Instituto de Informática - UFRGS

Redes de Computadores

Algoritmos de roteamento

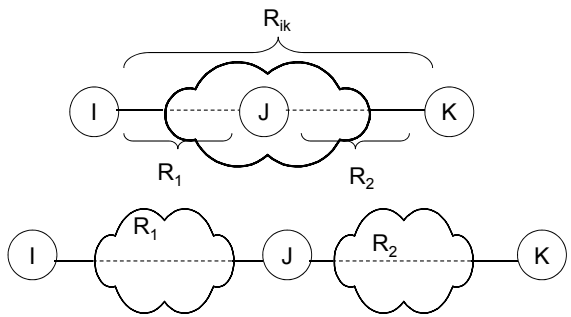
Aula 20

Introdução

- Um 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
- Algoritmo de roteamento deve encontrar o menor caminho
  - São derivados da teoria dos grafos
- Algoritmos
  - Estáticos: caminho mais curto
  - Dinâmicos: vetor de distância (*distance vector*) , estado de enlace (*link state*)

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"*



Algoritmo estático: caminho mais curto

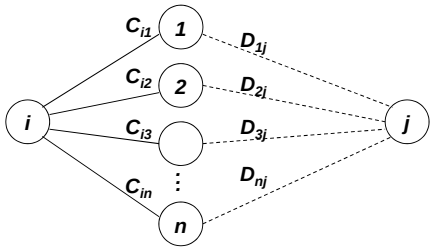
- Mais simples
- Considera uma topologia da rede (fixa)
  - Matriz de conexões, nó *i* é conectado ao nó *j* a um custo *c*
  - O custo associado a um arco pode ter várias interpretações
    - e.g.: número de saltos (hops), menor atraso, largura de banda, etc...
- Consiste basicamente em encontrar uma seqüência de nós a serem percorridos para "ir" de um nó *i* a um nó *j*
  - Algoritmo global, isto é, se tem o conhecimento completo do grafo
  - Calculado de forma centralizada e distribuída para os roteadores
- Vários algoritmos da teoria de grafos
  - Mais conhecido é o Algoritmo de Dijkstra (1959)

Vetor de distância

- Algoritmo distribuído
- Local
- Sistema autônomo é visto como um grafo
  - Roteador é um nó
  - Rede é uma aresta conectando dois nós
- Objetivo: encontrar o menor caminho entre dois nós
  - Algoritmo Bellman-Ford (Ford-Fulkerson)

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*.



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

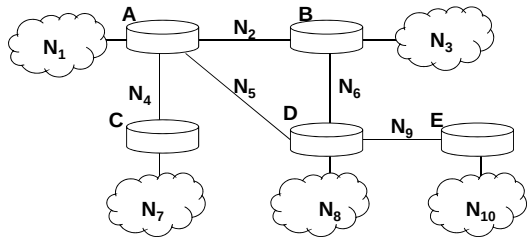
Exemplo de tabela de roteamento

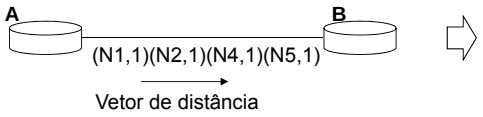
A

Destino	Custo	Próximo
N1	1	-
N2	1	-
N4	1	-
N5	1	-

B

Destino	Custo	Próximo
N2	1	-
N3	1	-
N6	1	-

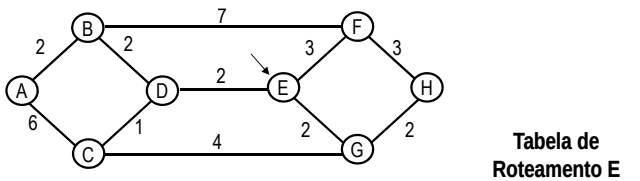




B

Destino	Custo	Próximo
N2	1	-
N3	1	-
N6	1	-
N1	2	A
N4	2	A
N5	2	A

Exemplo: roteamento por vetor de distância



$C^E(v)$

D	2
F	3
G	2

Custo do enlace

$C^D(d)$

A	4
B	2
C	1
D	0
E	2
F	9
G	5
H	7

$C^F(d)$

A	9
B	7
C	9
D	9
E	3
F	0
G	5
H	4

$C^G(d)$

A	9
B	7
C	4
D	5
E	2
F	5
G	0
H	2

Vetores de distância vizinhos imediatos

$D^E(d,v)$

	D	F	G
A	6	12	11
B	4	10	9
C	3	12	6
D	2	12	7
E	-	-	-
F	11	3	7
G	7	8	2
H	9	7	4

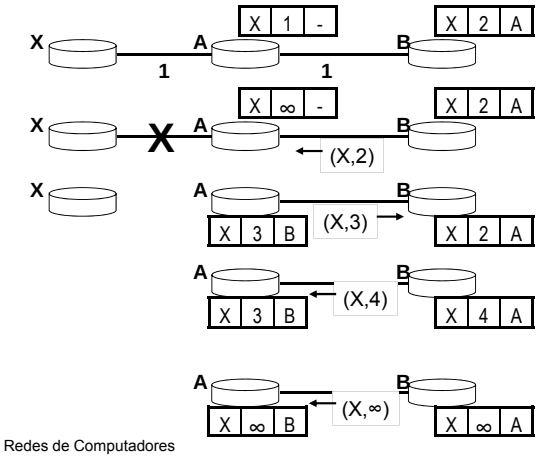
$C^E(D)+C^D(d)$ $C^E(F)+C^F(d)$ $C^E(G)+C^G(d)$

