Roteamento pelo Estado do *Link*
Link State Routing

- **É um algoritmo de roteamento dinâmico.**
- **Variantes do algoritmo *Link State* são largamente utilizadas.**
- **Substituiu o algoritmo *Distance Vector Routing* a partir de 1979, na ARPANET.**

A substituição ocorreu devido a dois problemas apresentados pelo algoritmo *Distance Vector*:

- **O algoritmo não leva em conta a largura de banda da linha, ao escolher as rotas. Como inicialmente todas as linhas eram de 56 kbps, não considerar a largura de banda não era um problema. No entanto, após algumas linhas terem sido aumentadas para 230 kbps e outras para 1544 Mbps, este fator passou a ter importância.**
- **O algoritmo freqüentemente demora muito para convergir.**

As etapas básicas do algoritmo *Link State* podem ser enunciadas de forma simples:

1. Descobrir quem são os vizinhos e aprender os endereços dos vizinhos na rede.
2. Medir o atraso ou custo para cada um de seus vizinhos.
3. Construir um pacote contendo toda a informação recentemente adquirida.
4. Enviar este pacote a todos os outros roteadores.
5. Computar o caminho mais curto a cada outro roteador.

A topologia completa e todos os atrasos são experimentalmente medidos e distribuídos para cada roteador.

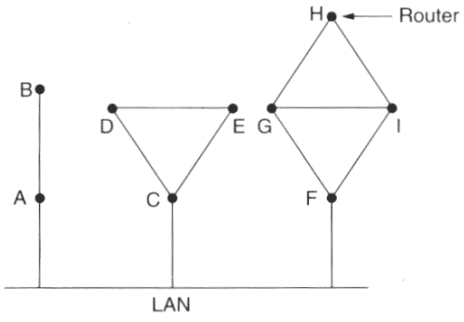
Então, o algoritmo de Dijkstra pode ser usado para determinar o caminho mais curto a cada outro roteador.

1. Descobrir os Vizinhos e Aprender os Endereços dos Vizinhos.

- **Quando um roteador é inicializado, sua primeira tarefa é aprender quem são seus vizinhos.**
- **Esta tarefa é executada enviando um pacote especial chamado HELLO sobre cada linha ponto-a-ponto.**
- **O roteador no outro extremo da linha envia de volta um *reply* informando quem é.**

As informações de identificação dos roteadores (nomes) precisam ser globalmente únicos porque, quando um roteador distante escuta mais tarde que três roteadores são todos conectados a F, é essencial que ele possa determinar se os três roteadores se referem ao mesmo F.

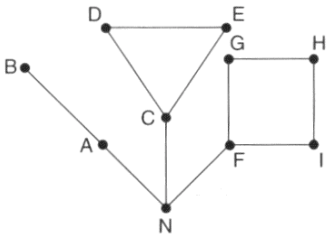
Quando dois ou mais roteadores são conectados por uma LAN, a situação é levemente mais complicada.



(a)

A Figura (a) ilustra uma LAN a que 3 roteadores A, C e F são diretamente conectados, cada um destes roteadores sendo conectado a um ou mais roteadores adicionais.

Uma forma de modelar a LAN é considerá-la propriamente como um nó, como mostrado na Figura (b).
Na Figura (b) foi introduzido um nó artificial N, ao qual A, C e F são conectados. A forma de ir de A a C é representada pelo caminho ANC.



(b)

2. Medir o Custo da Linha.

- **O algoritmo de roteamento *Link State* requer que cada roteador saiba (ou tenha uma razoável estimativa) o atraso relativo a cada um de seus vizinhos.**
- **A forma mais direta de determinar este atraso é enviar um pacote especial chamado ECHO sobre a linha, que o roteador no outro extremo enviará de volta imediatamente.**
- **Medindo o tempo de ida e volta e dividindo por dois, o roteador que envia o pacote ECHO adquire uma estimativa razoável do atraso. Para obter resultados ainda melhores, o teste pode ser conduzido várias vezes, e tomada a média sobre os valores obtidos.**

A medida do tempo de ida e volta pode ser feita de duas formas:

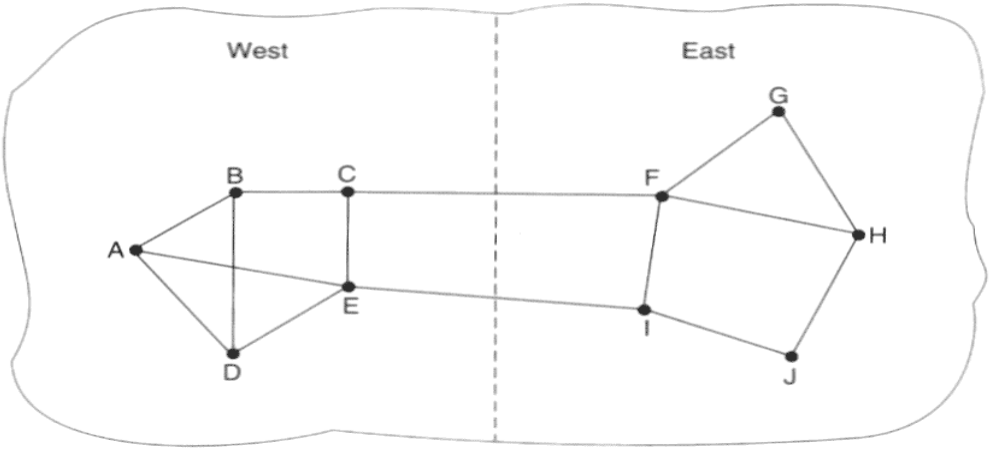
- **Começando quando o pacote ECHO é colocado na fila, ou seja, entra na fila.**
- **Começando quando o pacote ECHO atinge a frente da fila.**

Começando quando o pacote ECHO é colocado na fila.

- **Esta medida de tempo de ida e volta inclui o atraso.**
- **Um roteador que tenha que escolher entre duas linhas com a mesma largura de banda, uma das quais é fortemente carregada o tempo todo e a outra não, irá considerar a rota sobre a linha não carregada como o caminho mais curto.**
- **Esta escolha irá resultar em melhor desempenho.**

Começando quando o pacote ECHO atinge a frente da fila.

- **Esta medida de tempo de ida e volta não inclui o atraso. O problema que pode ocorrer incluindo o atraso é exemplificado através da Figura abaixo.**



- **A sub-rede mostrada na Figura é dividida em duas partes, leste e oeste, conectadas por 2 linhas, CF e EI.**
- **Suponhamos que a maior parte do tráfego entre leste e oeste esteja utilizando a linha CF e, como resultado, esta linha seja fortemente carregada com longos atrasos.**
- **Incluindo o atraso de enfileiramento no cálculo do caminho mais curto, o caminho EI será considerado mais atrativo.**
- **Após a instalação das novas tabelas, a maior parte do tráfego leste-oeste irá ocupar EI, sobrecarregando esta linha.**
- **Em conseqüência, na próxima atualização, CF será escolhido como o caminho mais curto.**
- **Como resultado, as tabelas de roteamento poderão oscilar, conduzindo a um roteamento de características erráticas.**

Se a ocupação (carga) é ignorada e somente a largura de banda é considerada, este problema não ocorrerá.

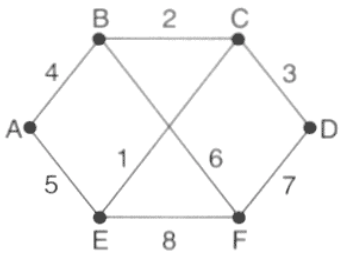
Uma solução alternativa consiste em espalhar a carga sobre ambas as linhas, mas esta solução não utiliza o conceito de "melhor caminho".

3. Construir Pacotes *Link State*.

- **Uma vez que a informação necessária para a troca tenha sido coletada, o próximo passo é construir, para cada roteador, um pacote contendo todo o conjunto de dados obtido.**
- **O pacote inicia com a identidade do emissor, seguida por um número de seqüência, um campo denominado *age*, uma lista de vizinhos e o atraso relativo a cada vizinho.**

Uma sub-rede exemplo é dada na Figura (a) abaixo, com atrasos conforme mostrados nos *links*.

Os pacotes *Link State* correspondentes para todos os seis roteadores são mostrados na Figura (b).



(a)

Link State Packets

A	
Seq.	
Age	
B	4
E	5

B	
Seq.	
Age	
A	4
C	2
F	6

C	
Seq.	
Age	
B	2
D	3
E	1

D	
Seq.	
Age	
C	3
F	7

E	
Seq.	
Age	
A	5
C	1
F	8

F	
Seq.	
Age	
B	6
D	7
E	8

(b)

A construção dos pacotes *Link State* é simples.

A parte difícil é determinar quando construí-los.

Uma possibilidade é construí-los periodicamente, ou seja, a intervalos regulares.

Outra possibilidade é quando algum evento ocorrer, tal como uma linha ou vizinho deixar de operar ou retornar ao serviço novamente, ou mudar suas características de forma considerável.

