

Algoritmos de Roteamento

Pedro Paulo V. Campos
Tarcísio Eduardo M. Crocomo

10 de novembro de 2010

Sumário

1 Introdução

2 Fundamentação

3 Algoritmos Estáticos

4 Algoritmos Dinâmicos

5 Conclusão

Sumário

1 Introdução

2 Fundamentação

3 Algoritmos Estáticos

4 Algoritmos Dinâmicos

5 Conclusão

Motivação

```
traceroute to nhk.co.jp (61.58.37.103)
 1 192.168.1.254 0.150 ms
 2 roteador.inf.ufsc.br 1.835 ms
 3 npd252e1-lgb-npd254rs.bb.ufsc.br 5.233 ms
 4 popsc-lg-ufsc-(...)-r250.bb.pop-sc.rnp.br 5.939 ms
 5 rnp-2g-194-251-v40-r251.bb.pop-sc.rnp.br 6.613 ms
 6 so-1-0-0-r1-rs.bkb.rnp.br 11.776 ms
 7 so-0-0-0-r1-df.bkb.rnp.br 51.419 ms
 8 so-0-2-0-r1-sp.bkb.rnp.br 66.531 ms
 (...)
16 xe-0-1-0.r21.miamfl02.us.bb.gin.ntt.net 177.438 ms
 (...)
20 as-1.r21.osakjp01.jp.bb.gin.ntt.net 396.762 ms
21 ae-2.r23.tokyjp01.jp.bb.gin.ntt.net 374.946 ms
22 129.250.3.75 407.060 ms
23 xe-1-1-0.a05.taiptw01.tw.ra.gin.ntt.net 430.482 ms
 (...)
27 nhk-grp.jp 408.878 ms
```

