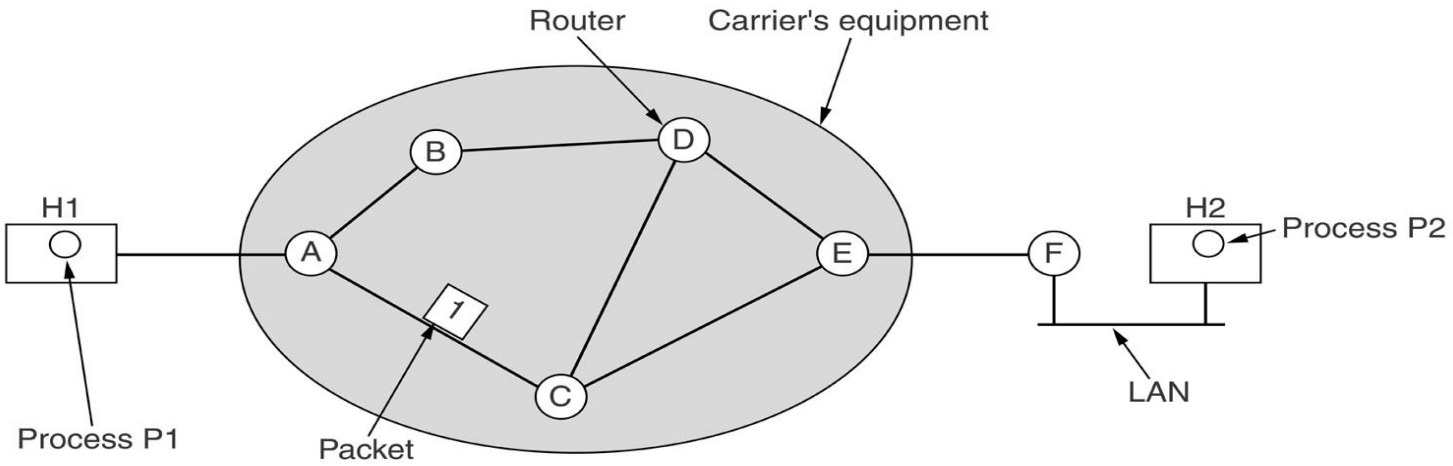
Parte 1
IP e Roteamento

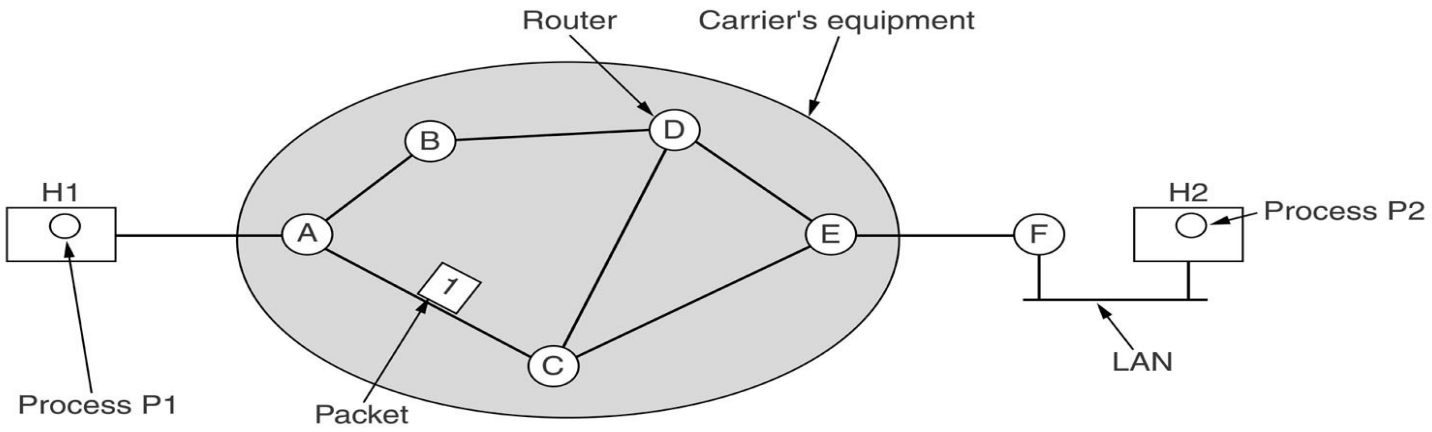
1.1 – Introdução

- ✔ A camada de rede está relacionada à transferência de pacotes entre a origem e o destino
- ✔ A camada de rede deve conhecer a topologia da sub-rede de comunicações (ou seja, o conjunto de todos os roteadores) e escolher os caminhos mais apropriados através dela
- ✔ A camada de rede deve ter o cuidado de escolher rotas que evitem sobrecarregar algumas das linhas de comunicação e roteadores



1.2 – Comutação de pacotes store-and-forward

- ✔ Um host transmite um pacote para o roteador mais próximo, seja em sua própria LAN ou sobre um enlace ponto a ponto para a concessionária de comunicações