4. Distribuir Pacotes *Link State*.

- **A parte mais delicada do algoritmo consiste em distribuir os pacotes de forma confiável.**
- **À medida que os pacotes são distribuídos e instalados, os roteadores que recebem os primeiros pacotes mudarão suas rotas.**
- **Conseqüentemente, os diferentes roteadores podem estar usando diferentes versões da topologia, o que pode conduzir a problemas como inconsistências, *loops*, máquinas inacessíveis, entre outros.**

No algoritmo básico de distribuição a técnica *flooding* é utilizada para distribuir os pacotes *Link State*.

Para controlar o processo de *flooding*, cada pacote contém um número de seqüência que é incrementado para cada novo pacote enviado.

Os roteadores acompanham cada par (roteador fonte, nº de seqüência) que enxergam.

Quando um novo pacote *link state* chega, é verificado contra uma lista de pacotes já vistos.

- **Se é novo, é encaminhado sobre todas as linhas, exceto aquela por onde chegou.**
- **Se é duplicado, é descartado.**
- **Se um pacote com um nº de seqüência inferior ao mais alto visto até então chega, ele é rejeitado como sendo obsoleto.**

Problemas do algoritmo, que são contornáveis:

1. **Os nºs de seqüência podem ser reutilizados, gerando confusão. Para evitar este problema, são utilizadas seqüências de 32 bits (a uma taxa de um pacote *link state* por segundo, levará 137 anos para que o número se repita).**
2. **Se um roteador tem problemas, irá perder o rastreamento de sua seqüência. Se reiniciar a operar com zero, o próximo pacote será rejeitado como duplicado.**
3. **Se um nº de seqüência for corrompido, e o nº 65540 for recebido ao invés, por exemplo, do número 4, os pacotes numerados de 5 a 65540 serão rejeitados como obsoletos, desde que o nº de seqüência assumido para o pacote é 65540.**

A solução para estes problemas é incluir um número identificado como *age* associado a cada pacote, após o nº de seqüência, e decrementá-lo (um decremento por segundo).

Quando o contador *age* atinge zero, a informação recebida daquele roteador é descartada.

Um refinamento é dado a este algoritmo, para torná-lo mais robusto:

- **Quando um pacote *link state* chega a um roteador para *flooding*, ele não é colocado na fila para transmissão imediatamente.**
- **Ao invés disto, é posto em uma área de espera por um pequeno intervalo de tempo.**
- **Se outro pacote *link state* da mesma fonte chegar antes do anterior ser transmitido, seus números de seqüência são comparados.**
- **Se forem iguais, o duplicado é descartado.**
- **Se forem diferentes, o anterior é enviado.**
- **Para prevenir erros nas linhas roteador-roteador, todos os pacotes *link state* têm o recebimento acusado.**
- **A estrutura de dados usada pelo roteador B para a sub-rede mostrada na Figura anterior é descrita na Figura abaixo.**

			Send flags			ACK flags			
Source	Seq.	Age	A	C	F	A	C	F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

- **Cada linha corresponde a um pacote *link state* recentemente recebido, mas ainda não completamente processado.**

- **A tabela grava onde o pacote é originado, seu número de seqüência e *age*, e o conteúdo de informação.**
- **Em adição, existem *flags* de envio e acusamento de recepção para cada uma das três linhas de B (para A, C e F, respectivamente).**
- **Os *flags* de envio significam que o pacote deve ser enviado sobre a linha indicada.**
- **Os *flags* de recepção significam que os pacotes devem ter seu recebimento acusado.**

			Send flags			ACK flags			
Source	Seq.	Age	A	C	F	A	C	F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Na Figura, o pacote *link state* de A chegou diretamente, de tal forma que deve ser enviado para C e F e ter seu recebimento acusado para A , conforme indicado pelos bits de *flag*.

De forma similar, o pacote de F tem que ser enviado para A e C e ter seu recebimento acusado para F.

Entretanto, a situação que ocorre com o terceiro pacote recebido de E é diferente. O pacote chegou duas vezes, uma vez via EAB e a outra vez via EFB. Conseqüentemente, deve ser enviado somente para C, mas ter seu recebimento acusado para tanto A quanto F, conforme indicado pelos bits.

Se um pacote duplicado chega enquanto o original ainda está no *buffer*, os bits têm que ser mudados.

Por exemplo, se uma cópia dos pacotes de estado de C chegam de F antes da quarta entrada da tabela ter sido encaminhada, os seis bits irão ser trocados para 100011 para indicar que o pacote deve ter recebimento acusado para F, mas não enviado.

