PROTOCOLOS DE ROTEAMENTO

- **O roteador em uma rede geograficamente distribuída é um dos principais dispositivos responsáveis pelo sucesso do ambiente.**
- **Roteadores são os dispositivos responsáveis pelo recebimento e redirecionamento dos pacotes na rede.**
- **A decisão de redirecionamento é usualmente baseada na topologia e nas condições de contorno do ambiente.**

Quanto à topologia, é preciso considerar que diferentes ambientes de rede têm complexidade diferenciada. Podem existir

- **configurações de rede como um sistema autônomo com poucos computadores e apenas uma sub-rede, ou**
- **configurações com milhares de computadores e inúmeras sub-redes (ou até centenas de milhares de computadores e milhares de sub-redes).**

Quanto às condições de contorno, é preciso considerar que uma rede pode ter em dado momento

- **um ou mais enlaces fora do ar ou**
- **apresentar um congestionamento em um determinado trecho da rede.**

As condições de contorno provêem informações tais como retardos e interrupções na configuração da rede.

- **Visando uma eficiência no redirecionamento dos pacotes em uma determinada rede, os roteadores trocam entre si, através de protocolos de roteamento, informações sobre o ambiente de rede.**
- **A informação de roteamento é composta pela topologia e condições de contorno.**
- **O algoritmo de roteamento, baseado nas informações de roteamento, efetua o melhor redirecionamento do pacote.**
- **O algoritmo de roteamento tem a função de montar uma tabela para que o roteamento dos pacotes seja efetuado de forma precisa.**

- Os protocolos de roteamento são as rotinas de elaboração de mapas ou tabelas pelas quais os roteadores descobrem o formato da rede.

- São baseados em algoritmos de roteamento, os quais adotam diferentes abordagens.

- A classificação dos protocolos de roteamento é feita de acordo com as diferentes abordagens adotadas pelos algoritmos de roteamento.

⇒ **Protocolos de roteamento adaptativos:**
Baseados em algoritmos de roteamento adaptativos, em que as decisões tomadas constituem um reflexo da carga da rede e de possíveis trocas na topologia da rede.

⇒ **Protocolos de roteamento não-adaptativos:**
Baseados em algoritmos de roteamento não-adaptativos (ou estáticos), que não consideram em suas decisões medidas (ou estimativas de tráfego) e a topologia da rede.

As Tabelas de Roteamento são classificadas em **Estáticas** e **Dinâmicas**.

1. Tabelas Estáticas:

- Resultam do tipo de abordagem conhecido como rotas diretas e estáticas.
- Todas as informações da tabela de roteamento são inseridas de maneira direta e não existe possibilidade de mudança de informação.

2. Tabelas Dinâmicas:

- São montadas e atualizadas constantemente, visando possibilitar que as interligações entre roteadores sejam efetuadas de forma contínua, no ambiente de rede.
- Os roteadores anunciam entre si, de forma dinâmica, as suas ligações, através de protocolos de comunicação roteador-roteador.
- É a forma mais tradicional de operação de protocolos de roteamento.

3. Abordagem intermediária adotada para a construção de tabelas de roteamento: Roteamento *Default*.

- Neste paradigma de roteamento é estabelecido um determinado nº mínimo de rotas de maneira direta, e os demais caminhos são atribuídos a um roteador *default*.

Protocolos que utilizam **Tabelas Dinâmicas** são chamados **Protocolos de Roteamento Dinâmicos.**

Na implementação de tais protocolos, dois parâmetros são essenciais para o estabelecimento do procedimento de roteamento:

1. **Conectividade:**
   O protocolo deve saber se existe uma ligação direta entre os dois segmentos a serem roteados ou quantos pulos (*hops*) são necessários para o acesso à rede remota.

2. **Custo:**
   Qual é o menor custo entre os diversos caminhos (*paths*) existentes para interligar as duas redes.

CONSIDERAÇÕES SOBRE PROTOCOLOS DE ROTEAMENTO

Os protocolos de roteamento são projetados para serem ferramentas capazes de realizar várias tarefas.

Podem fornecer informações sobre:

- ***throughput*** **(qualidade da transmissão, medida pelo nº de bytes do usuário que são transferidos por segundo, sobre algum intervalo de tempo)**
- **qualidade dos circuitos entre os roteadores**
- ***status*** **operacional de roteadores específicos**
- **nº de pontos intermediários (ou *hops*) que um pacote deverá atravessar**
- **caminhos alternativos disponíveis em caso de falhas na rede**

O *software* do roteador pondera essas e outras informações usando algoritmos específicos e depois determina como enviar um pacote em sua viagem entre segmentos da rede.

Outras informações que o *software* do roteador leva em consideração:

- **Velocidade de transmissão.**
- **Retardo de propagação:** tempo que um pacote leva para passar de uma rede a outra.
- **Retardos de enfileiramento:** ocorrem quando os roteadores ou as chaves de comutação recebem tráfego intenso demais.
- **Custo do *link*:** pode ser formado com o uso de alguma fórmula que considere, por exemplo, o aluguel mensal de linhas privadas, ou a tarifação de linhas telefônicas comuns. É possível instruir o *software* do roteador a só usar uma linha telefônica como último recurso, atribuindo a esse circuito um custo maior que o dos outros.

- Os roteadores modernos utilizam uma arquitetura adaptativa e distribuída.
- O *software* consegue se adaptar a mudanças no *status* de um circuito ou de outro roteador em questão de milissegundos, e cada roteador mantém suas próprias tabelas de informações.
- Os sistemas antigos eram menos flexíveis e dependiam dos dados armazenados em um nó central.

Quanto à localização física entre *hosts*, o roteamento pode ser Direto ou Indireto.