O Roteador



Figura: DI-604: 100 Mbps, \$60

O Roteador



Figura: Cisco CRS-3: 322 Tbps, \$60.000

Desafios

Correção Fornecer não apenas uma rota válida mas a melhor

Escalabilidade Tamanho de rede variável. Algoritmos eficientes

Estabilidade Rápida adaptação a mudanças ou problemas na rede

Robustez Funcionar por anos sem necessitar reinicialização

Justiça Visar eficiência global sem gerar *starvation*

Desafios

Correção Fornecer não apenas uma rota válida mas a melhor

Escalabilidade Tamanho de rede variável. Algoritmos eficientes

Estabilidade Rápida adaptação a mudanças ou problemas na rede

Robustez Funcionar por anos sem necessitar reinicialização

Justiça Visar eficiência global sem gerar *starvation*

Desafios

Correção Fornecer não apenas uma rota válida mas a melhor

Escalabilidade Tamanho de rede variável. Algoritmos eficientes

Estabilidade Rápida adaptação a mudanças ou problemas na rede

Robustez Funcionar por anos sem necessitar reinicialização

Justiça Visar eficiência global sem gerar *starvation*

Desafios

Correção Fornecer não apenas uma rota válida mas a melhor

Escalabilidade Tamanho de rede variável. Algoritmos eficientes

Estabilidade Rápida adaptação a mudanças ou problemas na rede

Robustez Funcionar por anos sem necessitar reinicialização

Justiça Visar eficiência global sem gerar *starvation*

Desafios

Correção Fornecer não apenas uma rota válida mas a melhor

Escalabilidade Tamanho de rede variável. Algoritmos eficientes

Estabilidade Rápida adaptação a mudanças ou problemas na rede

Robustez Funcionar por anos sem necessitar reinicialização

Justiça Visar eficiência global sem gerar *starvation*

Sumário

1 Introdução

2 Fundamentação

3 Algoritmos Estáticos

4 Algoritmos Dinâmicos

5 Conclusão

Modelos de Troca de Dados

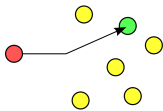


Figura: Unicast

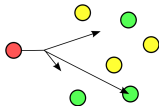


Figura: Anycast

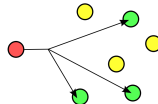


Figura: Multicast

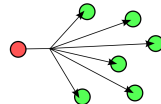


Figura: Broadcast

Modelos de Troca de Dados

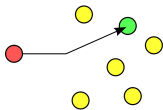


Figura: Unicast

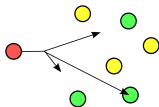


Figura: Anycast

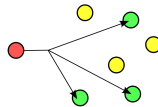


Figura: Multicast

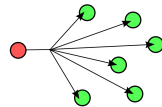


Figura: Broadcast

Modelos de Troca de Dados

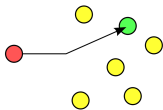


Figura: Unicast

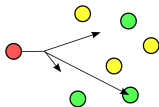


Figura: Anycast

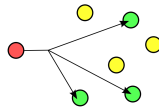


Figura: Multicast

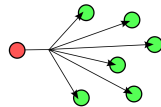


Figura: Broadcast

Modelos de Troca de Dados

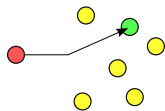


Figura: Unicast

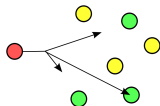


Figura: Anycast

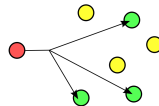


Figura: Multicast

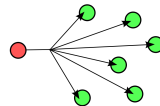


Figura: Broadcast

Modelos de Troca de Dados

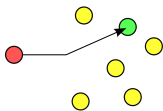


Figura: Unicast

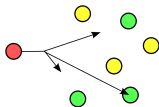


Figura: Anycast

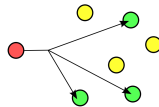


Figura: Multicast

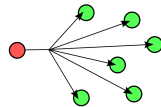


Figura: Broadcast

Métricas para Classificação de Rotas

- Número de hops
- Largura de banda
- Custo (monetário)
- Latência

Métricas para Classificação de Rotas

- Número de hops
- Largura de banda
- Custo (monetário)
- Latência

Métricas para Classificação de Rotas

- Número de hops
- Largura de banda
- Custo (monetário)
- Latência

Métricas para Classificação de Rotas

- Número de hops
- Largura de banda
- Custo (monetário)
- Latência

Sistemas Autônomos

- Como tornar escalável e administrável um conjunto de $\sim 2^{31}$ de computadores interconectados?
- Solução: Agrupar em um SA redes operadas por um ou mais operadores que apresentam uma única política clara de roteamento.
- Exemplo: AS11242 - POP-SC - Responsável por 73728 IPs

Sistemas Autônomos

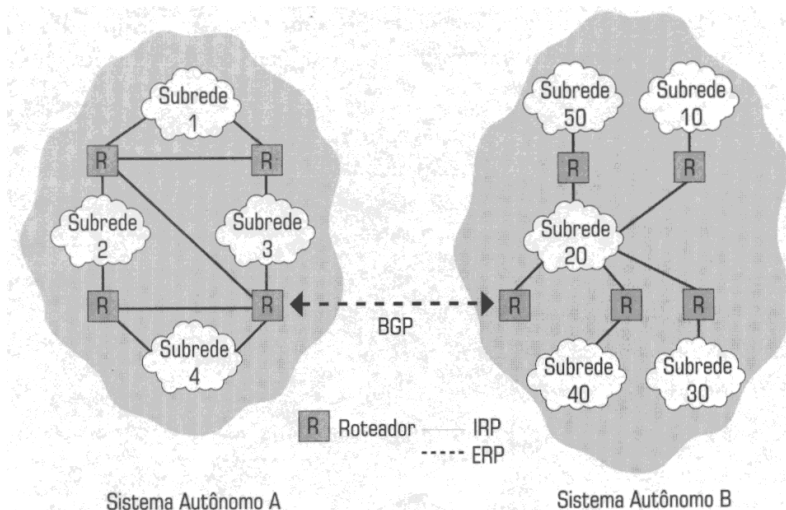
- Como tornar escalável e administrável um conjunto de $\sim 2^{31}$ de computadores interconectados?
- Solução: Agrupar em um SA redes operadas por um ou mais operadores que apresentam uma única política clara de roteamento.
- Exemplo: AS11242 - POP-SC - Responsável por 73728 IPs

Sistemas Autônomos

- Como tornar escalável e administrável um conjunto de $\sim 2^{31}$ de computadores interconectados?
- Solução: Agrupar em um SA redes operadas por um ou mais operadores que apresentam uma única política clara de roteamento.
- Exemplo: AS11242 - POP-SC - Responsável por 73728 IPs

Classificação de Protocolos de Roteamento

Quanto à Vizinhança



Classificação de Protocolos de Roteamento

Quanto à Vizinhança

Externos

$G(V,A)$

$V = \{v \mid v \text{ é um sistema autônomo}\}$

$A = \{(v_1, v_2, m) \mid v_1, v_2 \in V, \text{ há uma ligação direta entre } v_1 \text{ e } v_2 \text{ com um custo } m\}$

Internos

$G(V,A)$

$V = \{v \mid v \text{ é nodo da rede de um sistema autônomo}\}$

$A = \{(v_1, v_2, m) \mid v_1, v_2 \in V, \text{ há uma ligação direta entre } v_1 \text{ e } v_2 \text{ com um custo } m\}$

Classificação de Protocolos de Roteamento

Quanto à Vizinhança

Externos

 $G(V,A)$ $V = \{v \mid v \text{ é um sistema autônomo}\}$ $A = \{(v_1, v_2, m) \mid v_1, v_2 \in V, \text{ há uma ligação direta entre } v_1 \text{ e } v_2 \text{ com um custo } m\}$

Internos

 $G(V,A)$ $V = \{v \mid v \text{ é nodo da rede de um sistema autônomo}\}$ $A = \{(v_1, v_2, m) \mid v_1, v_2 \in V, \text{ há uma ligação direta entre } v_1 \text{ e } v_2 \text{ com um custo } m\}$

Classificação de Protocolos de Roteamento

Quanto à Escolha de Rotas

■ Estáticos (Não adaptativos)

- Menor caminho
- *Flooding*
- Baseado em Fluxo (*Flow-based*)

■ Dinâmicos (Adaptativos)

- Menor caminho
- Baseado em Estado (Link State)
- Baseado em Fluxo

Classificação de Protocolos de Roteamento

Quanto à Escolha de Rotas

- Estáticos (Não adaptativos)
 - Menor caminho
 - *Flooding*
 - Baseado em Fluxo (*Flow-based*)
 - Dinâmicos (Adaptativos)
 - Baseado em Estado
 - Baseado em Distância
 - Baseado em Potência
 - Baseado em Estado e Distância
 - Baseado em Estado e Potência
 - Baseado em Estado, Distância e Potência

