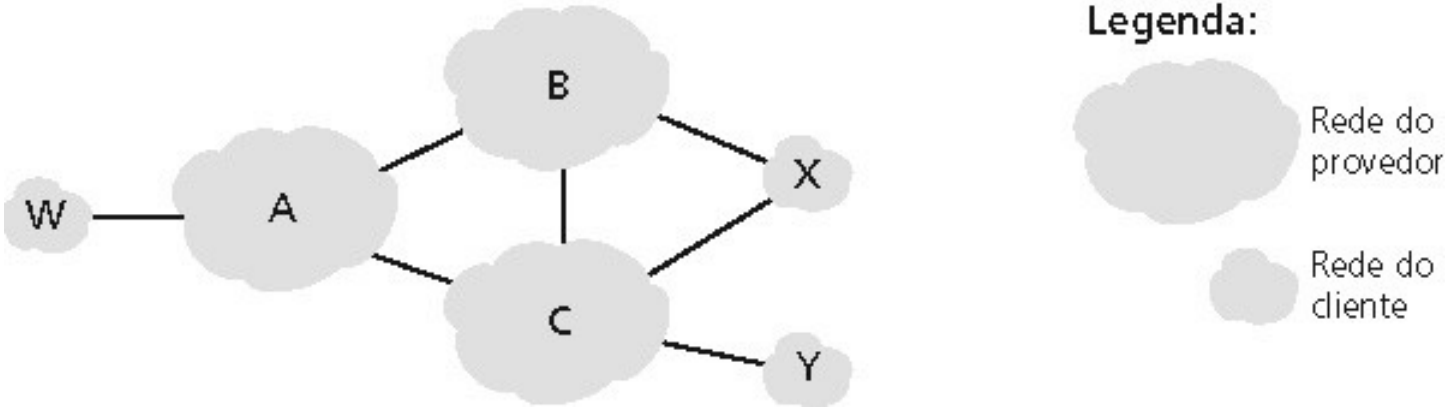
- Política de Roteamento do BGP:



  - A, B, C são redes do provedor
  - X, W, Y são clientes (das redes do provedor)
  - X é dual-homed: anexados a duas redes
  - X não quer rotear de B via X para C
  - ... então X não comunicará ao B uma rota para C

Border Gateway Protocol - BGP

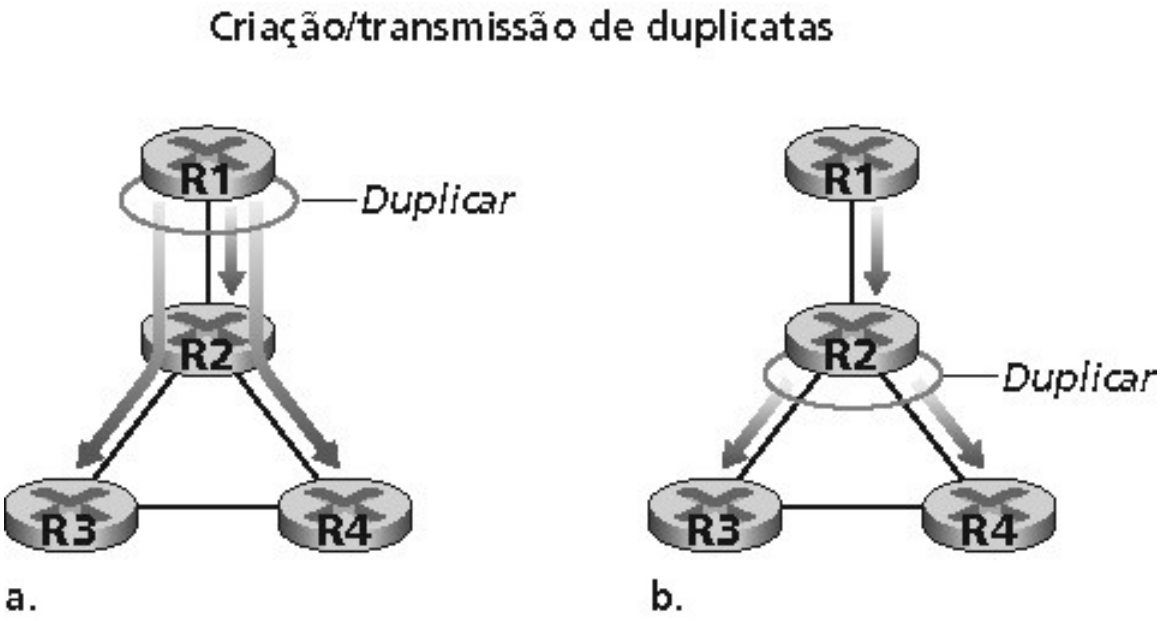
- Política de Roteamento do BGP:



  - A comunica ao B o caminho AW
  - B comunica ao X o caminho BAW
  - B deveria comunicar ao C o caminho BAW?
  - De jeito nenhum! B não obtém nenhum “rendimento” em rotear CBAW pois nem W nem C são seus clientes
  - B quer forçar C a rotear para W via A
  - B quer rotear **somente** de/para seus clientes!

