# ENCAMINHAMENTO

Ligar duas máquinas de modo a que estas possam comunicar é simples, pois basta assegurar que aquilo que uma transmite seja recebido pela máquina destinatária. Isso pode fazer-se com uma ligação directa. O permitir que uma máquina possa comunicar com outras, em momentos diferentes e a distâncias também distintas, leva à necessidade de um dispositivo que assegure a comunicação, que a encaminhe, para a máquina destinatária. De preferência, as máquinas origem e destino da comunicação apenas se deveriam preocupar com a identificação de quem vem e para quem vai a informação a ser transmitida. O mesmo acontece quando enviamos uma carta, preocupamo-nos em colocar o endereço de destinatário e o remetente, e toda a rede de comunicações dos correios preocupa-se em encaminhar a carta para o destino.

Se eventualmente enviar duas cartas com o mesmo destinatário e remetente, não é imprescindível que ambas percorram exactamente os mesmos caminhos, ou que cheguem na mesma ordem com que as coloquei no marco do correio. O importante, isso sim, é que cheguem ao destinatário, ou caso isso não seja possível, que sejam devolvidas.

Todo o processo de trânsito da correspondência, dos dados a serem transmitidos, é assegurado por diversos algoritmos de encaminhamento.

Um algoritmo é um modo de especificar um conjunto de ideias, de acções com o objectivo de realizar o objectivo pretendido de uma forma sistemática. Garantindo assim uma compreensão do que se pretende efectuar.

São diversos os algoritmos de encaminhamento, contudo têm tabelas de encaminhamento, ou seja, uma referência cruzada de modo a poder decidir por qual das diversas saídas terá de ser enviada a informação que chegou por uma das entradas.

## Algoritmos de Encaminhamento

Os algoritmos podem ser ou não adaptativos. Se o algoritmo é não adaptativo, pode também tomar o nome de **Estático**. Neste caso, as tabelas de encaminhamento são pré-definidas. Este é o algoritmo mais simples de se utilizar e como é fixo, não depende do tráfego da rede ou de qualquer outro

indicador. A tabela de encaminhamento é preenchida pelo gestor da rede e fica assim cristalizada até ele querer.

No caso de serem adaptativos, existem vários tipos de algoritmos, dependendo do modo como são construídos. Podem dividir-se em três grandes grupos: Centralizados, Distribuídos e Isolados.

**Centralizados**

Estes algoritmos necessitam da existência dentro da rede de um centro de controlo de encaminhamento (RCC – Routing Control Center). O RCC recebe periodicamente informação dos nós sobre:

- os vizinhos que estão operacionais
- quantidade de tráfego
- ocupação dos buffers locais.

Depois essa informação é tratada e o melhor caminho entre quais quer dois nós é calculado e as novas tabelas de encaminhamento são difundidas por toda a rede.

Este tipo de algoritmo é utilizado numa rede em que os nós e as linhas vão abaixo frequentemente, o que dificultaria a convergência de outro tipo de algoritmo para a melhor solução, e para redes onde existe uma grande variação de tráfego, podendo assim calcular-se tabelas para as horas de ponta e tabelas para uma quantidade baixa de tráfego.

**Isolados**

Esta situação é oposta à centralizada. Neste tipo de algoritmos, não existe qualquer tipo de informação com os outros nós. O encaminhamento é decidido localmente com informação inferida da própria comunicação conseguindo-se adaptar a alterações de topologia e tráfego. Todos os nós têm o mesmo algoritmo. É utilizado para redes cuja estrutura varia pouco. O encaminhamento é muito rápido e não há na rede informação adicional de controlo para efectuar o encaminhamento.

**Distribuídos**

Este tipo de algoritmo é de todos o mais dinâmico e adapta-se às características do tráfego da rede. Cada nó troca informação acerca do que

sabe da rede, com os seus vizinhos. Em cada nó existe também uma tabela com todos os nós da sub rede. Essa tabela contém dados que de algum modo são uma estimativa da "distância" a esse nó e qual a direcção a seguir. Ou seja, um caminho que ligue dois nós o qual transporta muito tráfego e com baixo ritmo está mais longe do que outro segmento com pouco tráfego e de alto débito, sem contabilizar as distâncias físicas envolvidas, atrasos, custos, e outros indicadores que possam ser úteis para decidir sobre o encaminhamento.

### ALGORITMOS ISOLADOS

**Batata Quente** (Baran 1964) – este algoritmo é o mais simples de todos. Cada nó tenta-se livrar o mais rapidamente possível do pacote como se de uma batata quente se tratasse nas mãos. O pacote é colocado na fila de saída mais curta, sem haver qualquer tipo de noção desta fila ser a que se dirige na direcção certa. Imagine-se a apanhar o primeiro autocarro que vê sem olhar para o número!

**Random Walk** – Este algoritmo coloca o pacote a ser encaminhado numa fila de saída qualquer, sem que haja qualquer tipo de preocupação acerca da quantidade de pacotes à espera de serem transmitidos, ou se essa saída vai porventura na direcção correcta. Idêntico ao anterior, contudo com uma distribuição aleatória sobre a saída a escolher. Este algoritmo pode ser melhorado se introduzirmos um peso, probabilidade diferente das saídas, que seja uma medida da escolha acertada nas melhores direcções.

**Backward Learning** (Baran) – Cada pacote tem um contador de saltos. Cada vez que o pacote passa por um nó, é incrementado o conteúdo desse contador. Quando o pacote chega ao destino, sabe-se a que distância, em saltos, está da origem. Porém esse valor pode ser usado em cada nó que fica assim a conhecer as distâncias em saltos que está dos equipamentos fonte. Guarda apenas o valor mais baixo ficando com o melhor caminho. Assim pode saber qual é a melhor fila de saída quando quer enviar um pacote para esse destinatário.