Classificação de Protocolos de Roteamento

Quanto à Escolha de Rotas

- Estáticos (Não adaptativos)
 - Menor caminho
 - *Flooding*
 - Baseado em Fluxo (*Flow-based*)
- Dinâmicos (Adaptativos)

Classificação de Protocolos de Roteamento

Quanto à Escolha de Rotas

- Estáticos (Não adaptativos)
 - Menor caminho
 - *Flooding*
 - Baseado em Fluxo (*Flow-based*)
- Dinâmicos (Adaptativos)
 - Vetor distância
 - Estado de link (Link State)
 - Baseado em fluxo

Classificação de Protocolos de Roteamento

Quanto à Escolha de Rotas

- Estáticos (Não adaptativos)
 - Menor caminho
 - *Flooding*
 - Baseado em Fluxo (*Flow-based*)
- Dinâmicos (Adaptativos)
 - Vetor distância
 - Estado do *link* (*Link State*)
 - Hierárquico

Classificação de Protocolos de Roteamento

Quanto à Escolha de Rotas

- Estáticos (Não adaptativos)
 - Menor caminho
 - *Flooding*
 - Baseado em Fluxo (*Flow-based*)
- Dinâmicos (Adaptativos)
 - Vetor distância
 - Estado do *link* (*Link State*)
 - Hierárquico

Classificação de Protocolos de Roteamento

Quanto à Escolha de Rotas

- Estáticos (Não adaptativos)
 - Menor caminho
 - *Flooding*
 - Baseado em Fluxo (*Flow-based*)
- Dinâmicos (Adaptativos)
 - Vetor distância
 - Estado do *link* (*Link State*)
 - Hierárquico

Classificação de Protocolos de Roteamento

Quanto à Escolha de Rotas

- Estáticos (Não adaptativos)
 - Menor caminho
 - *Flooding*
 - Baseado em Fluxo (*Flow-based*)
- Dinâmicos (Adaptativos)
 - Vetor distância
 - Estado do *link* (*Link State*)
 - Hierárquico

Rotas Ótimas

- É possível criar uma descrição das rotas ótimas sem levar em conta a topologia da rede?
- Como medir a qualidade de um algoritmo de roteamento?

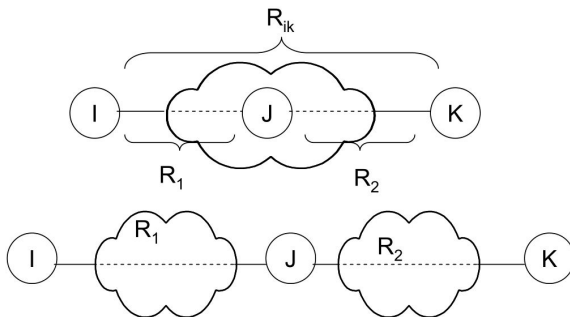
Rotas Ótimas

- É possível criar uma descrição das rotas ótimas sem levar em conta a topologia da rede?
- Como medir a qualidade de um algoritmo de roteamento?

Princípio de Otimização

Teorema

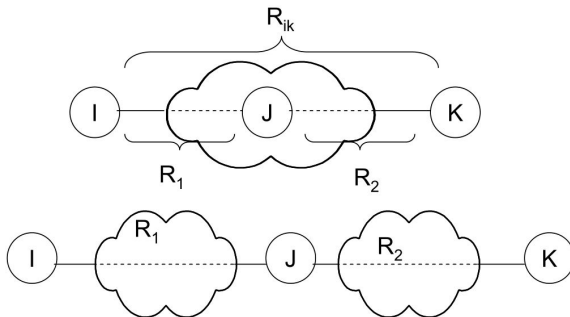
Se um roteador J estiver no caminho ótimo entre os roteadores I e K , o caminho ótimo de J a K também estará na mesma rota.



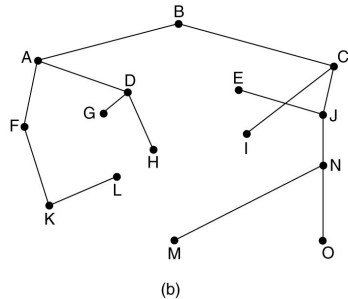
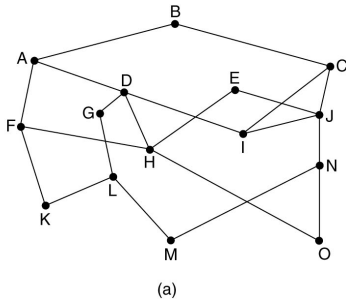
Princípio de Otimização

Prova (por contradição)

Se houvesse uma rota melhor que a enunciada entre J e K , ela poderia ser concatenada a R_1 para criar uma rota melhor entre I e K , contradizendo a afirmação que a rota R_{ik} é ótima. \square



Árvore de Escoamento



Sumário

1 Introdução

2 Fundamentação

3 Algoritmos Estáticos

4 Algoritmos Dinâmicos

5 Conclusão

Menor Caminho

- Um dos algoritmos mais simples
- A partir do modelo de grafos de uma rede interna gera uma sequência de nodos a serem percorridos para um pacote sair da origem e chegar ao destino.
- Algoritmo global, conhecimento completo do grafo
- Calculado de maneira centralizada e distribuída para os roteadores
- Algoritmo de Dijkstra