- ✔ O pacote é armazenado ali até chegar totalmente, de forma que o total de verificação possa ser conferido
- ✔ Em seguida, ele é encaminhado para o próximo roteador ao longo do caminho, até alcançar o host de destino, onde é entregue



1.3 – Serviços oferecidos à camada de transporte

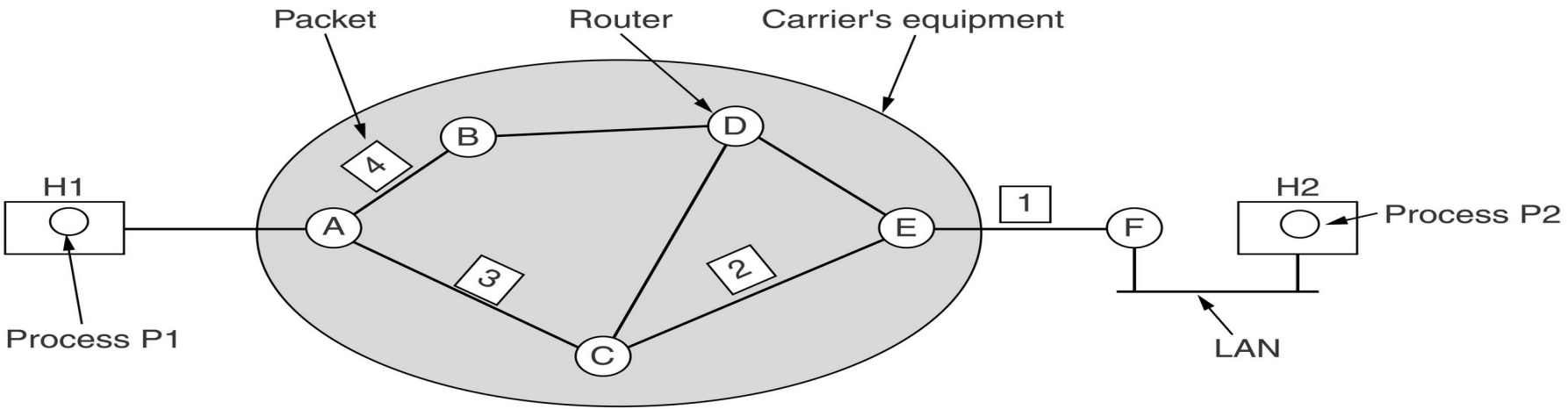
- Os serviços da camada de rede foram projetados tendo em vista os objetivos a seguir:
  - Os serviços devem ser independentes da tecnologia de roteadores
  - A camada de transporte deve ser isolada do número, do tipo e da topologia dos roteadores presentes
  - Os endereços de rede que se tornaram disponíveis para a camada de transporte devem usar um plano de numeração uniforme, mesmo nas LANs e WANs
- A camada de rede deve fornecer serviço orientado a conexões ou serviço sem conexões?