**Delta Learning** (Rudin 1976) – O custo de cada saída é medido. Periodicamente envia-se para o centro de controlo de encaminhamento (RCC). O RCC escolhe K caminhos distintos entre quaisquer dois nós. Consideram-se distintos os caminhos cuja diferença de custo for inferior a um determinado valor delta δ. Depois é enviado para cada nó os K caminhos calculados e o seu custo associado. A escolha do caminho a utilizar é da responsabilidade do nó. Podendo esta escolha ser aleatória, em função do custo, ou outra razão. A escolha dos valores de K e de δ transformam este algoritmo em centralizado ou isolado.

**Flooding** – Conhecido pelo nome de Inundação, este algoritmo acaba por inundar a rede toda com pacotes porque cada vez que recebe um numa das entradas, repete-o por todas as saídas excepto por onde foi recebido. Embora seja o mais robusto dos algoritmos baixa o rendimento da rede. Porém garante que se houver uma maneira de chegar de um ponto ao outro, o pacote chega lá. Para evitar uma saturação da rede, um afogamento completo, pode fazer-se o seguinte:

- Colocar um contador de saltos para que cada pacote não possa dar mais do que um determinado número de saltos. Cada vez que o pacote entra num nó, o seu valor é decrementado de um, e depois replicado para cada saída. Assim garante-se que os pacotes acabem por desaparecer e chega pelo menos uma cópia ao destinatário caso se saiba o número de saltos máximo p lá chegar.

- Coloca-se um número de sequência para evitar que o pacote ande às voltas e seja repetido vezes sem conta pelo mesmo nó.

- A repetição pode ser feita apenas para as saídas que sejam na direcção do destinatário.

Podem fazer-se combinações destas variantes.

## ALGORITMOS DISTRIBUÍDOS

Um algoritmo diz-se distribuído pelo facto do conhecimento global estar em cada nó que acaba por ter uma noção do que se passa na rede e decide o melhor caminho usando informação que lhe chega pelos próprios pacotes.

**Distributed Routing (Distance Vector Routing)**

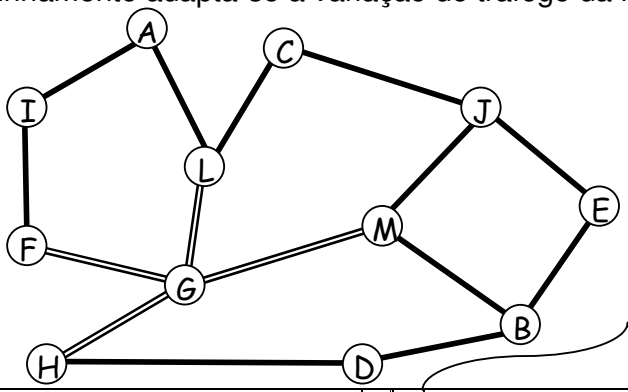
Cada nó contém uma tabela com todos os nós da rede.

Por cada entrada de cada nó está associado o custo de chegar a esse nó.

Essas tabelas são trocadas entre nós vizinhos.

Somando as tabelas dos vizinhos ao custo de transmissão entre vizinhos produz-se a nova tabela.

Deste modo, as alterações de custos são lentamente propagadas por toda a rede. O encaminhamento adapta-se à variação de tráfego da rede.




Distance Vector

| | Tabelas recebidas destes nós | | | | Nova Tabela para o nó G | | |
|---|---|---|---|---|---|---|---|
| Destino | F | H | L | M | Destino | Custo | Saída |
| A | 12 | 32 | 7 | 12 | A | 17 | L |
| B | 10 | 5 | 8 | 11 | B | 11 | H |
| C | 4 | 17 | 10 | 14 | C | 16 | F |
| D | 30 | 21 | 11 | 25 | D | 21 | L |
| E | 15 | 9 | 6 | 30 | E | 15 | H |
| F | 0 | 16 | 18 | 14 | F | 12 | F |
| G | 8 | 13 | 22 | 6 | G | 0 | – |
| H | 6 | 0 | 12 | 7 | H | 6 | H |
| I | 12 | 34 | 13 | 9 | I | 23 | M |
| J | 5 | 21 | 15 | 11 | J | 17 | F |
| L | 4 | 12 | 0 | 21 | L | 10 | L |
| M | 12 | 6 | 23 | 0 | M | 12 | H |
| Custo | 12 | 6 | 10 | 14 | Valor medido ao receber as tabelas | | |