Roteamento Direto:

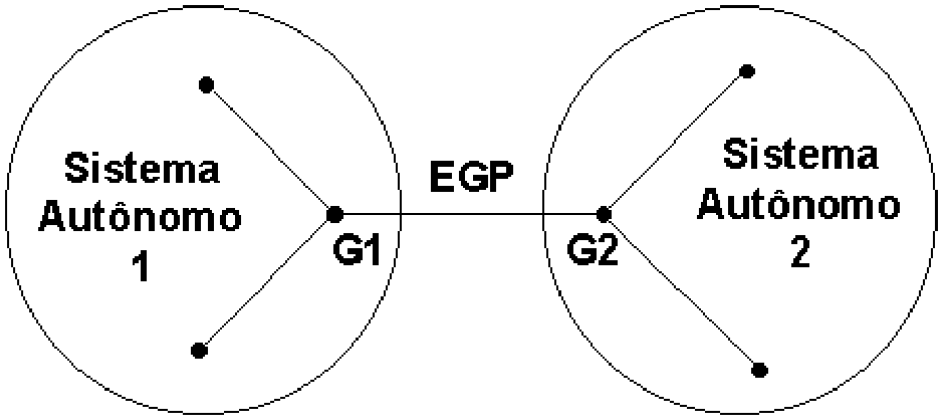
- **Comunicação entre dois *hosts* alocados em uma mesma rede física.**

Roteamento Indireto:

- **Conexão entre dois *hosts* alocados em redes distintas.**
- **É necessário o uso de *gateways* para efetuar o encaminhamento dos pacotes à rede destino.**

Quanto à vizinhança entre *gateways* que trocam informações de roteamento, os protocolos podem ser Internos ou Externos.

- Sistemas Autônomos são sistemas em que um *site* composto por uma ou diversas redes e *gateways* são administrados por uma única entidade.
- SAs têm livre escolha do protocolo a ser utilizado para descobrir, manter, divulgar e atualizar rotas dentro do seu universo
- SAs necessitam de um protocolo para se comunicar com sistemas autônomos vizinhos.



A Figura ao lado ilustra o conceito de vizinhança, em que dois SAs (1 e 2) estão conectados através de seus respectivos *Gateways* G1 e G2.

Protocolos de Roteamento Externos

Gateways **que trocam informações de roteamento com outros *gateways* que não pertencem ao mesmo Sistema Autônomo são considerados Vizinhos Exteriores e utilizam o protocolo EGP (*Exterior Gateway Protocol*) para se comunicarem.**

Protocolos EGP possuem três características principais:

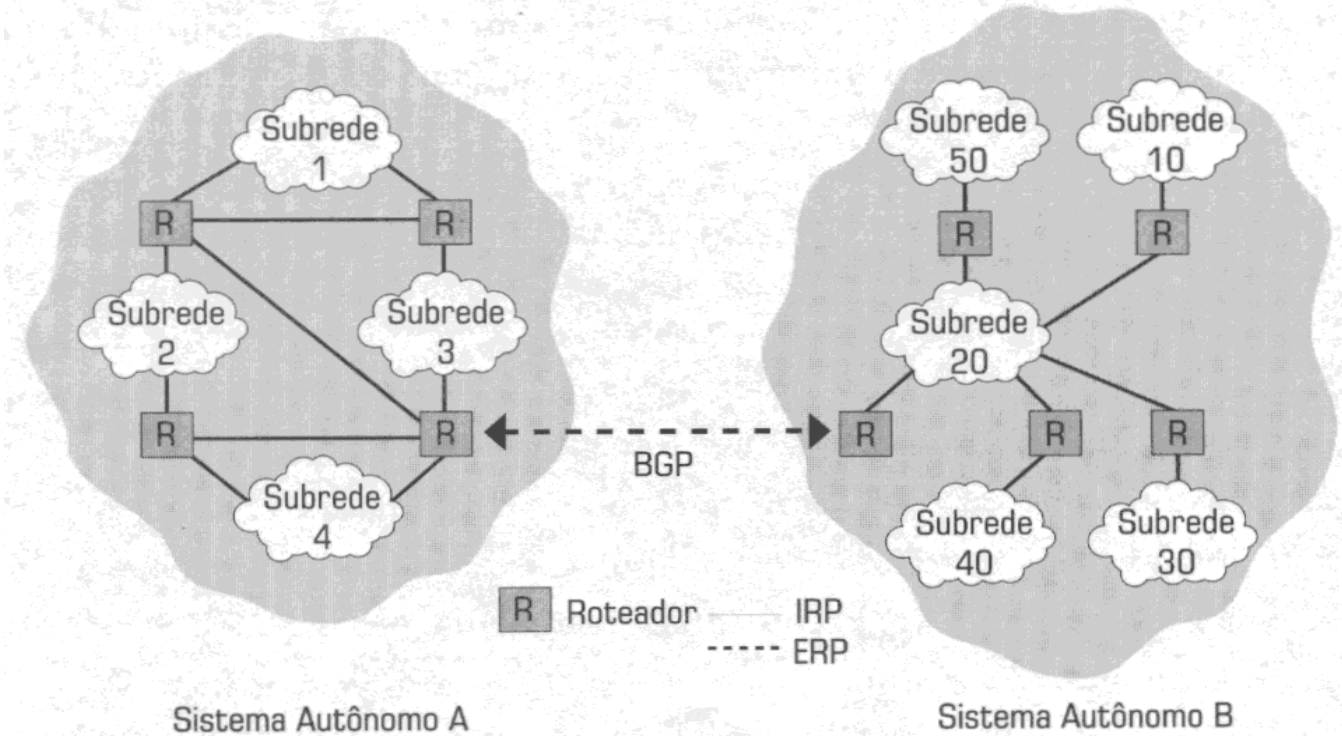
- **Suportam mecanismo de aquisição de vizinho.**
- **Fazem testes contínuos para ver se os vizinhos estão respondendo.**
- **Divulgam informação entre vizinhos utilizando mensagens de atualização de rotas.**

O protocolo BGP (*Border Gateway Protocol*) é um exemplo de protocolo EGP.