- Comunidade da Internet → sem conexões
  - 30 anos de experiência com uma rede de computadores ativa e real
  - A tarefa dos roteadores é tão somente movimentar pacotes
  - A sub-rede é inerentemente pouco confiável, independente de como tenha sido projetada
  - Os hosts devem aceitar o fato de que a sub-rede é pouco confiável e fazerem eles próprios o controle de erros e o controle de fluxo
  - Não deve ser realizada nenhuma forma de ordenação de pacotes e controle de fluxo, pois os hosts já cuidarão disso de qualquer maneira
  - Cada pacote deve ter o endereço de destino completo, pois todos são transportados independentemente de seus predecessores
  - e.g.: Internet

- Companhias telefônicas → orientado a conexões
  - 100 anos de experiência bem-sucedida com o sistema telefônico mundial
  - A qualidade de serviço é o fator dominante e, sem conexões na sub-rede, é muito difícil alcançar qualidade de serviço, em especial no caso de tráfego de tempo real, como voz e vídeo
  - e.g.: redes ATM
- É interessante observar que, à medida que as garantias de qualidade de serviço estão se tornando cada vez mais importantes, a Internet está evoluindo

- ✔ Se for oferecido o serviço sem conexões, os pacotes serão injetados individualmente na sub-rede e roteados de modo independente uns dos outros
- ✔ Não será necessária nenhuma configuração antecipada
- ✔ Os pacotes são chamados de datagramas, e a sub-rede de sub-rede de datagramas
- ✔ Se for usado o serviço orientado a conexões, terá de ser estabelecido um caminho desde o roteador de origem até o roteador de destino, antes de ser possível enviar quaisquer pacotes de dados
- ✔ Essa conexão é chamada circuito virtual, em analogia com os circuitos físicos estabelecidos pelo sistema telefônico, e a sub-rede é denominada sub-rede de circuitos virtuais

