Instituto de Informática - UFRGS

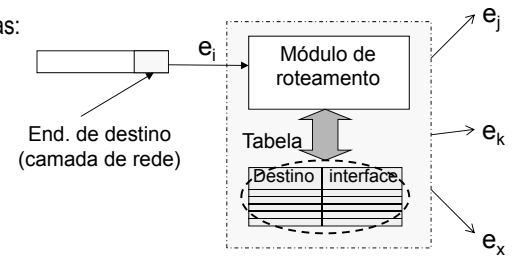
Redes de Computadores

Algoritmos de roteamento

Aula 19

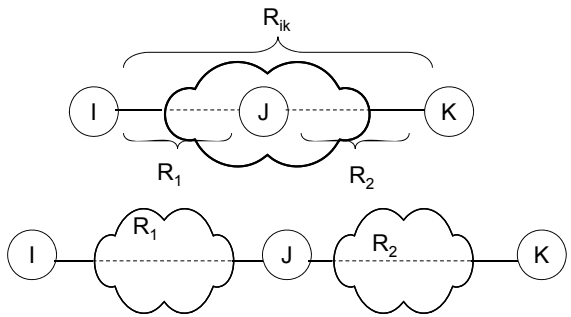
Introdução

- Uma rede é modelada através de um grafo direcionado onde os nós representam roteadores e as arestas ligações entre estes
  - Cada aresta é caracterizada por um custo
  - O custo de um caminho é a soma dos custos das arestas deste caminho
- Problema: encontrar o melhor caminho entre "A" e "B"
  - Algoritmos de roteamento
  - Duas estratégias básicas:
    - Vetor de distâncias
    - Estado de enlace



Princípio da otimização de uma rota

- *"Se o roteador J está no caminho ótimo do Roteador I para K, então a rota ótima de I para J e de J para K também estão contidos nesta mesma rota"*

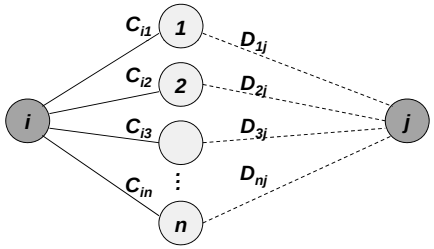


Vetor de distância

- Algoritmo distribuído
- Local
- Sistema autônomo é visto como um grafo
  - Roteador é um nó
  - Rede é uma aresta conectando dois nós
- Algoritmo Bellman-Ford

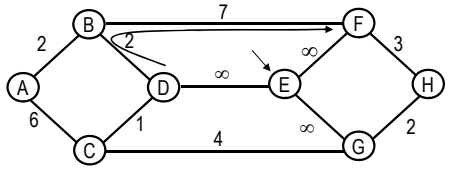
Princípio do algoritmo Bellman-Ford

Se os vizinhos de um nó *i* conhecem um caminho até um nó *j*, a menor distância entre o nó *i* e *j* e obtido encontrando o menor valor resultante da soma da distância de *i* até um vizinho e deste até o nó *j*.



$D_i(j) = \min_v \{c(i,v) + D_v(j)\}$ para todo vizinho *v* (*v*=1,2,3...n)

Exemplo: roteamento por vetor de distância (1)



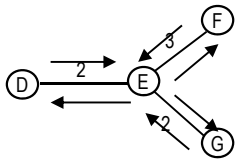
Roteador E está desligado

$D_D = \{ D_D(A); D_D(B); D_D(C); D_D(D); D_D(E); D_D(F); D_D(G); D_D(H)\} = \{4;2;1;0;\infty;9;5;7\}$

$D_F = \{ D_F(A); D_F(B); D_F(C); D_F(D); D_F(E); D_F(F); D_F(G); D_F(H)\} = \{9,7,9,9, \infty,0,5,4\}$

$D_G = \{ D_G(A); D_G(B); D_G(C); D_G(D); D_G(E); D_G(F); D_G(G); D_G(H)\} = \{9,7,4,5, \infty,5,0,2\}$

Exemplo: roteamento por vetor de distância (2)



E só conhece seus vizinhos imediatos

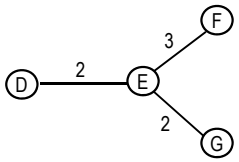
$D_E = \{ D_E(D); D_E(E); D_E(F); D_E(G)\} = \{ 2; 0; 3; 2\}$ ⟶ D, F, G