Protocolos de Roteamento Internos

Gateways **que trocam informações de roteamento somente com** ***gateways*** **do mesmo Sistema Autônomo são considerados Vizinhos Interiores e utilizam diversos protocolos denominados genericamente IGP (*Interior Gateway Protocols*).**

Os protocolos RIP (*Routing Information Protocol*), HELLO, OSPF (*Open Shortest Path First*) e IGRP (*Internal Gateway Routing Protocol*) são exemplos de protocolos IGP.



Sistemas Autônomos, ERP e IRP

Sistemas Autônomos (A e B), ambos com roteamento interno efetuado pelo protocolo OSPF (*Open Shortest Path First*), que é um protocolo IRP (*Interior Router Protocol*).

O roteamento externo que efetua a ligação entre os dois sistemas autônomos é realizado pelo protocolo BGP (*Border Gateway Protocol*), que é um protocolo ERP (*Exterior Router Protocol*).

ABORDAGENS UTILIZADAS PARA A TROCA DE INFORMAÇÃO ENTRE ROTEADORES

Os protocolos de roteamento podem efetuar a troca de informação entre os roteadores empregando as seguintes abordagens:

- **Roteamento Hierárquico (*Hierarchical Routing*)**
- **Roteamento por *Broadcast***
- **Roteamento por *Multicast***

Roteamento Hierárquico (*Hierarchical Routing*)

- **O roteamento hierárquico é realizado em áreas (chamadas regiões).**
- **Cada roteador conhece apenas a sua região.**
- **Para grandes redes são necessárias algumas sub-divisões para que os roteadores possam trabalhar com eficiência.**
- **Tais subdivisões atendem a hierarquias, constituindo:**
  - ***clusters* de regiões,**
  - **zonas de *clusters*,**
  - **grupos de zonas, etc.**

Roteamento por *Broadcast*

- Tipo de abordagem por difusão, em que os pacotes são enviados para todos os roteadores simultaneamente.
- Vários protocolos podem implementar o *broadcasting*, através de algoritmos como:
  - algoritmo de *broadcast* simples
  - algoritmo de *flooding*
  - algoritmo de multidestinos

Roteamento por *Multicast*

- Tipo de abordagem em que pacotes são enviados a um grupo seleto de roteadores.
- Existe a necessidade da figura de gerência de grupos.
- Tarefas de gerência executadas no protocolo por *multicast*:
  - criação/destruição de grupos,
  - requisição de processos que requeiram a junção/disjunção a um determinado grupo.

ALGORITMOS DE ROTEAMENTO

- Quando os pacotes são roteados de uma máquina fonte para uma máquina de destino, na maior parte das sub-redes, os pacotes poderão necessitar de muitos saltos (*hops*) para seguir sua jornada.
- A única exceção é para redes do tipo *broadcast*. No entanto, mesmo neste caso, o roteamento é problemático se a fonte e o destino não estão na mesma rede.
- Os algoritmos que escolhem as rotas e as estruturas de dados que serão usadas são muito importantes no projeto de redes.
- **O algoritmo de roteamento é o *software* responsável por decidir sobre qual linha de saída um pacote que chega deverá ser transmitido.**

- Se as sub-redes usam datagramas internamente, a decisão pela linha de saída deve ser feita a cada pacote de dados que chega, visto que a melhor rota pode mudar constantemente.

- Se as sub-redes usam circuitos virtuais (VCs) internamente, as decisões de roteamento são feitas somente quando um novo VC está sendo estabelecido.
  Conseqüentemente, pacotes de dados apenas seguem uma rota previamente estabelecida.
  Este caso é algumas vezes chamado *session routing*, porque uma rota permanece preponderante por uma seção inteira do usuário (por exemplo, uma seção de *login* em um terminal ou uma transferência de arquivos).

- Não importando se as rotas foram escolhidas independentemente para cada pacote ou somente quando novas conexões são estabelecidas, há propriedades que são desejáveis em um algoritmo de roteamento:

  → Correção
  → Estabilidade
  → Simplicidade
  → Eqüidade (imparcialidade)
  → Robustez
  → Desempenho ótimo.

Robustez:

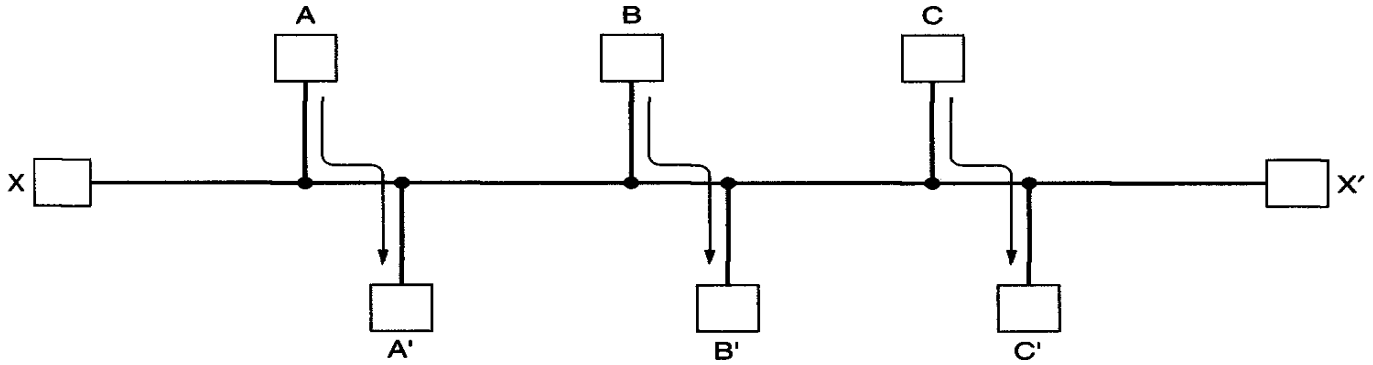
- Quando uma grande rede "entra no ar" espera-se que ela opere por anos sem falhas em nível de sistema.
- Durante tal período podem ocorrer falhas de *software* e *hardware* de todos os tipos. Os *hosts*, os roteadores e as linhas terão seu funcionamento interrompido e voltarão a funcionar repetidas vezes, e a topologia da rede irá mudar freqüentemente.
- O algoritmo de roteamento deverá ser hábil para lidar com mudanças na topologia e no tráfego sem requerer que as tarefas executadas em todos os *hosts* sejam abortadas, cada vez que algum roteador apresentar problemas.