Análise do Algoritmo para o Nó G

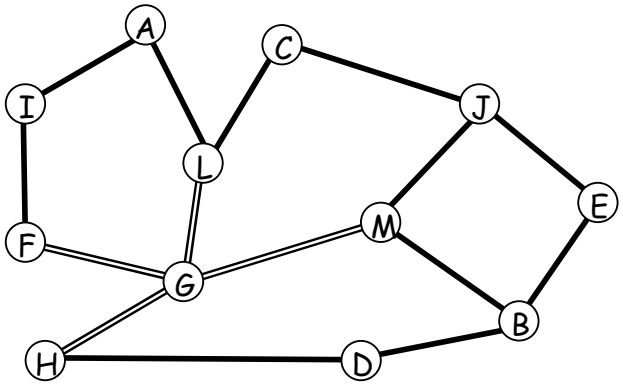
## Análise do Algoritmo para o Nó G

| | Tabelas recebidas dos nós | | | | Novos custos do Nó G | | | | Escolhas | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Destino | F | H | L | M | F | H | L | M | 1ª | | 2ª | | 3ª | |
| A | 12 | 32 | 7 | 12 | 24 | 38 | 17 | 26 | 17 | L | 24 | F | 26 | M |
| B | 10 | 5 | 8 | 11 | 22 | 11 | 18 | 25 | 11 | H | 18 | L | 22 | F |
| C | 4 | 17 | 10 | 14 | 16 | 23 | 20 | 28 | 16 | F | 20 | L | 23 | H |
| D | 30 | 21 | 11 | 25 | 42 | 27 | 21 | 39 | 21 | L | 27 | H | 39 | M |
| E | 15 | 9 | 6 | 30 | 27 | 15 | 16 | 44 | 15 | H | 16 | L | 27 | F |
| F | 0 | 16 | 18 | 14 | 12 | 22 | 28 | 28 | 12 | F | 22 | H | 28 | H |
| G | 8 | 13 | 22 | 6 | 20 | 19 | 32 | 20 | 0 | – | 0 | – | 0 | – |
| H | 6 | 0 | 12 | 7 | 18 | 6 | 22 | 21 | 6 | H | 18 | F | 21 | M |
| I | 12 | 34 | 13 | 9 | 24 | 40 | 23 | 23 | 23 | M | 23 | L | 24 | F |
| J | 5 | 21 | 15 | 11 | 17 | 27 | 25 | 25 | 17 | F | 25 | L | 25 | M |
| L | 4 | 12 | 0 | 21 | 16 | 18 | 10 | 35 | 10 | L | 16 | F | 18 | H |
| M | 12 | 6 | 23 | 0 | 24 | 12 | 33 | 14 | 12 | H | 14 | M | 24 | F |
| Custo | 12 | 6 | 10 | 14 | Valor medido ao receber as tabelas | | | | | | | | | |

**Multipath Routing (Bifurcated Routing)**

Procede-se como no caso do Distributed Routing.

Aproveitam-se K caminhos possíveis (neste exemplo K=3) para cada nó. Estes caminhos devem ser disjuntos.




Atribui-se a cada entrada por nó na tabela um valor que determine a probabilidade de escolha de cada caminho para esse nó.

Gera-se um valor aleatório, e em função dele escolhe-se a saída. Segue um exemplo para o nó G.

| | Tabelas recebidas dos nós | | | | Escolhas | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Destino | F | H | L | M | 1ª | | 2ª | | 3ª | |
| A | 12 | 32 | 7 | 12 | 0,61 | L | 0,32 | F | 0,07 | M |
| B | 10 | 5 | 8 | 11 | 0,52 | H | 0,34 | L | 0,14 | F |
| C | 4 | 17 | 10 | 14 | 0,45 | F | 0,35 | L | 0,20 | H |
| D | 30 | 21 | 11 | 25 | 0,55 | L | 0,34 | H | 0,11 | M |
| E | 15 | 9 | 6 | 30 | 0,66 | H | 0,25 | L | 0,09 | F |
| F | 0 | 16 | 18 | 14 | 0,40 | F | 0,35 | H | 0,25 | H |
| G | 8 | 13 | 22 | 6 | 0 | – | 0 | – | 0 | – |
| H | 6 | 0 | 12 | 7 | 0,61 | H | 0,23 | F | 0,16 | M |
| I | 12 | 34 | 13 | 9 | 0,45 | M | 0,33 | L | 0,22 | F |
| J | 5 | 21 | 15 | 11 | 0,35 | F | 0,33 | L | 0,32 | M |
| L | 4 | 12 | 0 | 21 | 0,42 | L | 0,25 | F | 0,33 | H |
| M | 12 | 6 | 23 | 0 | 0,53 | H | 0,30 | M | 0,17 | F |
| Custo | 12 | 6 | 10 | 14 | Valor medido ao receber as tabelas | | | | | |

**Broadcast Routing**

Existem diversos modos de fazer difusão.

- A fonte envia um pacote distinto para cada um dos destinatários. Este método é muito pesado e não é aconselhável. Exige também que a fonte conheça todos os endereços de destino.

- A fonte envia um pacote apenas e procede-se à técnica de Flooding. Também provoca um grande peso na rede.

- Multidestination Routing. Segundo este método, cada pacote tem uma lista de todos os destinatários. O nó envia para as saídas correspondentes e actualiza os destinos. Deste modo apenas envia pelos melhores caminhos e os pacotes vão ficando cada vez mais pequenos.

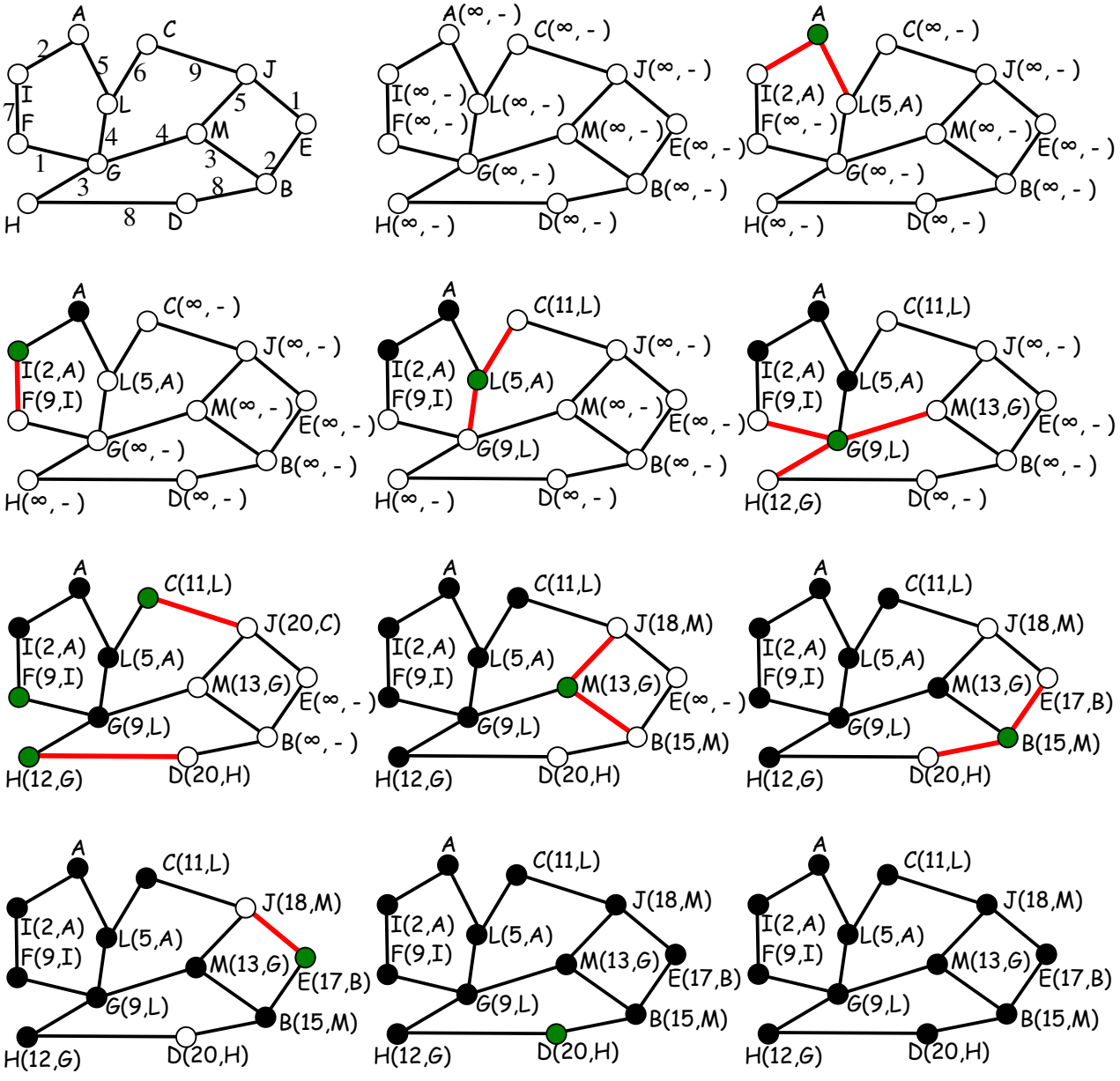
- Cada nó da rede utiliza a Spanning Tree do nó que originou o broadcast e deste modo garante-se que todos os nós recebem apenas uma cópia do pacote e enviam-no apenas pelas saídas adequadas. Esta informação existe no nó quando o algoritmo utilizado é o Link State Routing, porque cada nó tem conhecimento directo de toda a rede, o que não acontece com o Distance Vector Routing em que apenas se sabem dados directamente dos vizinhos. Outro modo seria o envio da Spanning Tree, quer no próprio pacote quer por outro método.