Menor Caminho

- Um dos algoritmos mais simples
- A partir do modelo de grafos de uma rede interna gera uma sequência de nodos a serem percorridos para um pacote sair da origem e chegar ao destino.
- Algoritmo global, conhecimento completo do grafo
- Calculado de maneira centralizada e distribuída para os roteadores
- Algoritmo de Dijkstra

Menor Caminho

- Um dos algoritmos mais simples
- A partir do modelo de grafos de uma rede interna gera uma sequência de nodos a serem percorridos para um pacote sair da origem e chegar ao destino.
- Algoritmo global, conhecimento completo do grafo
- Calculado de maneira centralizada e distribuída para os roteadores
- Algoritmo de Dijkstra

Menor Caminho

- Um dos algoritmos mais simples
- A partir do modelo de grafos de uma rede interna gera uma sequência de nodos a serem percorridos para um pacote sair da origem e chegar ao destino.
- Algoritmo global, conhecimento completo do grafo
- Calculado de maneira centralizada e distribuída para os roteadores
- Algoritmo de Dijkstra

Menor Caminho

- Um dos algoritmos mais simples
- A partir do modelo de grafos de uma rede interna gera uma sequência de nodos a serem percorridos para um pacote sair da origem e chegar ao destino.
- Algoritmo global, conhecimento completo do grafo
- Calculado de maneira centralizada e distribuída para os roteadores
- Algoritmo de Dijkstra

Flooding

- Envia pacotes para todos os vizinhos, exceto pra de onde ele veio
- Necessita de controle para evitar o envio de infinitos pacotes
- Não costuma ser prático, exceto quando seu efeito é efetivamente o desejado
- Escolhe o menor caminho, pois escolhe todos simultaneamente

Flooding

- Envia pacotes para todos os vizinhos, exceto pra de onde ele veio
- Necessita de controle para evitar o envio de infinitos pacotes
- Não costuma ser prático, exceto quando seu efeito é efetivamente o desejado
- Escolhe o menor caminho, pois escolhe todos simultaneamente

Flooding

- Envia pacotes para todos os vizinhos, exceto pra de onde ele veio
- Necessita de controle para evitar o envio de infinitos pacotes
- Não costuma ser prático, exceto quando seu efeito é efetivamente o desejado
- Escolhe o menor caminho, pois escolhe todos simultaneamente

Flooding

- Envia pacotes para todos os vizinhos, exceto pra de onde ele veio
- Necessita de controle para evitar o envio de infinitos pacotes
- Não costuma ser prático, exceto quando seu efeito é efetivamente o desejado
- Escolhe o menor caminho, pois escolhe todos simultaneamente

Baseado em Fluxo

- Conta carga da rede junto da topologia
- Fluxo médio conhecido anteriormente
- Cálculo do atraso médio entre nodos
- Algoritmo que determine menor atraso médio determina o roteamento

Baseado em Fluxo

- Conta carga da rede junto da topologia
- Fluxo médio conhecido anteriormente
- Cálculo do atraso médio entre nodos
- Algoritmo que determine menor atraso médio determina o roteamento

Baseado em Fluxo

- Conta carga da rede junto da topologia
- Fluxo médio conhecido anteriormente
- Cálculo do atraso médio entre nodos
- Algoritmo que determine menor atraso médio determina o roteamento

Baseado em Fluxo

- Conta carga da rede junto da topologia
- Fluxo médio conhecido anteriormente
- Cálculo do atraso médio entre nodos
- Algoritmo que determine menor atraso médio determina o roteamento

Sumário

1 Introdução

2 Fundamentação

3 Algoritmos Estáticos

4 Algoritmos Dinâmicos

5 Conclusão

Vetor Distância

■ Algoritmo distribuído

- Cada roteador possui uma tabela (vetor) contendo a melhor distância conhecida até cada destino e a linha de saída preferencial utilizada para alcançá-lo.

- Utilizado na ARPANET: *Routing Information Protocol* (RIP)

- Objetivo: Encontrar o menor caminho

Algoritmo de Bellman-Ford

Vetor Distância

- Algoritmo distribuído
- Cada roteador possui uma tabela (vetor) contendo a melhor distância conhecida até cada destino e a linha de saída preferencial utilizada para alcançá-lo.
- Utilizado na ARPANET: *Routing Information Protocol* (RIP)
- Objetivo: Encontrar o menor caminho

Vetor Distância

- Algoritmo distribuído
- Cada roteador possui uma tabela (vetor) contendo a melhor distância conhecida até cada destino e a linha de saída preferencial utilizada para alcançá-lo.
- Utilizado na ARPANET: *Routing Information Protocol* (RIP)
- Objetivo: Encontrar o menor caminho
 - Algoritmo de Bellman-Ford

Vetor Distância

- Algoritmo distribuído
- Cada roteador possui uma tabela (vetor) contendo a melhor distância conhecida até cada destino e a linha de saída preferencial utilizada para alcançá-lo.
- Utilizado na ARPANET: *Routing Information Protocol* (RIP)
- Objetivo: Encontrar o menor caminho
 - Algoritmo de Bellman-Ford