$D_D = \{ D_D(A); D_D(B); D_D(C); D_D(d); D_D(E); D_D(F); D_D(G); D_D(H)\} = \{4;2;1;0;2;9;5;7\}$ ⟶ E

$D_F = \{ D_F(A); D_F(B); D_F(C); D_F(D); D_F(E); D_F(F); D_F(G); D_F(H)\} = \{9,7,9,9,3,0,5,4\}$ ⟶ E

$D_G = \{ D_G(A); D_G(B); D_G(C); D_G(d); D_G(E); D_G(F); D_G(G); D_G(H)\} = \{9,7,4,5,2,5,0,2\}$ ⟶ E

Exemplo: roteamento por vetor de distância (2)



E só conhece seus vizinhos imediatos

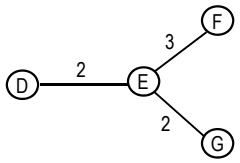
$D_E = \{ D_E(D); D_E(E); D_E(F); D_E(G)\} = \{ 2; 0; 3; 2\}$ ⟶ D, F, G

$D_D = \{ D_D(A); D_D(B); D_D(C); D_D(d); D_D(E); D_D(F); D_D(G); D_D(H)\} = \{4;2;1;0;2;9;5;7\}$ ⟶ E

$D_F = \{ D_F(A); D_F(B); D_F(C); D_F(D); D_F(E); D_F(F); D_F(G); D_F(H)\} = \{9,7,9,9,3,0,5,4\}$ ⟶ E

$D_G = \{ D_G(A); D_G(B); D_G(C); D_G(d); D_G(E); D_G(F); D_G(G); D_G(H)\} = \{9,7,4,5,2,5,0,2\}$ ⟶ E

Exemplo: roteamento por vetor de distância (2)



	D_D	D_F	D_G
A	4	9	9
B	2	7	7
C	1	9	4
D	0	9	5
E	2	3	2
F	9	0	5
G	5	5	0
H	7	4	2

Tabela de roteamento E

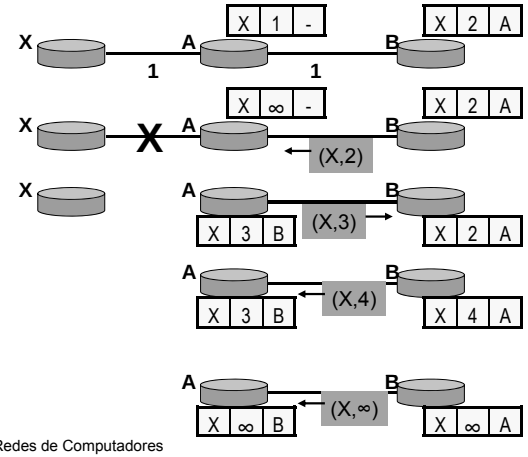
Destino	Next-hop	Custo
A	D	6
B	D	4
C	D	3
D	D	2
E	-	0
F	F	3
G	G	2
H	G	4

Algoritmo de roteamento por vetor de distância

- Adaptações ao algoritmo Bellman-Ford
  - O custo é o número de saltos (*hops*) então custo da aresta é 1
    - Menor caminho = menor número de intermediários
  - Roteadores atuam de forma assíncrona
    - Avaliam rotas sempre que recebem informações dos vizinhos
    - Sistema distribuído
- Informação = vetor de distância
  - {Rede de destino, Custo a partir do vizinho *v*}
- Cada roteador mantém uma tabela de roteamento
  - Uma entrada por rota: {Rede de destino, custo e próximo salto}
- Envio periódico e em alteração

Contagem para o infinito

- Problema de convergência no roteamento por vetor de distância

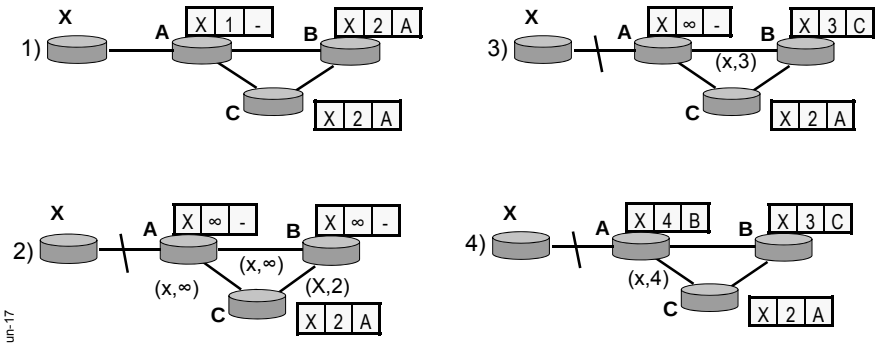


Não daria problema **se** *A* enviasse o vetor de distância **antes** de *B*, pois declararia que seu custo até *X* seria infinito.