Estabilidade:

- Existem algoritmos de roteamento que nunca convergem (nunca atingem uma condição de equilíbrio), não importando quanto tempo permaneçam operando.

Eqüidade (Imparcialidade) e Desempenho Ótimo:

- Muitas vezes são objetivos conflitantes.
- Suponha que haja tráfego suficiente entre A e A', entre B e B' e entre C e C' para saturar os *links* horizontais, na Figura abaixo.



- Para maximizar o fluxo total, o tráfego de X para X' precisará ser bloqueado inteiramente.
- Sob o ponto de vista de X e X', esta decisão pode não ser considerada justa.
- Algum compromisso entre eficiência global e justiça (imparcialidade) com relação a conexões individuais precisa ser adotado, por explo:

↓nº hops → ↓delay → ↓largura de banda → ↑desempenho global da rede

CLASSES DE ALGORITMOS DE ROTEAMENTO

1. ALGORITMOS NÃO-ADAPTATIVOS (OU ESTÁTICOS):

- Não baseiam suas decisões de roteamento em medidas ou estimativas de tráfego e topologia.
- A escolha da rota a ser usada para trafegar de I até J (quaisquer dois pontos na rede) é computada antecipadamente, *off-line*, e passada para os roteadores quando a rede é inicializada.

2. ALGORITMOS ADAPTATIVOS (OU DINÂMICOS):

- Modificam suas decisões de roteamento em resposta a mudanças na topologia e no tráfego.
- Diferem entre si nos seguintes aspectos:

→ forma em que obtêm a informação (localmente; a partir de roteadores adjacentes; a partir de todos os roteadores);

→ intervalo de tempo em que mudam as rotas (a cada Δt segundos; quando a carga muda; quando a topologia muda);

→ métrica que é utilizada para otimização (distância; nº de *hops*; tempo estimado de trânsito).

EXEMPLOS DE HEURÍSTICAS EM QUE SE BASEIAM OS ALGORITMOS DE ROTEAMENTO:

- *Shortest Path*: o conceito empregado neste tipo de algoritmo é a construção de um grafo da subrede. Cada ponto do grafo representa um roteador, e cada linha um meio de comunicação.

  Para achar uma rota entre dois roteadores quaisquer, o algoritmo acha a rota com menor distância (shortest path). Diversas métricas são empregadas neste tipo de protocolo, tais como o número de roteadores pulados na rede (hops) e a distância. Vale lembrar que protocolos baseados nesta abordagem são estáticos.

- *Flooding*: este tipo de protocolo estático é baseado na técnica de que cada pacote que chega será enviado para todas as interfaces do roteador, exceto aquela pela qual o pacote chegou. A conclusão mais obvia é que este tipo de abordagem provoca uma *inundação* de pacotes. Os protocolos baseados em *flooding*, embora não pareçam práticos, são utilizados onde uma solução robusta é necessária. Exemplos dessa utilização são: (1), o caso de redes militares, onde todos roteadores devem ser notificados para uma dada aplicação; (2), em banco de dados distribuídos, onde muitas vezes todas as bases devem ser atualizadas. Existem procedimentos chamados de represadores, que ajudam a melhorar o desempenho deste tipos de algoritmos. Dentre estas técnicas temos:

  1. Contadores de hops – fazem o decremento a cada pulo (hop).
  2. Reconhecimento de pacotes flooding – estes não serão reenviados.
  3. Flooding seletivo – só é feito um broadcast numa determinada direção.

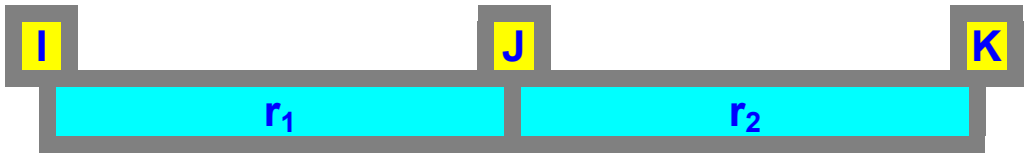
- *Flow-Based* (baseado em fluxo): neste método estático, a carga de ligação entre dois pontos quaisquer e a topologia são levados em consideração. As métricas (carga e topologia) são conhecidas previamente para a consideração do roteamento nos protocolos baseados nesta abordagem.
- *Distance Vector* (vetor de distância): a maioria das redes modernas emprega algoritmos de roteamento dinâmico. O algoritmo conhecido por distance vector é um dos dois mais utilizados. A concepção deste protocolo é manter sua tabela (ou vetor) com as melhores distâncias para cada destino através de uma específica linha. O vetor é mantido atualizado através do processo dinâmico de troca de informação com seus roteadores vizinhos. Algumas informações do vetor são, por exemplo, o número de hops, o tempo de retardo em milisegundos e o número total de pacotes enfileirados ao longo do path (caminho).