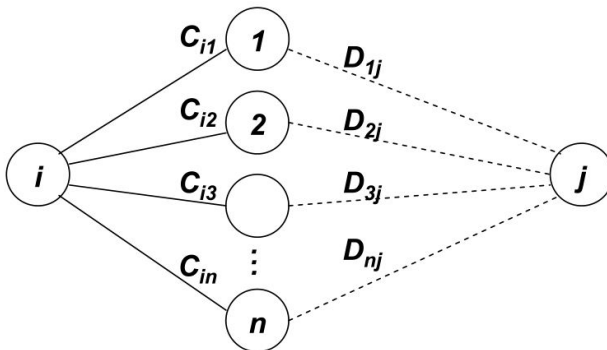
Vetor Distância

- Algoritmo distribuído
- Cada roteador possui uma tabela (vetor) contendo a melhor distância conhecida até cada destino e a linha de saída preferencial utilizada para alcançá-lo.
- Utilizado na ARPANET: *Routing Information Protocol* (RIP)
- Objetivo: Encontrar o menor caminho
 - Algoritmo de Bellman-Ford

Algoritmo de Bellman-Ford

Princípio

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



Algoritmo

Bellman-Ford(G, w, s)

Initialize-Single-Source(G, s)

for $i \leftarrow 1$ to $|V[G]| - 1$ **do**

for all $(u, v) \in E[G]$ **do**

$d[v] \leftarrow \min(d[v], d[u] + w(u, v))$

end for

end for

for all $(u, v) \leftarrow E[G]$ **do**

$d[v] \leftarrow \min(d[v], d[u] + w(u, v))$

end for

return TRUE

Algoritmo

Bellman-Ford(G, w, s)

Initialize-Single-Source(G, s)

for $i \leftarrow 1$ to $|V[G]| - 1$ **do**

for all $(u, v) \leftarrow E[G]$ **do**

$d[v] \leftarrow \min(d[v], d[u] + w(u, v))$

end for

for all $(u, v) \leftarrow E[G]$ **do**

$d[v] \leftarrow \min(d[v], d[u] + w(u, v))$

end for

return TRUE

Algoritmo

Bellman-Ford(G, w, s)

Initialize-Single-Source(G, s)

for $i \leftarrow 1$ to $|V[G]| - 1$ **do**

for all $(u, v) \leftarrow E[G]$ **do**

 Relax(u, v)

end for

end for

for all $(u, v) \leftarrow E[G]$ **do**

end for

return TRUE

Algoritmo

```
Bellman-Ford( $G, w, s$ )
Initialize-Single-Source( $G, s$ )
for  $i \leftarrow 1$  to  $|V[G]| - 1$  do
    for all  $(u, v) \leftarrow E[G]$  do
        Relax( $u, v$ )
    end for
end for
for all  $(u, v) \leftarrow E[G]$  do

    end for
return TRUE
```

Algoritmo

```
Bellman-Ford( $G, w, s$ )
Initialize-Single-Source( $G, s$ )
for  $i \leftarrow 1$  to  $|V[G]| - 1$  do
    for all  $(u, v) \leftarrow E[G]$  do
        Relax( $u, v$ )
    end for
end for
for all  $(u, v) \leftarrow E[G]$  do

    end for
return TRUE
```

Algoritmo

```
Bellman-Ford( $G, w, s$ )
Initialize-Single-Source( $G, s$ )
for  $i \leftarrow 1$  to  $|V[G]| - 1$  do
    for all  $(u, v) \leftarrow E[G]$  do
        Relax( $u, v$ )
    end for
end for
for all  $(u, v) \leftarrow E[G]$  do
    Relax( $u, v$ )
end for
return TRUE
```

Algoritmo

```

Bellman-Ford( $G, w, s$ )
Initialize-Single-Source( $G, s$ )
for  $i \leftarrow 1$  to  $|V[G]| - 1$  do
    for all  $(u, v) \leftarrow E[G]$  do
        Relax( $u, v$ )
    end for
end for
for all  $(u, v) \leftarrow E[G]$  do
    if  $d[v] > d[u] + w(u, v)$  then
        Relax( $u, v$ )
    end if
end for
return TRUE

```

Algoritmo

```
Bellman-Ford( $G, w, s$ )
Initialize-Single-Source( $G, s$ )
for  $i \leftarrow 1$  to  $|V[G]| - 1$  do
    for all  $(u, v) \leftarrow E[G]$  do
        Relax( $u, v$ )
    end for
end for
for all  $(u, v) \leftarrow E[G]$  do
    if  $d[v] > d[u] + w(u, v)$  then
        return FALSE
    end if
end for
return TRUE
```

Algoritmo

```
Bellman-Ford( $G, w, s$ )
Initialize-Single-Source( $G, s$ )
for  $i \leftarrow 1$  to  $|V[G]| - 1$  do
    for all  $(u, v) \leftarrow E[G]$  do
        Relax( $u, v$ )
    end for
end for
for all  $(u, v) \leftarrow E[G]$  do
    if  $d[v] > d[u] + w(u, v)$  then
        return FALSE
    end if
end for
return TRUE
```

Algoritmo

```
Bellman-Ford( $G, w, s$ )
Initialize-Single-Source( $G, s$ )
for  $i \leftarrow 1$  to  $|V[G]| - 1$  do
    for all  $(u, v) \leftarrow E[G]$  do
        Relax( $u, v$ )
    end for
end for
for all  $(u, v) \leftarrow E[G]$  do
    if  $d[v] > d[u] + w(u, v)$  then
        return FALSE
    end if
end for
return TRUE
```