Soluções para contagem ao infinito

- Horizonte dividido
  - Envio parcial do vetor de distância a cada interface
  - Não envia informação na direção (interface) em que a rota foi aprendida
    - ex: se B sabe que o caminho para X foi aprendido via A, não precisa informar a rota para X para o A
- Inversão envenenada
  - Responde a todas direções (interfaces) porém, para aquela em que "aprendeu" a informação divulga a rota com distância "infinita"
    - ex.: se B aprendeu de A o caminho para X, então declara para A que o custo de B para ser é infinito

Problema: Topologias com caminhos alternativos



Origem do problema:
Quando o nó *B* informa a *A* que tem caminho para um nó X, *A* não tem como saber se ele próprio está no caminho "visto" por B.

⇨ Falta visão global

Soluções para contagem ao infinito (*cont*.)

- Definir "um valor" para infinito
  - Ao atingir o valor de infinito (ex. 16) o destino é declarado inatingível

Estado de enlace

- Algoritmo distribuído
- Global
- Todos nós possuem a topologia completa da rede, mas tem visão diferente das rotas
  - Ex.: o caminho de A e B são diferentes para atingir X
- Usa o algoritmo de Dijkstra para construir a tabela de roteamento

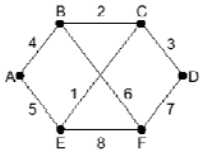
Construindo tabela de roteamento

- Quatro etapas
  - Construção do estado de cada enlace (*Link State Packet* – LSP)
  - Disseminação do LSP para demais roteadores (*flooding* – inundação)
  - Formação da topologia da rede e determinação do menor caminho (*Dijkstra*)
  - Cálculo da tabela de rotas

Link State Packet (LSP)

- Composto por:
  - Identificação do nó
  - Lista dos enlaces diretos do nó com custos
  - Número de sequência
    - Distinguir novos LSPs dos antigos ou duplicados
  - Idade
    - Tempo de validade da informação do LSP em um roteador (time to live)

LSP são gerados periodicamente ou sempre que houver mudanças



Link State Packets

A		B		C		D		E		F	
Seq.		Seq.		Seq.		Seq.		Seq.		Seq.	
Age		Age		Age		Age		Age		Age	
B	4	A	4	B	2	C	3	A	5	B	6
E	5	C	2	D	3	F	7	C	1	D	7
		F	6	E	1			F	8	E	8

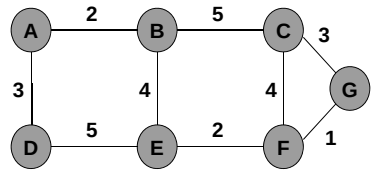
Divulgação de informações (inundação)

- Enviar informações (LSPs) através de *flooding*
- Problema é que nós podem ter visões diferentes da topologia
  - Os primeiros a receber as informações já podem usá-las
- Melhorias:
  - Número de sequência: saber se um LSP é novo ou não
    - Se novo, é considerado e reenviado para as saídas, senão, é descartado
  - Idade: dupla função
    - Eliminar pacotes de laços de roteamento
    - Dizer por quanto tempo aquela informação deve ser armazenada no nó
      - e.g.: decrementar esse valor uma vez por segundo, ao chegar em zero, "limpa" a entrada referente ao nó

Determinação do menor caminho

- Após receber LSPs cada nó constrói a topologia
  - Observação: pode não estar completa e ser diferente da visão dos demais roteadores
- Algoritmo de *Dijkstra*
  - Seleciona nó como raiz da árvore de caminhos
  - Busca o menor caminho para cada destino

Construção da tabela de rota



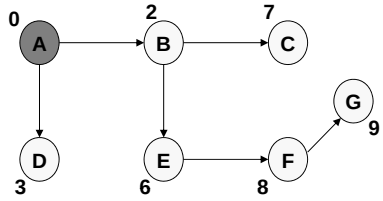


Tabela roteamento A

Destino	Custo	Próximo
A	0	-
B	2	-
C	7	B
D	3	-
E	6	B
F	8	B
G	9	B

Comparação

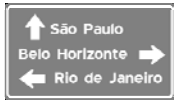
Vetor de distância	Estado de enlace
Descentralizado; local.	Descentralizado ou centralizado; global.
Cada nó envia informações para seus vizinhos imediatos.	Cada nó envia informações para todos os outros nós.
A informação enviada é o custo (estimado) para todos os nós .	A informação enviada é o custo do nó para cada um de seus vizinhos imediatos.
A informação é enviada periodicamente e enquanto houver modificação nos custos.	A informação é enviada sempre que uma ocorrer alterações na rede.
Um nó determina o *next-hop* usando o algoritmo distribuído (Bellman-Ford) sobre os custos recebidos.	Um nó constrói a topologia completa da rede (segundo sua visão) e usa um algoritmo qualquer de caminho mínimo entre dois pontos.