- Link State: este protocolo com abordagem dinâmica é um dos dois mais populares algoritmos empregados em redes modernas. O link state substituiu o protocolo vector distance na rede ARPANET. Dentre as razões, a métrica de distância não leva em consideração a largura de banda das linhas, e o conceito de vetor demora muito na convergência. São cinco os pontos nos quais este protocolo se baseia:

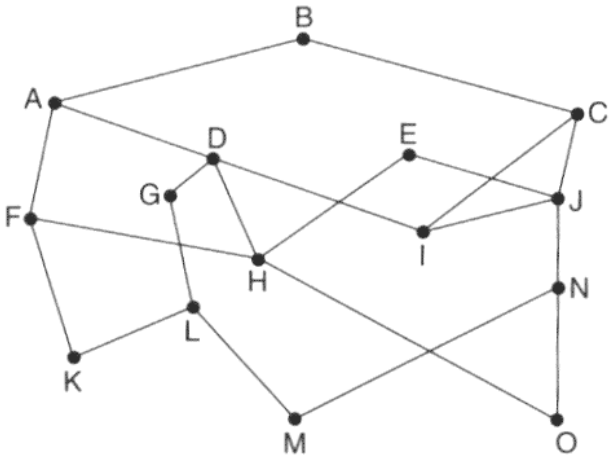
  1. Descobrir seus vizinhos e seus endereços de rede.
  2. Calcular o retardo ou custo para cada um de seus vizinhos.
  3. Construir um pacote contanto tudo o que este aprendeu.
  4. Propagar o pacote conhecedor de todas as informações para todos os roteadores.
  5. Computar o menor caminho para todos os outros roteadores.

PRINCÍPIO GERAL SOBRE ROTAS ÓTIMAS

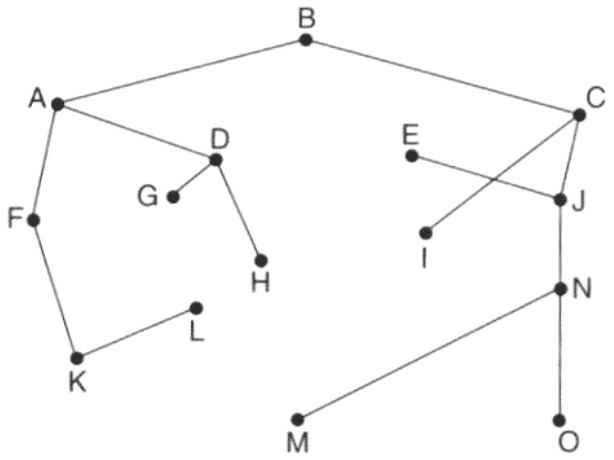
- **Se o roteador J está no caminho ótimo do roteador I p/ o roteador K, então o caminho ótimo de J p/ K também passa pela mesma rota.**
- **Denomine a parte da rota que vai de I p/ J como r_1 e o resto da rota como r_2.**



- **Se existisse uma rota melhor do que r_2 de J p/ K, poderia ser concatenada com r_1 para melhorar a rota de I para K, contradizendo a afirmação de que r_1r_2 é ótima.**
- **Como conseqüência direta do Princípio Geral sobre Rotas Ótimas, pode-se afirmar que o conjunto de rotas ótimas de todas as fontes para um dado destino forma uma árvore cujas raízes estão no destino.**
- **Tal árvore (estrutura de roteamento) é chamada *sink tree* e é ilustrada na Figura que segue, onde a distância métrica é o número de *hops*.**



(a)



(b)

(a) Uma sub-rede. **(b) Uma *sink tree* para o roteador B.**

- **O objetivo de todos os algoritmos de roteamento é descobrir e usar a *sink tree* para todos os roteamentos.**
- **Como todas as árvores, uma *sink tree* também não contém *loops*, de tal forma que cada pacote pode ser entregue dentro de um nº finito de *hops*.**
- **O princípio geral sobre rotas ótimas e a *sink tree* constituem termos de comparação para a medida da qualidade dos algoritmos de roteamento.**

ALGORITMOS DE ROTEAMENTO ESTÁTICOS

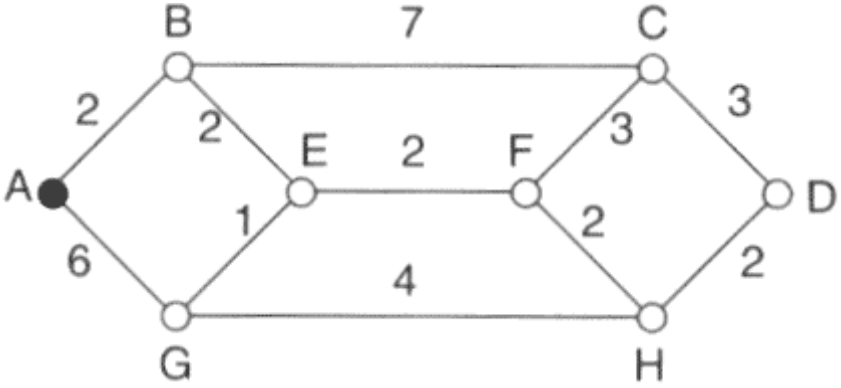
Roteamento pelo Caminho mais Curto
Shortest Path Routing