Algoritmo

```
Bellman-Ford( $G, w, s$ )
Initialize-Single-Source( $G, s$ )
for  $i \leftarrow 1$  to  $|V[G]| - 1$  do
    for all  $(u, v) \leftarrow E[G]$  do
        Relax( $u, v$ )
    end for
end for
for all  $(u, v) \leftarrow E[G]$  do
    if  $d[v] > d[u] + w(u, v)$  then
        return FALSE
    end if
end for
return TRUE
```

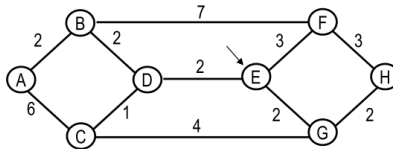
Algoritmo

```
Bellman-Ford( $G, w, s$ )
Initialize-Single-Source( $G, s$ )
for  $i \leftarrow 1$  to  $|V[G]| - 1$  do
    for all  $(u, v) \leftarrow E[G]$  do
        Relax( $u, v$ )
    end for
end for
for all  $(u, v) \leftarrow E[G]$  do
    if  $d[v] > d[u] + w(u, v)$  then
        return FALSE
    end if
end for
return TRUE
```

Algoritmo

```
Bellman-Ford( $G, w, s$ )
Initialize-Single-Source( $G, s$ )
for  $i \leftarrow 1$  to  $|V[G]| - 1$  do
    for all  $(u, v) \leftarrow E[G]$  do
        Relax( $u, v$ )
    end for
end for
for all  $(u, v) \leftarrow E[G]$  do
    if  $d[v] > d[u] + w(u, v)$  then
        return FALSE
    end if
end for
return TRUE
```

Vetor Distância



$C^E(v)$

D	2
F	3
G	2

Custo do enlace

$C^D(d)$	$C^F(d)$	$C^G(d)$
A 4	A 9	A 9
B 2	B 7	B 7
C 1	C 9	C 4
D 0	D 9	D 5
E 2	E 3	E 2
F 9	F 0	F 5
G 5	G 5	G 0
H 7	H 4	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

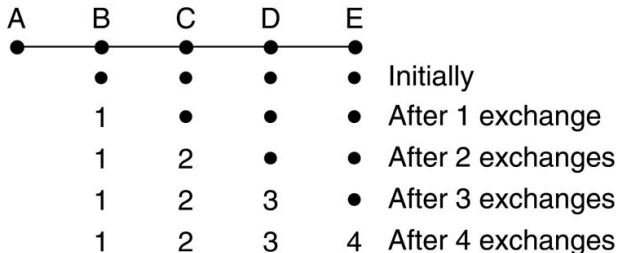
$C^E(D)+C^D(d)$ $C^E(F)+C^F(d)$ $C^E(G)+C^G(d)$

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

Convergência do Vetor Distância

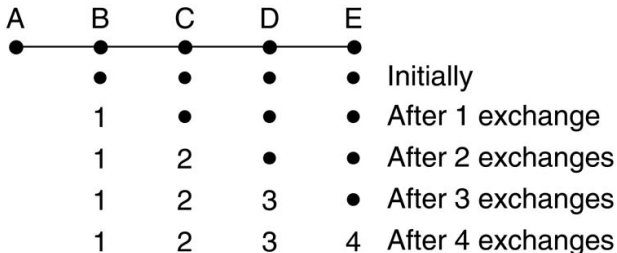
- Vetor distância reage bem (linearmente) a boas notícias:



- Já a más notícias...

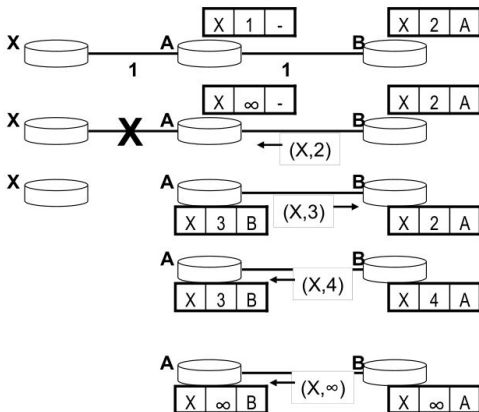
Convergência do Vetor Distância

- Vetor distância reage bem (linearmente) a boas notícias:



- Já a más notícias...