- Reverse Path Forwarding. Segundo este método parte-se do pressuposto que o melhor caminho entre o nó que originou o broadcast e aquele que entretanto o recebeu coincide com aquele que este nó utiliza para enviar pacotes para o nó que originou o broadcast. Neste caso, se o pacote recebido coincide com a linha por onde seria enviado, ele é repetido por todas as outras linhas excepto essa de onde recebeu. Caso contrário, o pacote é um duplicado e não é reenviado. Este algoritmo é simples de implementar e tem a vantagem de minimizar os pacotes duplicados.

**Shortest Path Routing (Dijkstra 1959)**

O algoritmo de Dijkstra aplicado à procura do caminho mais curto entre um ponto inicial e qualquer outro da rede.

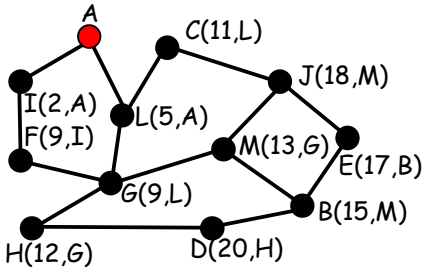
- Marcam-se todos os nós com letras e as ligações com o peso associado.
- Marcar todos os nós com distância infinita (∞) e nó desconhecido (-).
- Escolhe-se um nó para fonte, e marca-se como permanente ● .
- Propaga-se a distância aos outros nós, actualizando os nós com a nova distância e o nó fonte (caso seja inferior à que já tinha).
- Escolhe-se o nó não permanente ○ com a distância menor, marca-se como permanente ● , assume-se que é a fonte e repete-se o ponto anterior.

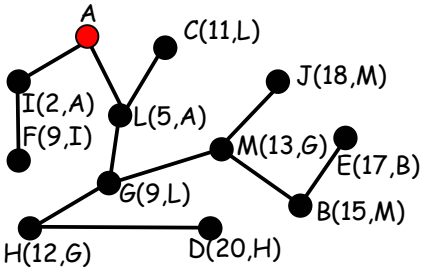


**Spanning Tree**

Uma árvore é um circuito aberto em que cada nó só é visitado uma única vez, ou seja só há um caminho possível entre dois nós de uma árvore. O mesmo se passa para um subconjunto da árvore.

Um modo possível de fazer a Spanning Tree consiste em utilizar um algoritmo como o do Dijkstra e posteriormente, utilizando como raiz da árvore o nó inicial, proceder à limpeza dos troços não utilizados.



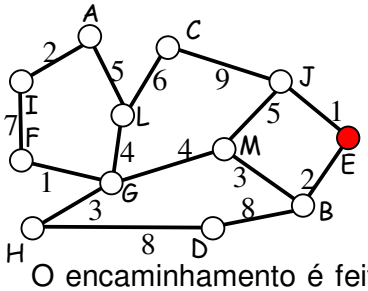



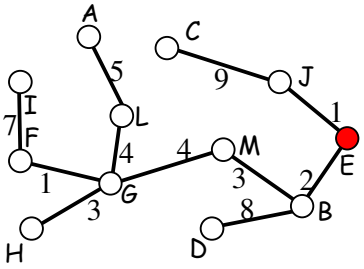

**Optimal Routing**

Este algoritmo parte do seguinte princípio: Se o nó B está no caminho óptimo entre A e C, então o caminho óptimo de B para C é esse mesmo.

O conjunto de caminhos óptimos de todos os nós para um mesmo de destino, forma uma árvore com raiz no destino. (**Sink Tree**).

Por exemplo, fazendo o destino o ponto E




O encaminhamento é feito utilizando a Sink Tree, de modo que qualquer pacote é enviado apenas pelos ramos da rede que pertencem à árvore cujo nó de destino é a raiz. (neste exemplo, o ponto E)

Evita que os pacotes sejam repetidos duas ou mais vezes pelo mesmo nó.