Tabela de Roteamento E

Dst	custo, via
A	6,D
B	4,D
C	3,D
D	2,D
E	-,-
F	3,F
G	2,G
H	4,G

Contagem para o infinito

- Problema de convergência no roteamento por vetor de distância



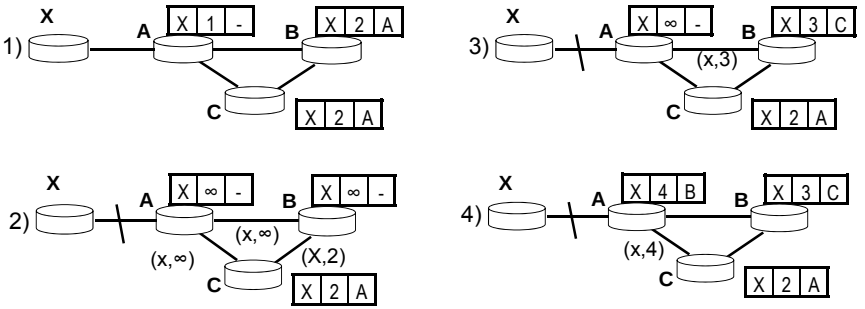
Não daria problema **se** *A* enviasse o vetor de distância **antes** de *B*, pois declararia que seu custo até *X* seria infinito.

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

Soluções para contagem ao infinito (*cont*.)

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



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

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

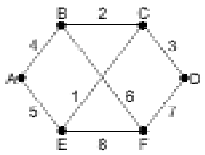
Construindo tabela de roteamento

- Quatro etapas
  - Construção do estado de cada enlace (*Link State Packet* – LPC)
  - Disseminação do LSP para demais roteadores (*flooding* – inundaçãp)
  - 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
    - Previnir que um LSP antigo permaneça muito tempo

Gerados periodicamente ou sempre que houver mudanças



Link State Packets

A	
Seq.	
Age	
B	4
E	5

B	
Seq.	
Age	
A	4
C	2
F	6

C	
Seq.	
Age	
B	2
D	3
E	1

D	
Seq.	
Age	
C	3
F	7

E	
Seq.	
Age	
A	5
C	1
F	8

F	
Seq.	
Age	
B	6
D	7
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

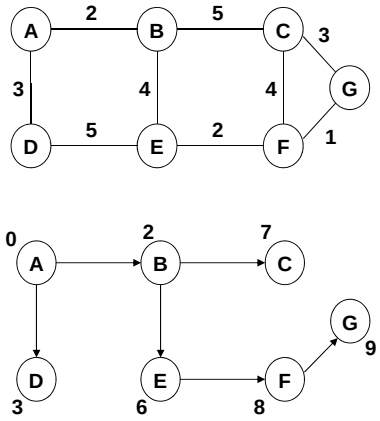


Tabela roteamento A

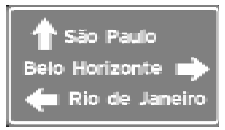
Destino	Custo	Próximo
A	0	-
B	2	-
C	7	B
D	3	-
E	6	B
F	8	B
G	9	B

Comparação

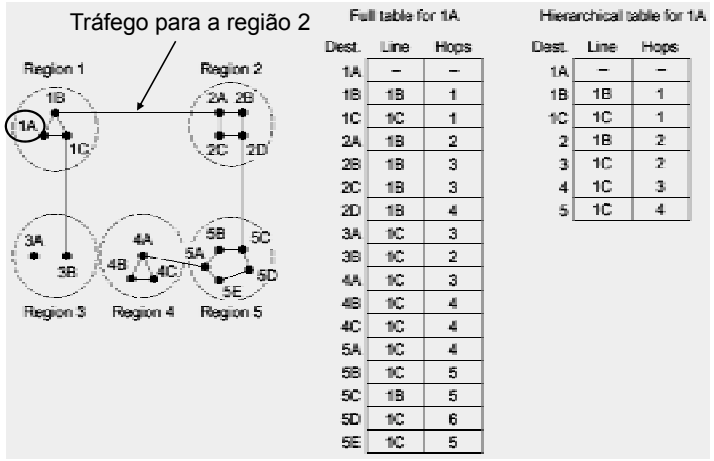
Vetor de distância	Estado de enlace
Cada nó envia informações para seus vizinhos imediatos.	Cada nó envia informações para todos os outros nós.
A informação enviado é o custo (estimado) para todos os nós .	A informação enviada é o custo do nó para cada um de seus vizinhos imediatos.
A informação é enviada periodicamente.	A informação é enviada sempre que uma troca ocorrer na rede.
Um nó determina o *next-hop* usando o algoritmo distribuído (e.g. 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.

Roteamento hierárquico

- Problemas com vetor de distância e estado de enlace
  - 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
- 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



Exemplo de roteamento hierárquico



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

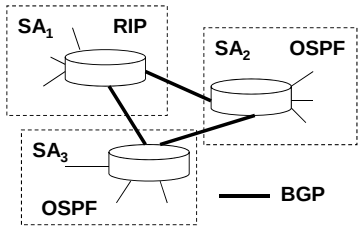
Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Roteamento hierárquico na Internet

- Roteamento na Internet tem dois níveis de hierarquia
  - Interdomínio: executam um protocolo de roteamento de acordo com as necessidades de uma organização (*Interior Routing Protocol* – IRP)
    - Sistemas autônomos (AS – Autonomous Systems)
  - Intradomínio: executam um mesmo protocolo de roteamento (*Exterior Routing Protocol* – ERP)

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