- **Técnica simples, amplamente adotada.**
- **Consiste em construir um grafo da sub-rede, em que cada nó representa um roteador e cada arco do grafo representa uma linha de comunicação (ou *link*).**
- **Para escolher uma rota entre um dado par de roteadores, o algoritmo precisa apenas determinar o caminho mais curto entre os roteadores, no grafo.**
- **A heurística para determinar o caminho mais curto entre fonte e destino pode ser baseada em diferentes métricas:**
  - **número de *hops* entre fonte e destino.**
  - **distância física (geográfica).**
  - **fila média e atraso de transmissão associados a cada arco no caminho, para algum pacote padrão de teste, transmitido a intervalos regulares (hora a hora, p. explo). O caminho mais curto é o caminho mais rápido, ao invés daquele com menor número de arcos ou km.**

- **Em geral, os *labels* nos arcos podem ser computados como função de mais de um argumento, entre os quais:**

- **distância,**
- **custo de comunicação,**
- **largura de banda,**
- **tamanho médio da fila,**
- **tráfego médio,**
- **alguma medida de atraso,**

- **O algoritmo pode calcular o caminho mais curto através de um dos critérios citados ou através de uma combinação ponderada dos diferentes critérios.**



» **Caminho mais curto = menor número de *hops* entre fonte e destino.**
Caminhos ABC e ABE são igualmente longos.

» **Caminho mais curto = menor distância geométrica (em km).**
Caminho ABC é muito mais longo do que caminho ABE (assumindo que a figura esteja desenhada em escala).

Algoritmo de Dijkstra

(*Shortest Path Routing*)

1. **Cada nó é etiquetado (um *label* associado ao nó é escrito entre parênteses) com a distância desde o nó fonte, ao longo do melhor caminho conhecido.**
2. **Inicialmente, todos os caminhos são desconhecidos, de forma que todos os nós são etiquetados com "infinito" (∞).**
3. **À medida que o algoritmo prossegue e os caminhos são determinados, os *labels* podem mudar, refletindo melhores caminhos.**
4. **Um *label* pode ser experimental ou permanente.**
5. **Inicialmente, todos os *labels* são experimentais.**
6. **Quando é descoberto que um *label* representa o menor caminho possível entre a fonte e aquele específico nó, ele é feito permanente e não será mais alterado.**

Exemplo de funcionamento do algoritmo de Dijkstra:

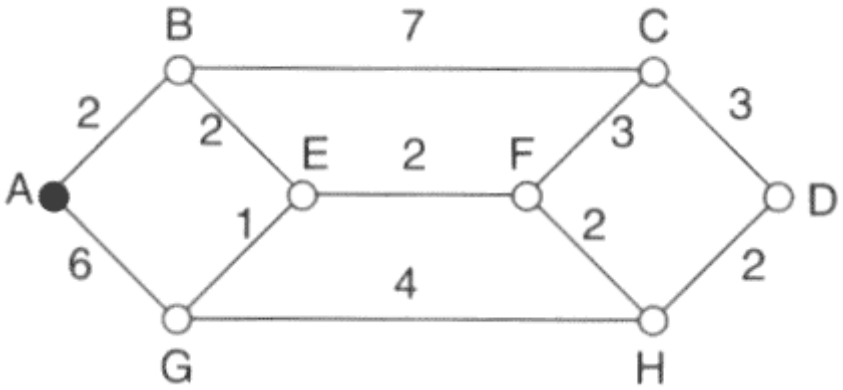


Figura (a)

Os arcos mostrados no grafo da Figura (a) estão associados a pesos que representam (por exemplo) distância física entre os nós.

Desejamos determinar o menor caminho, de A até D.

1. **Nó A é marcado como nó permanente (círculo cheio).**
2. **Nó A é chamado "nó de trabalho".**

3. **Cada um dos nós adjacentes ao nó A é examinado e cada nó é re-etiquetado com a distância entre o nó adjacente e o nó A (*label* provisório).**
4. **Sempre que um nó é re-etiquetado, também é marcado com a identificação do nó a partir do qual a "sondagem" foi feita, para que o caminho final possa ser reconstruído posteriormente.**
5. **Tendo sido examinados todos os nós adjacentes ao nó A, todos os nós do grafo etiquetados provisoriamente são verificados e aquele nó com o menor valor de etiqueta é feito permanente. Este nó passa a ser o novo nó de trabalho. No caso, o nó B, mostrado na Figura (b).**

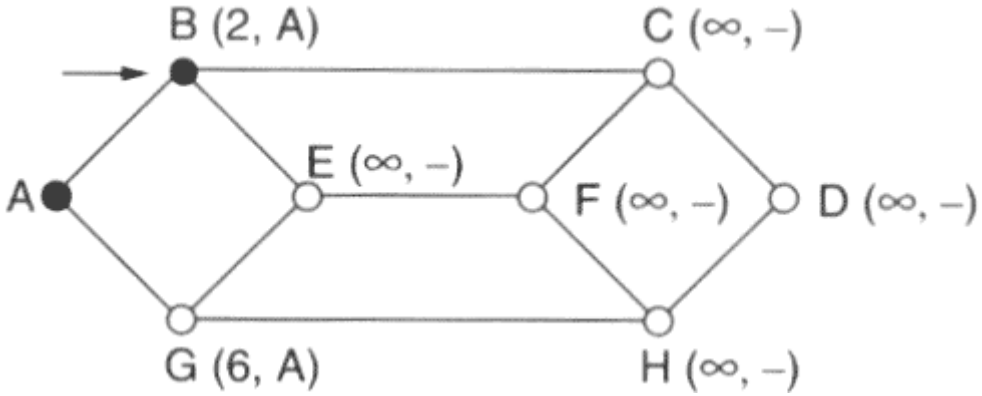


Figura (b)