No caso de Broadcasting

- diminui o número total de pacotes a difundir.
- se o pacote recebido veio do caminho óptimo desse nó e o próprio, então propaga-se por todas as saídas.
- se o pacote não entrou pelo caminho óptimo, então é uma repetição, e deve ser ignorado. (Reverse path forwarding)

**Hierarchical Routing**

As tabelas de encaminhamento aumentam proporcionalmente com o aumento do número de nós da rede. Aumenta também os recursos necessários para resolver a questão do encaminhamento os quais são a memória de cada nó, o tempo de CPU e a largura de banda entre nós que determinam a eficácia do encaminhamento. Também se inclui o algoritmo a usar.

Caso o número de nós seja muito grande, é necessário utilizar um algoritmo hierárquico como é o caso do maior dos exemplos que é a rede telefónica.

O algoritmo hierárquico divide todo o espaço da rede em regiões. Dentro de cada região, o encaminhamento é bem conhecido, mas nada se sabe acerca de informação fora dessa região. Duas redes distintas são encaradas como regiões diferentes, de modo que uma não sabe nada acerca da estrutura interna da outra. A divisão hierárquica pode ser feita em diversos níveis, usando um termo para cada um deles, como é o exemplo de regiões, agregados, zonas, grupos, aldeias, vilas.

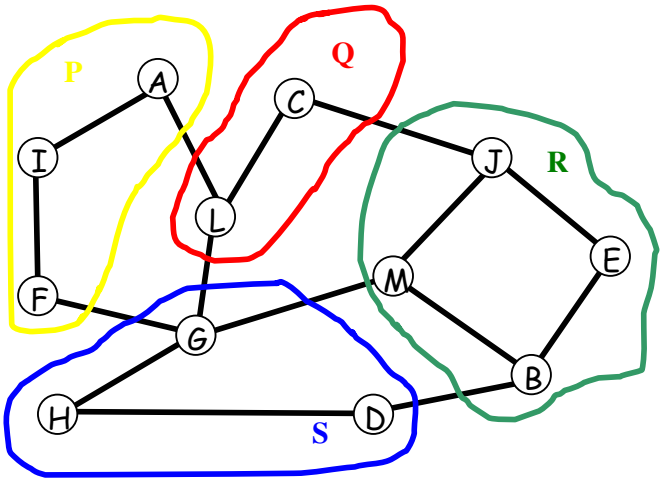
Um exemplo da hierarquia a dois níveis é o caso do TCP/IP cujo endereço está dividido entre um endereço de rede e um de equipamento. O nível hierárquico aumenta com a utilização de sub-networking. Enquanto o pacote é encaminhado, apenas o endereço de rede é que é importante, depois de chegar à gateway da rede de destino, aí é que se olha para o endereço do equipamento.

A tabela de cada nó fica reduzida porque todos os nós que pertencem à mesma rede ficam condensados num único nó, que identifica a rede. A distância em saltos é efectuada supondo que cada rede é um nó.

Coloca-se uma nova questão, a de quantos níveis hierárquicos a utilizar. Kamoun e Kleinrock (1979) descobriram que o número óptimo de níveis para uma rede com N nós é de ln (N)

Como não se sabe nada acerca da estrutura interna das redes vizinhas, pode acontecer que um nó que esteja perto do ponto de vista físico passe agora com o uso deste algoritmo a estar mais distante, numa rede acessível apenas através de uma outra intermédia. É o caso do exemplo da figura, em que o nó A para aceder aos nós C, L e J, necessita de passar pela rede S.

**Hierarchical Routing**




| Tabela habitual do nó A | | |
|---|---|---|
| Destino | Saída | Saltos |
| A | – | – |
| B | L | 5 |
| C | L | 2 |
| D | L | 4 |
| E | L | 4 |
| F | I | 2 |
| G | L | 2 |
| H | L | 3 |
| I | I | 1 |
| J | L | 3 |
| L | L | 1 |
| M | L | 3 |

| Tabela Hierárquica para o nó A | | |
|---|---|---|
| Destino | Saída | Saltos |
| A | – | – |
| F | I | 2 |
| I | I | 1 |
| **Q** | **Q** | 1 |
| **R** | I | 4 |
| **S** | I | 3 |

**Link State Routing**

O protocolo de encaminhamento RIP utilizado na ARPANET desde o seu início em 1969 até 1979 baseava-se no algoritmo de Distance Vector Routing. A métrica utilizada era uma medida do atraso das filas de espera dos pacotes a serem transmitidos, e todas as saídas tinham o mesmo ritmo binário de 56Kbps. Ao serem adicionadas linhas de 230Kbps e 1,544Mbps era necessário introduzir um peso sobre a largura de banda, mas embora isto tivesse sido feito, o algoritmo demorava muito a convergir, principalmente quando um nó ficava fora de serviço. Assim, foi substituído em 1979 pelo algoritmo de Link State Routing.

Hoje existem diversas variantes deste algoritmo.

Algoritmo:

1. Descobrir os vizinhos e os seus endereços de rede
2. Medir o atraso, custo, entre o próprio nó e cada vizinho
3. Construir um pacote com toda a informação conhecida
4. Enviar esse pacote de sinalização para todos os nós da rede
5. Calcular o caminho mais curto para cada nó da rede (Dijkstra)

A grande diferença entre este algoritmo e o Distance Vector (distributed routing) é o de cada nó enviar o que sabe para todos os nós e não só para aqueles que lhe são vizinhos.

**Descobrir os vizinhos e os seus endereços de rede**

Para saber quem são os vizinhos, cada nó ao ser iniciado envia por cada linha de saída um pacote tipo HELLO para se dar a conhecer. Cada nó ao receber um pacote HELLO deve responder com a sua identificação. Cada nó deve estar identificado de um modo único e universal, para que se possam distinguir, ou seja um nó tem um identificador e cada identificador determina um e um só nó. Caso haja dois ou mais nós ligados à mesma rede local, esta é introduzida nas tabelas como se de um nó se tratasse.