Roteamento de Broadcast

- Talvez o modo mais direto de conseguir comunicação broadcast é o nó remetente enviar uma cópia separada do pacote para cada destino (inundação não controlada), vejamos:



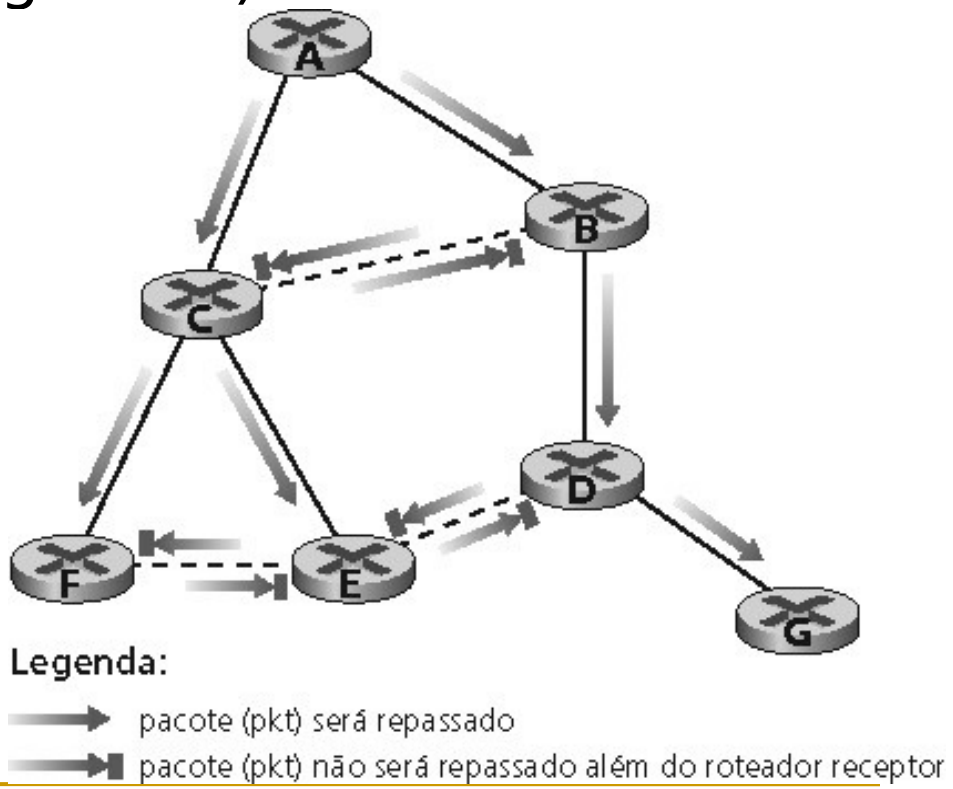
Roteamento de Broadcast

- Embora esse esquema seja simples e elegantes, tem uma falha fatal:
  - Se o grafo tiver ciclos, então uma ou mais cópias de cada pacote de broadcast permanecerão em ciclo indefinido;

- A chave para evitar uma tempestade de broadcast é um nó escolher sensatamente quando repassa um pacote:
  - Inundação controlada: cada nó mantém uma lista de endereços de fonte e números de sequência para cada pacote broadcast que já recebeu, duplicou e repassou;

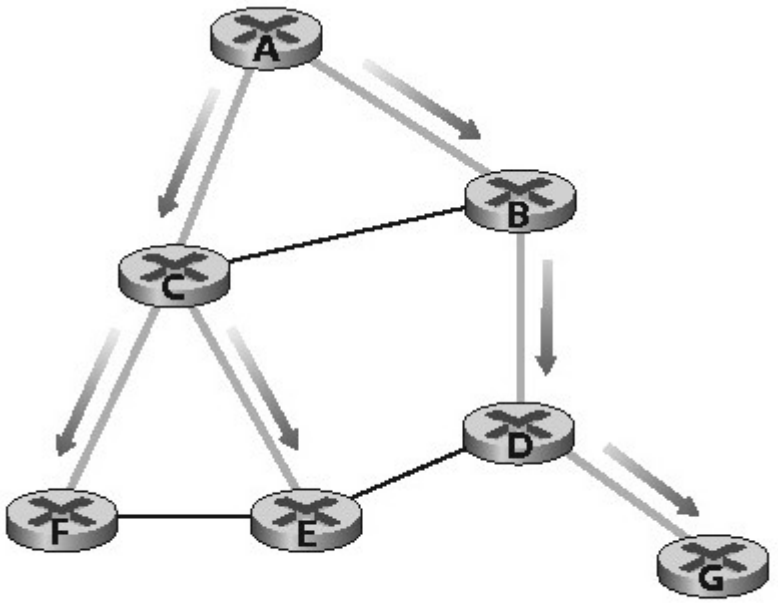
Roteamento de Broadcast

- Um segunda abordagem da inundação controlada é conhecida como repasse pelo caminho inverso (Reverse Path Forwarding – RPF):
  - Quando um roteador recebe um pacote broadcast com um dado endereço de fonte, ele transmite o pacote para todos os seus enlaces (exceto para aquele do qual o pacote foi recebido);