Problemas com algoritmos de roteamento

- O vetor de distância e estado de enlace apresentam os seguintes problemas
  - Escalabilidade
  - Questão de autonomia administrativa
    - Evitar tráfego de outras redes em uma rede interna de uma organização
    - Não usar algoritmos de roteamento impostos por outros
- Solução: roteamento hierárquico e sistemas autônomos

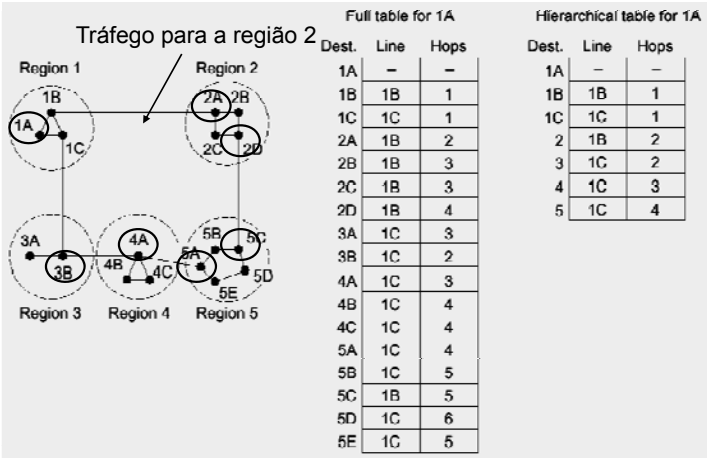
Roteamento hierárquico

- Roteadores são organizados de forma hierárquica em regiões
  - Um roteador conhece detalhes internos apenas da sua região e nenhum detalhe sobre a região vizinha
    - Conhece apenas o roteador responsável pela região vizinha
- Princípio usado quotidianamente
  - Exemplo: ensina como chegar a SP, BH e RJ sem se preocupar com a zona, bairro, rua e casa



- As regiões são demarcadas como “sistemas autônomos”

Exemplo de roteamento hierárquico e SA



Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

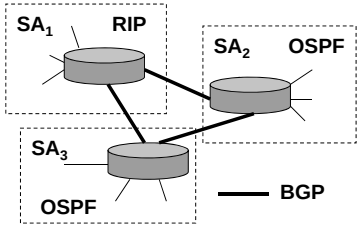
Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Sistemas autônomos (SA)

- Composto por:
  - Um conjunto de roteadores sob o mesmo controle administrativo
    - Executam um mesmo protocolo de roteamento (baseado em vetor de distância OU estado de enlace)
  - Roteador de borda (*gateway routers*)
    - Responsável por ligar o SA a outros Sas
    - Atua no sentido de melhorar a escalabilidade
- Dois protocolos de roteamento
  - Interno ao sistema autônomo (IRP – *Internal Routing Protocol*)
    - Independência administrativa
  - Externo ao sistema autônomo (ERP – *External Routing Protocol*)
    - Obedecer regras externas entre SA

Roteamento hierárquico na Internet

- *Interior Routing Protocol*
  - RIP (*Routing Information Protocol*) → Vetor de distância
  - OSPF (*Open Shortest Path First*) → Estado de enlace
- *Exterior Routing Protocol*
  - BGP (*Border Gateway Protocol*)



- Protocolos *multicast*
  - IGMP, MOSPF, DVMPR, CBT, PIM,...

Maiores detalhes e sequência na disciplina de protocolos

Leituras complementares

- Stallings, W. *Data and Computer Communications* (6th edition), Prentice Hall 1999.
  - Capítulo 10, seção 10.2 e anexo 10.A
  - Capítulo 16, seção 16.1
- Tanenbaum, A. *Redes de Computadores* (4ª edição), Campus2003.
  - Capítulo 5, seções 5.2.2, 5.2.4, 5.2.5 e 5.2.6
- Carissimi, A.; Rochol, J; Granville, L.Z; *Redes de Computadores*. Série Livros Didáticos. Bookman 2009.
  - Capítulo 5, seções 5.2.3, 5.2.4 e 5.3