**Medir o atraso, custo, entre o próprio nó e cada vizinho**

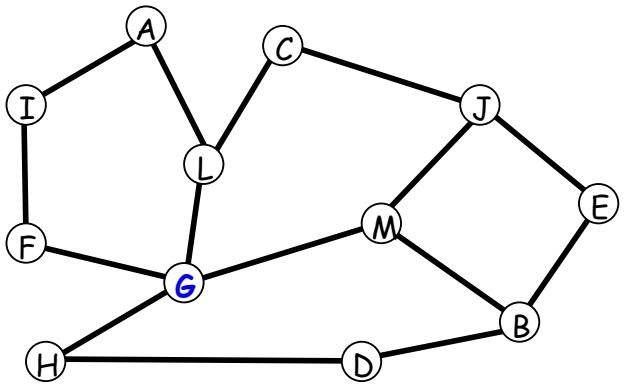
Para ter uma avaliação sobre o custo de uma ligação entre o nó e um vizinho, cada nó envia um pacote tipo ECHO que é devolvido por cada um dos nós. Para evitar confusões entre uma emissão e uma recepção de um pacote ECHO, cada um deles é identificado como sendo um pedido (request) ou resposta (answer).

Por cada pedido de ECHO que se envia coloca-se um contador que é parado quando se receber o ECHO de resposta. Caso o contador comece a contar assim que se coloca na lista de saída, significa que o custo, o valor, terá uma medida também do atraso da linha de saída; caso o contador comece a contar assim que o pacote é transmitido, apenas se quer ter em conta o atraso da comunicação. O mesmo sucede para o momento em que o contador é parado aquando da recepção do ECHO de resposta.

Ao dividir o valor do contador por dois tem-se a noção do atraso, custo, da ligação. É habitual fazer-se a média dos tempos medidos de diversos pacotes para o mesmo vizinho.

**Construir um pacote com toda a informação conhecida**

O pacote construído com toda a informação que o nó sabe acerca dos seus vizinhos toma o nome de Link State Packet. O pacote contém um cabeçalho com a identificação do nó, um número de sequência e um campo para indicar a idade do pacote, e segue-se a lista de todos os vizinhos e os tempos de atraso medidos entre eles e o nó identificado.




| G | |
|---|---|
| Sequência | |
| Idade | |
| L | 4 |
| F | 1 |
| M | 4 |
| H | 3 |

**Enviar esse pacote de sinalização para todos os nós da rede**

O Link State Packet é enviado para cada nó da rede, contudo esta tarefa pode ser afectada por diversos factores perturbadores. Á medida que os nós vão recebendo os pacotes, alteram as suas configurações, e de um modo global cada nó tem uma noção da topologia que está incompleta e por isso diferente de qualquer outro nó. As topologias e pesos das ligações só ficam completas quando todos os nós receberem informação de todos os outros nós. Até lá a convergência é difícil porque podem surgir situações inconsistentes, ilhas fechadas (loop), nós inacessíveis, e outras.

A melhor forma de realizar isto é através da utilização do algoritmo de Flooding. Para evitar que cada nó difunda pacotes repetidos coloca-se o campo de Sequência e só os pacotes com um valor diferente do número de sequência é que são enviados por todas as saídas. Para evitar que não se saiba a sequência correcta enviada de um determinado nó, é associado um contador no próprio pacote Link State de modo que por cada unidade de tempo pré determinada esse valor é decrementado e assim pode-se ignorar a sequência recebida e aceitar a próxima como nova e correcta. Isto é necessário para o caso de um nó ser reiniciado após algum tempo em funcionamento. O seu número de sequência iria sofrer uma descontinuidade.

Como a um determinado nó podem chegar pacotes de um outro nó distante por qualquer uma das suas entradas e cada uma delas distar tempos diferentes do nó emissor, pode esperar-se um pouco assim que se recebe um pacote de Link State para verificar se chega outro com um número de sequência mais recente, e só então processar e difundir a informação.

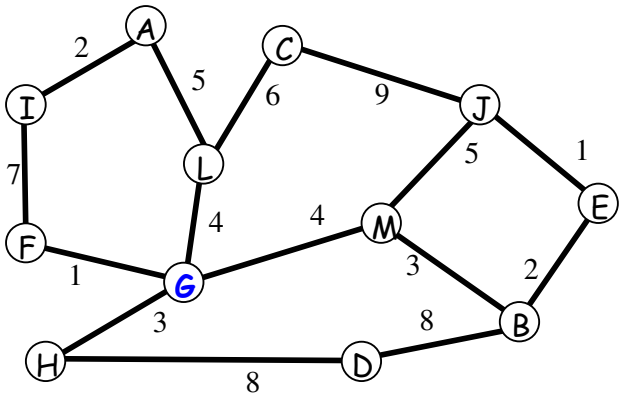
Em cada nó existe uma estrutura que mantém o estado da informação recebida e da que foi enviada. Adiciona-se assim um conjunto de flags a cada Link State de cada nó.

Para evitar erros entre cada um dos nós é enviado um acknowledge por cada Link State recebido de cada nó vizinho.

**Calcular o caminho mais curto para cada nó da rede (Dijkstra)**

Constroi-se a tabela de encaminhamento com a informação recebida.

**Link State Routing**




| Nó | Seq | Idade | Flags de Envio | | | | Flags de ACK | | | | Lista dos vizinhos | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | F | L | H | M | F | L | H | M | | | | | | |
| A | 12 | 24 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | I | 2 | L | 5 | | |
| B | 15 | 24 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | E | 3 | M | 3 | D | 8 |
| C | 14 | 23 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | L | 6 | J | 7 | | |
| D | 12 | 23 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | B | 8 | H | 8 | | |
| E | 11 | 25 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | J | 1 | B | 4 | | |
| F | 16 | 25 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | I | 7 | **G** | 1 | | |
| H | 14 | 24 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | **G** | 4 | D | 9 | | |
| I | 14 | 23 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | F | 7 | A | 3 | | |
| J | 12 | 23 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | E | 2 | M | 5 | C | 9 |
| L | 13 | 10 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | A | 6 | C | 5 | **G** | 4 |
| M | 16 | 22 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | **G** | 4 | J | 7 | B | 3 |

0 – indica que estava a 1 e que o ACK foi enviado.