Problema da Contagem ao Infinito



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

Estado do *Link*

■ Descobrir vizinhos

■ Propagar informação

- Quem

- Vizinhas

- Número de sequência

■ Criação do mapa da rede

■ Recalcular e reenviar em caso de falha

■ Cálculo da melhor rota é independente

■ OSPF - Open Shortest Path First:

- Propagação em fluxo controlado em redes hierárquicas

- Roteamento baseado em estado

Estado do *Link*

- Descobrir vizinhos
- Propagar informação

- Quem
- Vizinhos
- Número de sequência

- Criação do mapa da rede
- Recalcular e reenviar em caso de falha
- Cálculo da melhor rota é independente
- OSPF - Open Shortest Path First:

Protocolo de roteamento utilizado em redes locais e em redes de área

Estado do *Link*

- Descobrir vizinhos
- Propagar informação
 - Quem
 - Vizinhos
 - Número de sequência
 - Criação do mapa da rede
 - Recalcular e reenviar em caso de falha
 - Cálculo da melhor rota é independente
 - OSPF - Open Shortest Path First:
 - Propagação em árvore espalhada em todas as direções

Estado do *Link*

- Descobrir vizinhos
- Propagar informação
 - Quem
 - Vizinhos
 - Número de sequência
- Criação do mapa da rede
- Recalcular e reenviar em caso de falha
- Cálculo da melhor rota é independente
- OSPF - Open Shortest Path First:
 - Cada nó envia mensagens para todos os vizinhos

Estado do *Link*

- Descobrir vizinhos
- Propagar informação
 - Quem
 - Vizinhos
 - Número de sequência
- Criação do mapa da rede
- Recalcular e reenviar em caso de falha
- Cálculo da melhor rota é independente
- OSPF - Open Shortest Path First:

Algoritmo de roteamento baseado em estado de link

Estado do *Link*

- Descobrir vizinhos
- Propagar informação
 - Quem
 - Vizinhos
 - Número de sequência
- Criação do mapa da rede
- Recalcular e reenviar em caso de falha
- Cálculo da melhor rota é independente
- OSPF - Open Shortest Path First:

Algoritmo de roteamento dinâmico baseado no estado dos links. Cada roteador mantém uma tabela de vizinhos e uma tabela de rotas.

Estado do *Link*

- Descobrir vizinhos
- Propagar informação
 - Quem
 - Vizinhos
 - Número de sequência
- Criação do mapa da rede
- Recalcular e reenviar em caso de falha
- Cálculo da melhor rota é independente
- OSPF - Open Shortest Path First:

Estado do *Link*

- Descobrir vizinhos
- Propagar informação
 - Quem
 - Vizinhos
 - Número de sequência
- Criação do mapa da rede
- Recalcular e reenviar em caso de falha
- Cálculo da melhor rota é independente
- OSPF - Open Shortest Path First:
 - Protocolo muito utilizado em redes internas, baseado em link-state.

Estado do *Link*

- Descobrir vizinhos
- Propagar informação
 - Quem
 - Vizinhos
 - Número de sequência
- Criação do mapa da rede
- Recalcular e reenviar em caso de falha
- Cálculo da melhor rota é independente
- OSPF - Open Shortest Path First:
 - Protocolo muito utilizado em redes internas, baseado em link-state.

Estado do *Link*

- Descobrir vizinhos
- Propagar informação
 - Quem
 - Vizinhos
 - Número de sequência
- Criação do mapa da rede
- Recalcular e reenviar em caso de falha
- Cálculo da melhor rota é independente
- OSPF - Open Shortest Path First:
 - Protocolo muito utilizado em redes internas, baseado em link-state.

Roteamento Hierárquico

- Utilizado no roteamento externo (inter-rede)
- ~2bi de dispositivos → 35.000 Sistemas Autônomos (2010)
- Exemplo: Border Gateway Protocol (BGP)
- Responsável por tornar a Internet uma aplicação verdadeiramente distribuída.
- Obediência a leis internacionais e decisões políticas

Roteamento Hierárquico

- Utilizado no roteamento externo (inter-rede)
- ~2bi de dispositivos → 35.000 Sistemas Autônomos (2010)
- Exemplo: Border Gateway Protocol (BGP)
- Responsável por tornar a Internet uma aplicação verdadeiramente distribuída.
- Obediência a leis internacionais e decisões políticas

Roteamento Hierárquico

- Utilizado no roteamento externo (inter-rede)
- ~2bi de dispositivos → 35.000 Sistemas Autônomos (2010)
- Exemplo: Border Gateway Protocol (BGP)
- Responsável por tornar a Internet uma aplicação verdadeiramente distribuída.
- Obediência a leis internacionais e decisões políticas

Roteamento Hierárquico

- Utilizado no roteamento externo (inter-rede)
- ~2bi de dispositivos → 35.000 Sistemas Autônomos (2010)
- Exemplo: Border Gateway Protocol (BGP)
- Responsável por tornar a Internet uma aplicação verdadeiramente distribuída.
- Obediência a leis internacionais e decisões políticas

Roteamento Hierárquico

- Utilizado no roteamento externo (inter-rede)
- ~2bi de dispositivos → 35.000 Sistemas Autônomos (2010)
- Exemplo: Border Gateway Protocol (BGP)
- Responsável por tornar a Internet uma aplicação verdadeiramente distribuída.
- Obediência a leis internacionais e decisões políticas

Roteamento Hierárquico

- Funcionamento básico: Similar ao vetor distância. Roteadores enviam uns aos outros duas informações:
 - Que eles estão vivos e quais redes (Faixas de IPs) estão sob sua responsabilidade.
 - Qual a rota completa que utilizam para chegar ao destino (Solução para o problema da contagem ao infinito!)