6. **Partindo do novo nó de trabalho (nó B), cada um dos nós adjacentes é examinado. Se a soma do *label* em B com a distância entre o nó B e o nó que está sendo considerado for menor que o label naquele nó, estará definido o menor caminho e o nó é re-etiquetado.**
7. **Após todos os nós adjacentes ao nó de trabalho terem sido inspecionados e os nós provisórios alterados (se possível), é executada uma busca sobre todo o grafo pelo nó etiquetado provisoriamente de menor valor. Este nó é tornado permanente e passa a ser o nó de trabalho para o próximo passo do algoritmo. Em nosso exemplo, o novo nó de trabalho é o nó E, mostrado na Figura (c).**

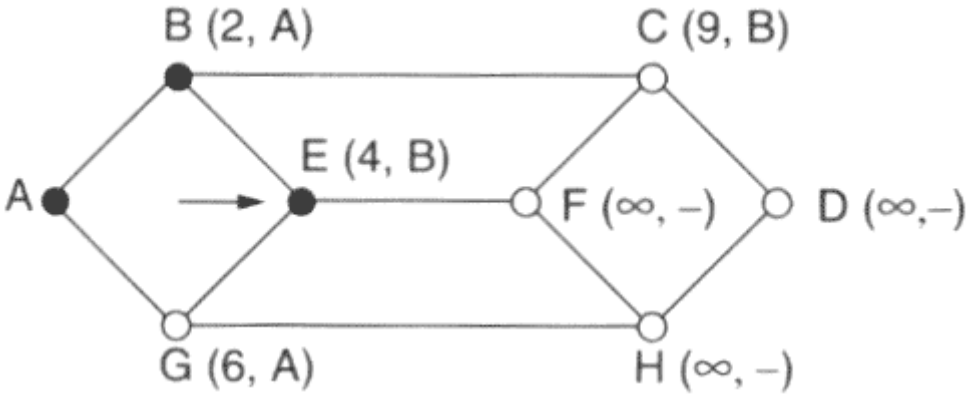


Figura (c)

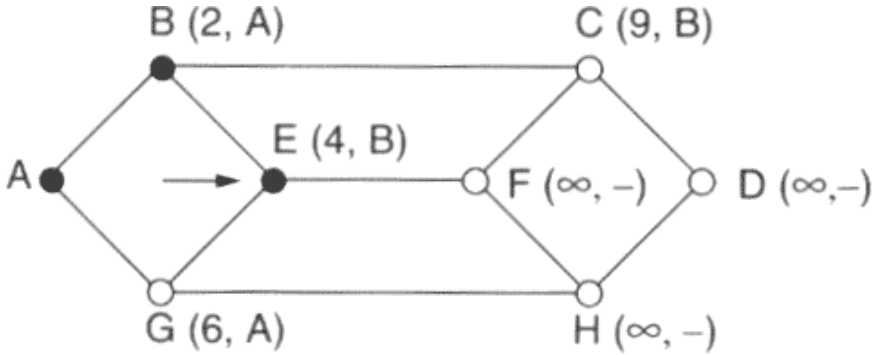


Figura (c)

Observe a Figura (c), em que o nó E foi feito permanente. Suponhamos que houvesse um caminho mais curto do que ABE, por exemplo, AXYZE.

Há duas possibilidades:

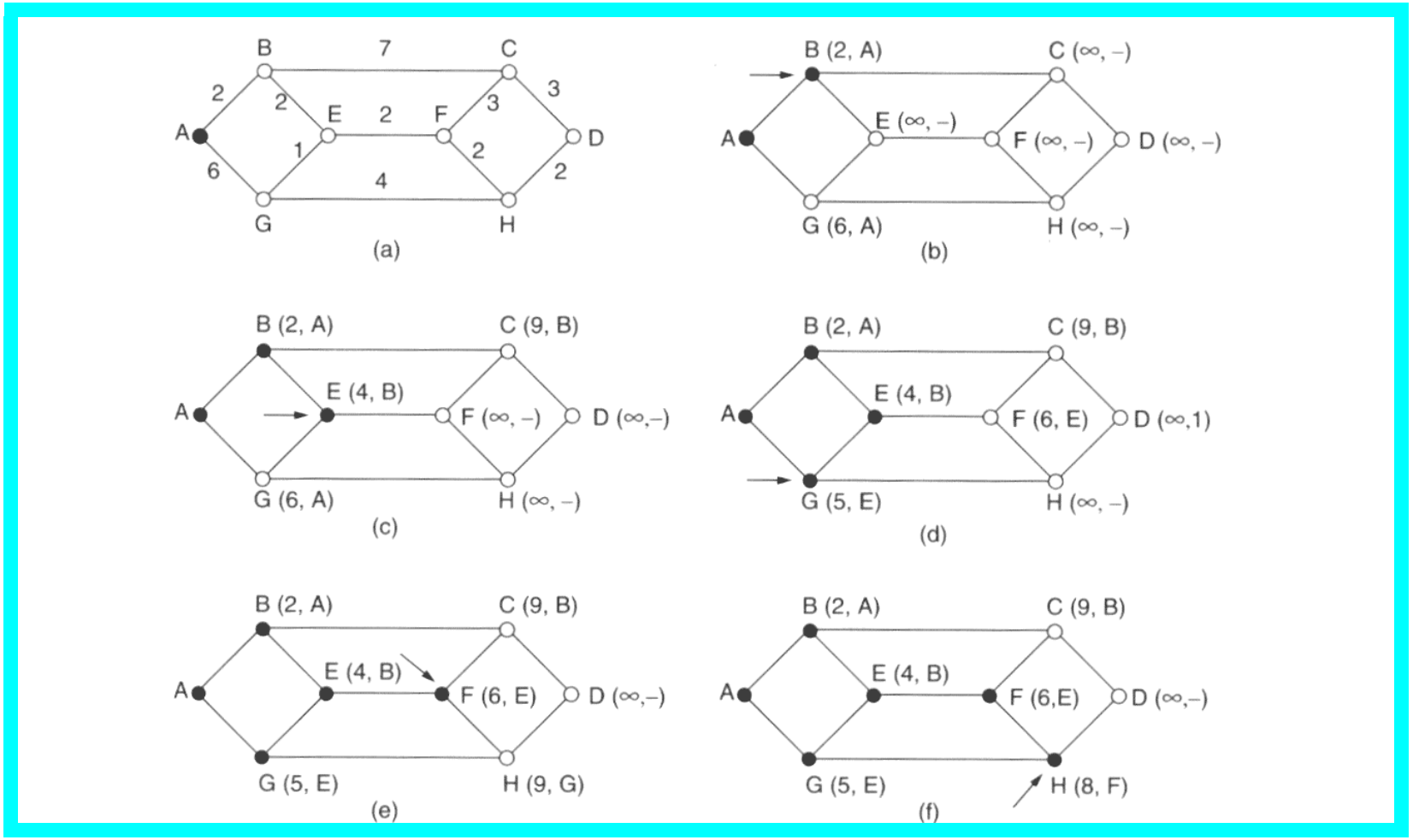
i) o nó Z já foi feito permanente:

- **nó E já foi testado (quando o nó Z foi feito permanente), de tal forma que o caminho AXYZE não escapou da atenção do algoritmo.**

ii) o nó Z ainda não foi feito permanente (o nó Z é um nó provisório):

- **ou o *label* em Z é maior ou igual ao *label* em E, neste caso, AXYZE não pode ser um caminho mais curto do que ABE, ou é menor já tenha sido feito permanente (caso i),**
- **ou o *label* em Z é menor do que em E, caso em que Z e não E se tornará nó permanente primeiro, permitindo que o nó E seja testado a partir de Z.**

As Figuras (d), (e) e (f) mostram os demais passos de execução do algoritmo.


B
7
C
2
2
3
3
E
2
F
A
D
1
2
6
2
4
G
H
(a)
B (2, A)
C (∞, –)
E (∞, –)
A
F (∞, –)
D (∞, –)
G (6, A)
H (∞, –)
(b)
B (2, A)
C (9, B)
E (4, B)
A
F (∞, –)
D (∞,–)
G (6, A)
H (∞, –)
(c)
B (2, A)
C (9, B)
E (4, B)
A
F (6, E)
D (∞,1)
G (5, E)
H (∞, –)
(d)
B (2, A)
C (9, B)
E (4, B)
A
F (6, E)
D (∞,–)
G (5, E)
H (9, G)
(e)
B (2, A)
C (9, B)
E (4, B)
A
F (6,E)
D (∞,–)
G (5, E)
H (8, F)
(f)

Roteamento pelo Algoritmo *Flooding*

- **O algoritmo *Flooding* (inundação) é outro algoritmo estático.**
- **Neste algoritmo, cada pacote que chega é enviado sobre cada linha de saída, exceto aquela pela qual foi recebido.**
- **O algoritmo *Flooding* gera um vasto número de pacotes duplicados.**
- **Na verdade, gera um número infinito de pacotes, a menos que alguma medida seja tomada para amortecer o processo.**

O algoritmo *flooding* não é prático em muitas aplicações, mas tem alguns usos:

- **Em aplicações militares, onde grandes números de roteadores devem ser notificados, a robustez do algortimo pode ser altamente desejável.**
- **Em aplicações de bases de dados distribuídas, muitas vezes é necessário atualizar todas as bases de dados ao mesmo tempo, caso em que o algoritmo *flooding* pode ser útil.**

Medidas que podem ser tomadas para amortecer o processo de inundação:

- **Uma medida é ter um contador de *hops* no cabeçalho de cada pacote, o qual é decrementado a cada *hop*, com o pacote sendo descartado quando o contador chegar a zero.**
- **Idealmente, o contador de *hops* deve ser inicializado com o comprimento do caminho desde a fonte até o destino.**
- **Se o transmissor não sabe esta medida (tamanho do caminho fonte/destino), o contador pode ser inicialiado para o pior caso, ou seja, o tamanho completo da sub-net.**

- **Uma técnica alternativa para amortecer a "inundação" de pacotes é rastrear quais os pacotes que resultaram da inundação, evitando enviá-los uma segunda vez.**
- **Para tanto pode-se fazer com que o roteador fonte coloque um número seqüencial em cada pacote que recebe do *host*.**
- **Cada roteador, então, necessita de uma lista por roteador fonte informando qual a seqüência de números originada daquela fonte já foi vista.**
- **Se um pacote está na lista, então não é *flooded*.**
- **Para evitar que a lista cresça sem limites, cada lista deve ser incrementada por um contador *k*, significando que todos os números seqüenciais até *k* já foram vistos**
- **Quando um pacote chega é fácil verificar se o pacote é uma duplicata; se é, é descartado.**
- **Além disso, a lista completa abaixo de *k* não é necessária, desde que *k* efetivamente a sumariza.**

- **Uma variação do algoritmo *flooding* que é levemente mais prática é chamada de *flooding* seletivo.**
- **Neste algoritmo os roteadores não enviam cada pacote sobre cada linha, apenas sobre aquelas linhas que estão indo aproximadamente na direção certa.**

O algoritmo *flooding* é utilizado como uma métrica a partir da qual todos os demais algoritmos podem ser comparados.
***Flooding* sempre escolhe o menor caminho, porque escolhe cada caminho possível em paralelo.**
Conseqüentemente, nenhum outro algoritmo pode produzir um atraso mais curto (se ignorarmos o *overhead* gerado pelo processo de *flooding* propriamente dito).