7 – indica que é um valor diferente daquele indicado na topologia apenas porque cada nó tem a sua própria percepção do atraso da ligação.

**G** – indica que a estrutura apresentada é do nó G.

**Multicast Routing**

Este tipo de difusão é utilizado quando se quer enviar o mesmo pacote de um equipamento terminal para diversos destinos. O número de destinatários pode ser pequeno ou grande comparado com a totalidade da rede. Um modo de o fazer seria a de utilizar a técnica do Broadcast e depois cada destinatário apenas usaria a informação caso fosse para si. Este método para além de pesado também faz com que a informação que se destina a um determinado grupo de utilizadores chegue a locais desconhecidos fora desse mesmo grupo, o que pode não ser conveniente.

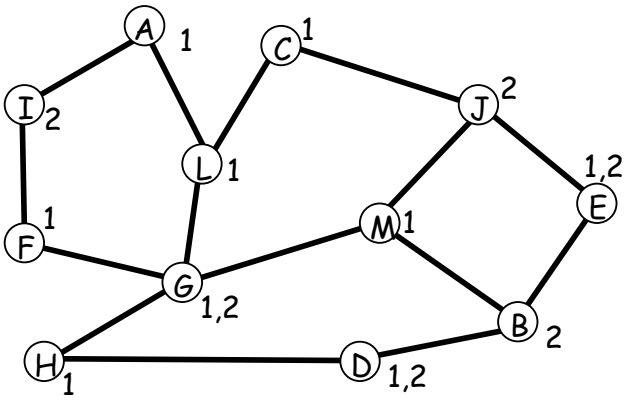
Para efectuar o multicasting é necessário haver uma gestão de grupos de modo a poder criar e desfazer grupos, acrescentar e remover processos dos equipamentos terminais desses grupos. Qualquer que seja a gestão a utilizar, ou o modo como esta é feita, é contudo necessário que a inclusão ou exclusão de um grupo seja indicada aos nós aos quais esses equipamentos terminais estão ligados. Caso não se indique essa acção a responsabilidade tem de cair sobre o nó que terá de inquirir, a determinados intervalos de tempo, se houve alguma alteração nos grupos, ou por exemplo apenas quando recebe um pacote para um grupo. Cada nó, ao saber quais dos processos dos seus equipamentos terminais é que pertencem a cada grupo difunde essa informação pelos restantes nós da rede.

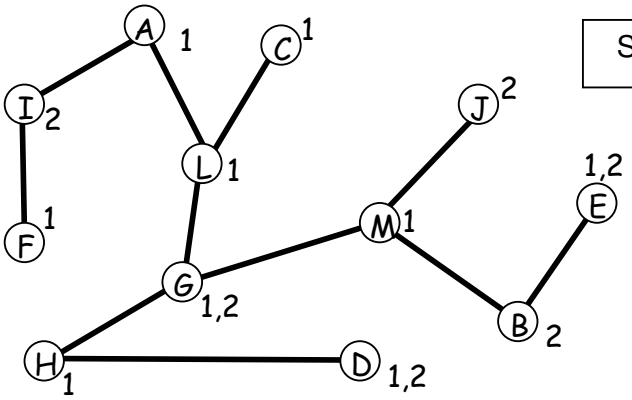
Utilizando a Spanning Tree com a identificação dos grupos nos nós, é possível gerar outras árvores a partir dessa apenas com os ramos que ficam depois de se apararem aqueles troços que não são necessários por não levarem a membros do grupo. Os pacotes de multicast são enviados apenas nos troços que permaneceram.

Utilizando o algoritmo do Distance Vector Routing, pode ser utilizado uma estratégia diferente para aparar os troços. Utiliza-se o Reverse Path Forwarding. Sempre que um nó recebe um pacote de multicast e não tem nenhum processo utilizador a quem entregar nem outros nós ligados, envia para a linha de onde veio o pacote um outro do tipo PRUNE, para que esse nó seja retirado da árvore (se possível). Caso um nó tenha recebido em todas as suas entradas um pacote tipo PRUNE, também pode responder com o mesmo tipo de pacote, e desse modo toda a sub rede a partir dele é eliminada.

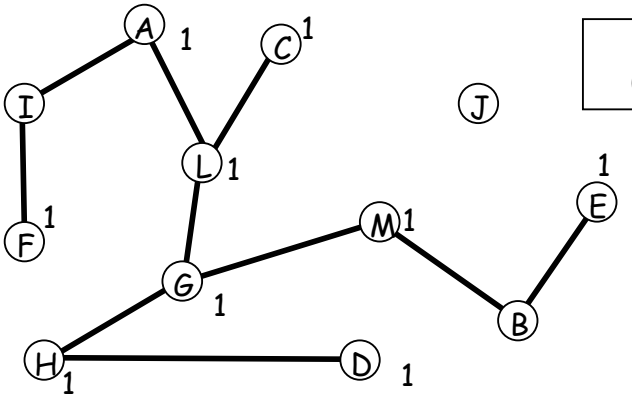
A desvantagem deste tipo de difusão é que é necessário construir uma Pruned Spanning Tree para cada grupo, e isso é difícil de manter. Numa rede com um valor médio de g grupos cada um com m membros, implica guardar em cada grupo gm Spanning Trees aparadas.

Utilizando Core-based trees melhora-se esta versão anterior do Distance Vector com Reverse Path Forwarding dos pacotes PRUNE. Utiliza-se apenas uma Pruned Spanning Tree por grupo que é situada no nó que está mais perto do centro da árvore desse grupo. Todos os membros do grupo enviam para esse nó central o Core, os pacotes que se destinam ao grupo, e esse nó depois difunde-os pela árvore que tem guardada.