Roteamento de Broadcast

- Percebe-se que o RPF não impede pacote redundantes. Por exemplo, na figura abaixo os nós B, C, D e F recebem um ou dois pacotes redundantes:
  - Exemplo de uma spanning tree – uma árvore que contém todos os nós, sem exceção, em um grafo.
  - Se cada enlace tiver um custo associado, spanning tree mínima é a árvore buscada;



a. Broadcast iniciado em A

Roteamento de Broadcast

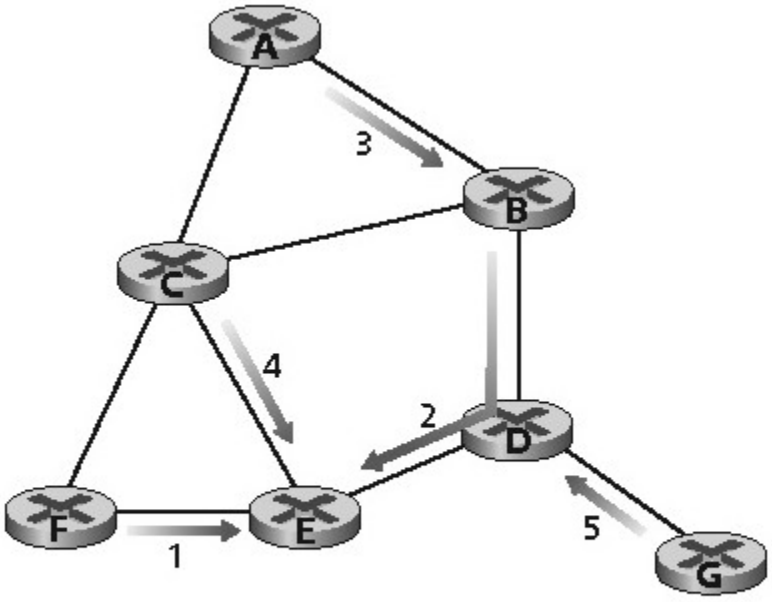
- Uma abordagem é mostrar aos roteadores qual de seus vizinhos no grafo são vizinhos à spanning tree;

- Existem diversos algoritmos distribuídos de spanning tree. Usaremos a abordagem Kurose que relata o nó central:
  - Os nós transmitem um unicast mensagens de adesão à árvore endereçadas ao nó central
  - Vejamos um exemplo, em que o nó E seja selecionado como centro da árvore;

Roteamento de Broadcast

- Suponha que o nó F primeirmente se junte à árvore e repasse uma mensagem de adessão à árvore a E;
- O único enlace EF se torna a spanning tree inicial

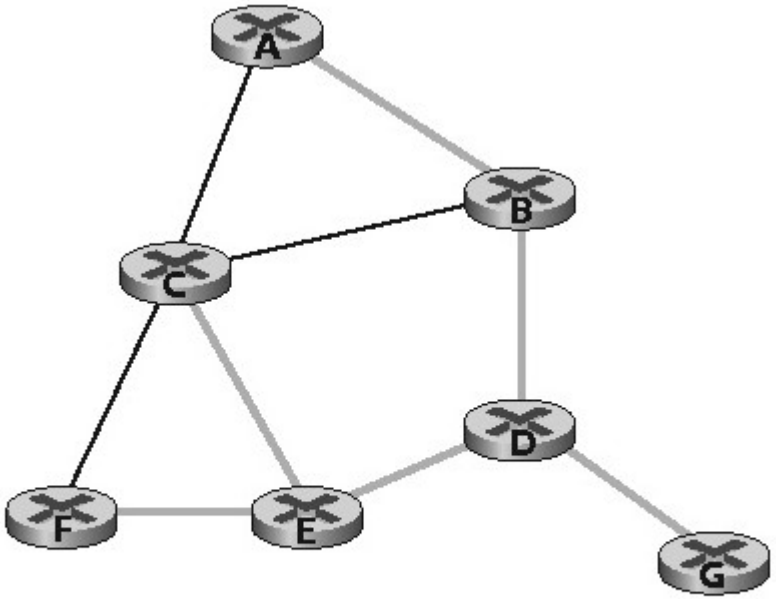
- O nó B então se junta à spanning tree enviando a E sua mensagem de adesão à árvore;
- Supondo que a rota do caminho unicast de E para B seja por D. Nesse caso, a ocorre um exerto do caminho BDE à spanning tree.



a. Construção da spanning tree passo a passo

Roteamento de Broadcast

- Em seguida o nó A se junta passando por B;
- Logo após, o nó C se junta à spanning tree
- Finalmente o G envia sua mensagem de adesão à árvore a E;



b. Spanning tree construída

Bibliografia

- KUROSE, J.F e ROSS, K.W.: *Computer Networking third edition a top-down approach featuring the Internet,* 3 ed, São Paulo: Pearson Addison Wesley, 2006.

- TANENBAUM, A.S.: *Redes de Computadores,* Elsevier, Rio de Janeiro: 2003.