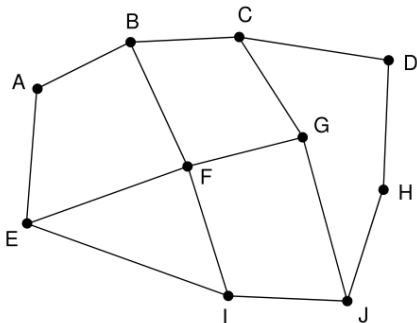
Roteamento Hierárquico

- Funcionamento básico: Similar ao vetor distância. Roteadores enviam uns aos outros duas informações:
 - Que eles estão vivos e quais redes (Faixas de IPs) estão sob sua responsabilidade.
 - Qual a rota completa que utilizam para chegar ao destino (Solução para o problema da contagem ao infinito!)

Roteamento Hierárquico

- Funcionamento básico: Similar ao vetor distância. Roteadores enviam uns aos outros duas informações:
 - Que eles estão vivos e quais redes (Faixas de IPs) estão sob sua responsabilidade.
 - Qual a rota completa que utilizam para chegar ao destino (Solução para o problema da contagem ao infinito!)

Roteamento Hierárquico



Informações sobre D que
F recebe de seus vizinhos

De B: "Eu utilizo BCD"

De G: "Eu utilizo GCD"

De I: "Eu utilizo IFGCD"

De E: "Eu utilizo EFGCD"

- O que acontece se G cair?

Sumário

1 Introdução

2 Fundamentação

3 Algoritmos Estáticos

4 Algoritmos Dinâmicos

5 Conclusão

Conclusão

- Grafos e seus algoritmos são ferramentas extremamente úteis em redes de computadores. Permitem a modelagem e solução de diferentes situações encontradas na área.
- A Internet como a conhecemos só existe hoje graças aos avanços no desenvolvimento de melhores e mais eficientes algoritmos de roteamento.
- Não há uma "bala de prata", diferentes algoritmos servem a diferentes propósitos

Conclusão

- Grafos e seus algoritmos são ferramentas extremamente úteis em redes de computadores. Permitem a modelagem e solução de diferentes situações encontradas na área.
- A Internet como a conhecemos só existe hoje graças aos avanços no desenvolvimento de melhores e mais eficientes algoritmos de roteamento.
- Não há uma "bala de prata", diferentes algoritmos servem a diferentes propósitos

Conclusão

- Grafos e seus algoritmos são ferramentas extremamente úteis em redes de computadores. Permitem a modelagem e solução de diferentes situações encontradas na área.
- A Internet como a conhecemos só existe hoje graças aos avanços no desenvolvimento de melhores e mais eficientes algoritmos de roteamento.
- Não há uma "bala de prata", diferentes algoritmos servem a diferentes propósitos

Bibliografia

- TANENBAUM, A. S. Redes de Computadores. 4 ed. São Paulo: Elsevier, 2003.
- DE CASTRO, M. C. F. Redes Comutadas. setembro de 2002. Disponível em: <<http://www.ee.pucrs.br/~decastro/download.html>>.
- CARISSIMI, A. Algoritmos de roteamento. 2008. Disponível em: <<http://www.inf.ufrgs.br/~asc/redes/pdf/aula21.pdf>>.
- CAMPONOGARA, E. Caminhos Mínimos Com Uma Fonte. abril de 2009. Disponível em: <http://www.das.ufsc.br/~camponog/Disciplinas/DAS-9003/slides_CLR/114-shortest-path.pdf>

Bibliografia

- TANENBAUM, A. S. Redes de Computadores. 4 ed. São Paulo: Elsevier, 2003.
- DE CASTRO, M. C. F. Redes Comutadas. setembro de 2002. Disponível em: <<http://www.ee.pucrs.br/~decastro/download.html>>.
- CARISSIMI, A. Algoritmos de roteamento. 2008. Disponível em: <<http://www.inf.ufrgs.br/~asc/redes/pdf/aula21.pdf>>.
- CAMPONOGARA, E. Caminhos Mínimos Com Uma Fonte. abril de 2009. Disponível em: <http://www.das.ufsc.br/~camponog/Disciplinas/DAS-9003/slides_CLR/114-shortest-path.pdf>

Bibliografia

- TANENBAUM, A. S. Redes de Computadores. 4 ed. São Paulo: Elsevier, 2003.
- DE CASTRO, M. C. F. Redes Comutadas. setembro de 2002. Disponível em: <<http://www.ee.pucrs.br/~decastro/download.html>>.
- CARISSIMI, A. Algoritmos de roteamento. 2008. Disponível em: <<http://www.inf.ufrgs.br/~asc/redes/pdf/aula21.pdf>>.
- CAMPONOGARA, E. Caminhos Mínimos Com Uma Fonte. abril de 2009. Disponível em: <http://www.das.ufsc.br/~camponog/Disciplinas/DAS-9003/slides_CLR/114-shortest-path.pdf>

Bibliografia

- TANENBAUM, A. S. Redes de Computadores. 4 ed. São Paulo: Elsevier, 2003.
- DE CASTRO, M. C. F. Redes Comutadas. setembro de 2002. Disponível em: <<http://www.ee.pucrs.br/~decastro/download.html>>.
- CARISSIMI, A. Algoritmos de roteamento. 2008. Disponível em: <<http://www.inf.ufrgs.br/~asc/redes/pdf/aula21.pdf>>.
- CAMPONOGARA, E. Caminhos Mínimos Com Uma Fonte. abril de 2009. Disponível em: <http://www.das.ufsc.br/~camponog/Disciplinas/DAS-9003/slides_CLR/114-shortest-path.pdf>