5. Computar as Novas Rotas.

- **Uma vez que um roteador tenha acumulado um conjunto completo de pacotes *link state*, ele pode construir o grafo da sub-rede inteira, porque cada *link* está representado.**
- **Cada *link* é, de fato, representado duas vezes, uma para cada direção. A média dos dois valores pode ser utilizada ou podem ser utilizados separadamente.**
- **O algoritmo Djikstra pode rodar localmente para construir o caminho mais curto para todos os possíveis destinos.**
- **Os resultados deste algoritmo podem ser instalados em tabelas de roteamento, e a operação normal reiniciada.**

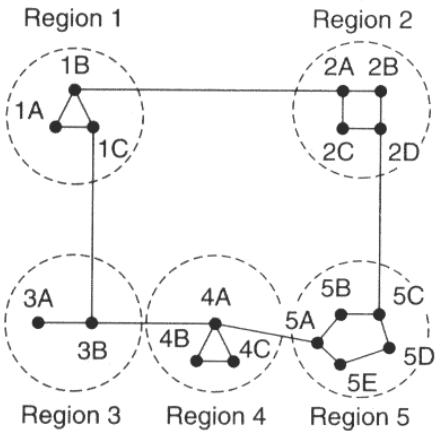
ROTEAMENTO HIERÁRQUICO

- **À medida que uma rede cresce em tamanho, as tabelas de roteamento dos roteadores crescem proporcionalmente.**

- **Não apenas a memória dos roteadores é aumentada com o aumento das tabelas, como também o tempo de CPU necessário para "varrer" as tabelas e a largura de banda necessária para enviar informações de *status*.**

- **Uma rede pode crescer a um ponto onde não é mais possível para cada roteador ter uma entrada para cada outro roteador, de tal forma que o roteamento deverá ser feito de forma hierárquica, como é feito em redes telefônicas.**
  - **Quando o roteamento hierárquico é usado, os roteadores são divididos em regiões, com cada roteador sabendo todos os detalhes sobre como rotear os pacotes dentro de sua própria região, mas não sabendo nada sobre a estrutura interna de outras regiões.**
  - **Quando diferentes redes são conectadas, é natural considerar cada uma como uma região separada para liberar os roteadores em uma rede de ter que saber a estrutura topológica dos outros.**
  - **Para redes grandes, uma hierarquia de dois níveis pode ser insuficiente; pode ser necessário agrupar as regiões em *clusters*, os *clusters* em zonas, as zonas em grupos e assim por diante.**

A Figura abaixo exemplifica uma hierarquia de dois níveis, com cinco regiões:



(a)

Full table for 1A

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

(b)

Hierarchical table for 1A

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

(c)

A tabela completa de roteamento para o roteador 1A tem 17 entradas, como mostrado em (b).

Quando o roteamento é feito hierarquicamente, como mostrado em (c), há entradas para todos os roteadores locais como antes, mas todas as outras regiões foram condensadas em um único roteador, de tal forma que todo o tráfego para a região 2 seguirá pela linha 1B-2A, mas o resto do tráfego remoto irá pela linha 1C-3B.

O roteamento herárquico reduziu a tabela de 17 para 7 entradas.

- **À medida que a razão do número de regiões para o número de roteadores por região aumenta, a economia na tabela aumenta.**
- **Infelizmente, este ganho em espaço na tabela representa um custo na forma de aumento no tamanho do caminho.**
- **Por exemplo, a melhor rota de 1A para 5C é através da região 3, porque isto é mais conveniente para a maior parte dos destinos na região 5.**

Quando uma única rede se torna muito grande, uma questão interessante surge: Quantos níveis uma hierarquia deve possuir?

Por exemplo, considere uma sub-rede com 720 roteadores.

Se não houvesse hierarquia, cada roteador necessitaria de 720 entradas na tabela de roteamento.

Se a sub-rede é particionada em 24 regiões de 30 roteadores cada, cada roteador precisa de 30 entradas locais mais 23 entradas remotas para um total de 53 entradas.

Se uma hierarquia de três níveis é escolhida, com oito *clusters*, cada um contendo 9 regiões de 10 roteadores, cada roteador necessita de 10 entradas para roteadores locais, 8 entradas para roteadores para outras regiões dentro de seu próprio *cluster*, e sete entradas para *clusters* distantes, para um total de 25 entradas.

Kamoun e Kleinrock mostraram em 1979 que o número ótimo de níveis para uma sub-rede de *N* roteadores é *lnN*, requerendo um total de *elnN* entradas por roteador.

Desta forma, o aumento no efetivo tamanho médio do caminho causado pelo roteamento hierárquico é suficientemente pequeno, de tal forma que é usualmente aceitável.