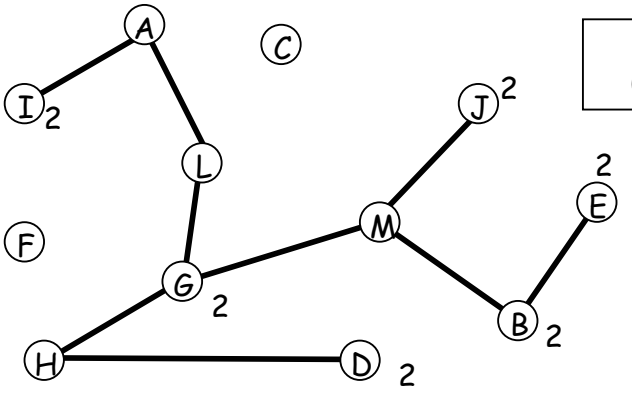



Spanning Tree para o nó A




Pruned Spanning Tree (aparada) para o grupo 1




Pruned Spanning Tree (aparada) para o grupo 2

**O Algoritmo de Encaminhamento deve ser**

Correcto – O resultado do algoritmo deve dar uma solução possível e de acordo com o critério utilizado para minimizar os custos pretendidos, como por exemplo o caminho mais curto.

Simples – A execução do algoritmo deve produzir resultados o mais rápido possível. Um algoritmo simples não se mede em tamanho, mas no modo como ele é concretizado. Um algoritmo fácil de se entender é também simples porque ajuda a conferir se o resultado está correcto. Cada nó da rede pode ter de efectuar o mesmo algoritmo, ou então apenas um e este enviar o resultado para todos os outros. Depende do tipo de algoritmo.

Robusto – Um algoritmo diz-se robusto se ele for capaz de funcionar correctamente apesar de surgirem imprevistos e contudo produzir resultados correctos. Numa rede com muitos nós espera-se que ela funcione horas a fio mas é possível que um nó vá a baixo e volte de novo acima. Outras situações como a alteração da topologia por adição ou remoção de ligações ou equipamento também afectam toda a rede e o resultado do encaminhamento. A quantidade de tráfego que circula na rede e as variações abruptas que este possa ter são condições esperadas e que podem ser consideradas no próprio algoritmo.

Estável – A complexidade da rede ou do algoritmo devem ter como resultado uma ou mais soluções, mas essas soluções devem manter-se caso se mantenham as condições iniciais. Há algoritmos que não convergem em certas situações independentemente do tempo que possam levar a correr, e isso não é aceitável. Um algoritmo estável chega a uma condição de equilíbrio e permanece lá.

Justo – Há certas soluções que para beneficiarem certas condições ou restrições pedidas acabam por prejudicar o encaminhamento de outros nós. Não seria aceitável que o algoritmo colocasse de parte nós ou desse percursos demasiado longos para beneficiar um ou outro nó ou ligação.

Óptimo – A solução óptima é aquela que se enquadra dentro das restrições impostas e que procura ser justa para com todos. Minimizar atrasos, maximizar os fluxos de saída pode ser contraditório uma vez que acaba por haver filas para guardar os pacotes. Há assim um compromisso a chegar em que a função de minimização do algoritmo tem diversos pesos para todos os critérios que pretendem ser contabilizados.

Associado a cada ligação há a métrica, ou seja o valor com que avaliamos essa ligação. A Métrica pode ser considerada como o número de saltos que é necessário fazer entre dois nós (Hop). Pode ser o tempo médio de atraso dos pacotes entre nós vizinhos. Pode ser a distância entre dois nós vizinhos, ou a qualidade da ligação. Ou um outro peso qualquer que sirva de referência para essa ligação e susceptível de ser comparado com outro peso de qualquer outra ligação. É sobre este valor que os algoritmos de encaminhamento vão decidir acerca dos caminhos a utilizar.

## Comentários acerca do Encaminhamento – Routing

Existem diversos algoritmos de encaminhamento e muitos deles baseiam-se nos que aqui estão descritos. Por exemplo, o Shortest Path First, que pode ser concretizado com o algoritmo de Dijkstra, é também designado de OSPF, Open Short Path First. O termo Open, refere-se ao sistema ser aberto do ponto de vista de acesso a terceiros e ter diversas implementações distintas. Quando utilizado para enviar pacotes de multicast, por exemplo numa rede que envia rádio ou vídeo para muitos utilizadores, o protocolo toma o nome de MOSPF (Multicast Open Short Path First).

O Distributed Routing, utiliza em cada nó uma tabela que contem a saída preferida e o peso associado quando se pretende enviar um pacote para qualquer outro nó da rede. A essa tabela pode dar-se o nome de vector, Distance Vector. Entenda-se o termo distância como sendo o peso da ligação. Muitos algoritmos podem tomar nomes diferentes apenas pelo facto de terem pequenas diferenças com este Distributed Routing. Este algoritmo é também conhecido por DVR (Distance Vector Routing). Este algoritmo esteve em uso inicialmente na ARPANET e ficou conhecido no protocolo TCP/IP como RIP (Routing Information Protocol). Quando utilizado essencialmente como base do protocolo de multicasting, toma o nome de DVMRP (Distance Vector Multicast Routing Protocol).

Estes algoritmos podem ainda ser analisados do modo como se comportam dinamicamente. Ou seja, o que acontece quando um nó é ligado ou desligado, ou um troço danificado ou restaurado. A maior parte das vezes, apercebemo-nos de que são as boas notícias que se propagam rapidamente, porque essa acção é activa, parte da iniciativa do nó que ficou activo, ou do troço que ficou operacional. As más notícias porém são lentas, porque são passivas, ou seja elas vão-se sabendo por uma difusão em que o peso da ligação aumenta gradualmente até um máximo considerado infinito, ou seja ligação em aberto. Para evitar este aumento a valores muito elevados, coloca-se esse máximo com um valor que seja igual ao maior peso válido na rede somado de um.

# Assimilação de Conceitos

- Link State Routing
- Hierarchical Routing
- Protocolos de Routing Multicast
- Distance Vector Multicast Routing Protocol (DVMRP)
- Internet Group Management Protocol (IGMP)
- Multicast Open Shortest Path First (MOSPF)
- Protocol Independent Multicast (PIM)

# Para Aprofundar

- Algoritmo de Dijkstra
- Routing for Mobile Hosts
- Routing for Mobile Hosts and Routers