1.4 – Implementação do serviço sem conexões



A's table

Dest.	initially Line	later Line
A	–	–
B	B	B
C	C	C
D	B	B
E	C	B
F	C	B

C's table

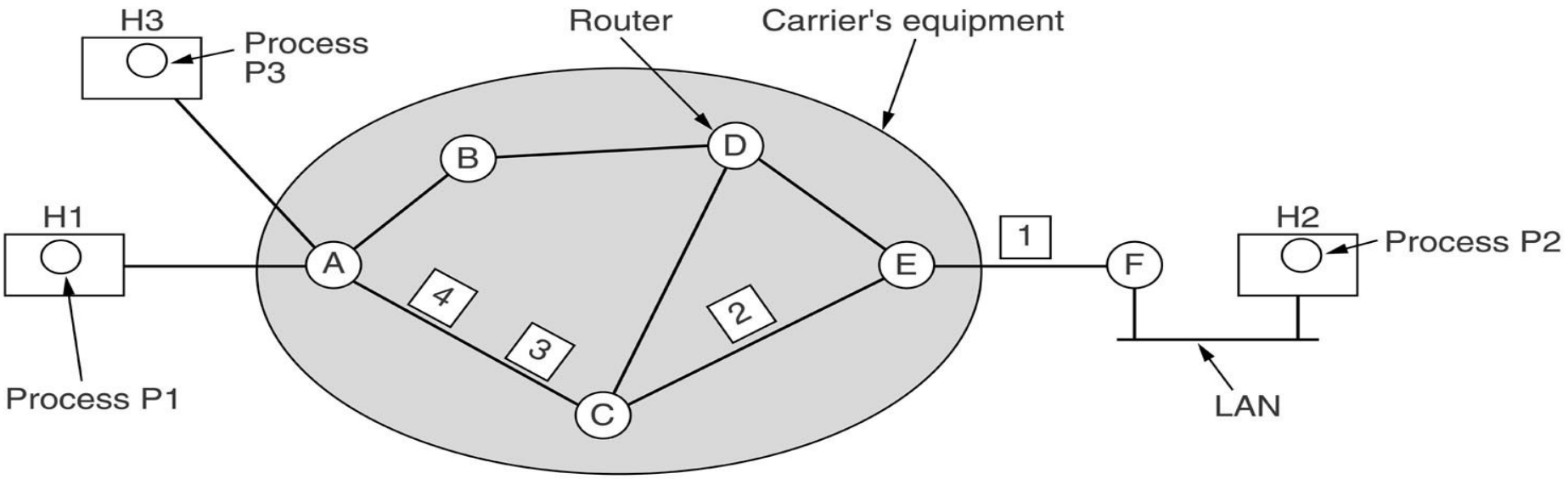
Dest.	Line
A	A
B	A
C	–
D	D
E	E
F	E

E's table

Dest.	Line
A	C
B	D
C	C
D	D
E	–
F	F

1.5 – Implementação do serviço orientado a conexões

- ✔ A ideia que rege os circuitos virtuais é evitar a necessidade de escolher uma nova rota para cada pacote enviado
- ✔ Quando uma conexão é estabelecida, escolhe-se uma rota desde a máquina de origem até a máquina de destino, e essa rota é armazenada em tabelas internas dos roteadores.
- ✔ A rota é usada por todo o tráfego que flui pela conexão
- ✔ Quando a conexão é liberada, o circuito virtual também é encerrado
- ✔ Cada pacote transporta um identificador, informando a que circuito virtual ele pertence



A's table

In		Out	
H1	1	C	1
H3	1	C	2

C's table

A	1	E	1
A	2	E	2

E's table

C	1	F	1
C	2	F	2

1.6 – Comparação entre sub-redes de circuitos virtuais e de datagramas

- Compromisso entre espaço de memória do roteador e largura de banda
  - Os circuitos virtuais permitem que os pacotes contenham números de circuitos em vez de endereços de destino completos
  - Se os pacotes tenderem a ser muito pequenos, um endereço de destino completo em cada pacote poderá representar um volume significativo de *overhead* e, portanto, haverá desperdício de largura de banda
  - O preço pago pelo uso de circuitos virtuais internamente é o espaço de tabela dentro dos roteadores
  - Uma sub-rede de datagramas pode ter uma entrada para cada destino possível, enquanto uma sub-rede de circuitos virtuais só precisa de uma entrada para cada circuito virtual

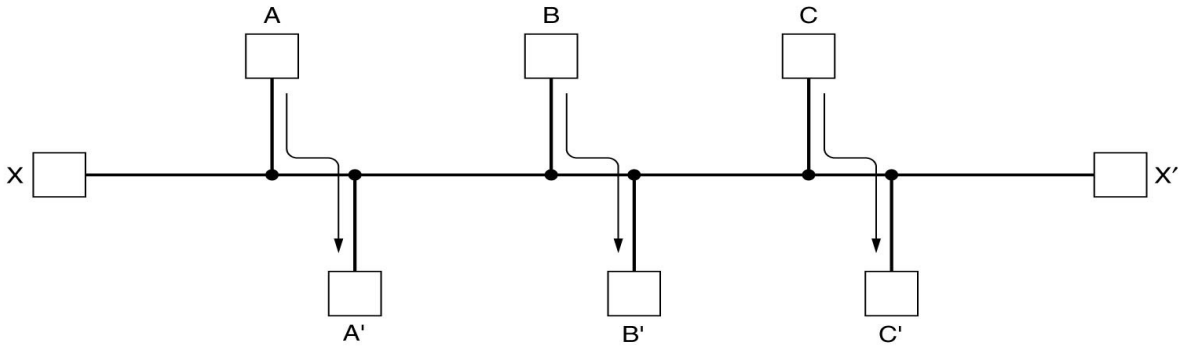
- Compromisso entre o tempo de configuração e o tempo de análise de endereço
  - O uso de circuitos virtuais requer uma fase de configuração, o que leva tempo e consome recursos
  - Entretanto, é fácil descobrir o que fazer com um pacote de dados em uma sub-rede de circuitos virtuais
- Os circuitos virtuais têm algumas vantagens na garantia de qualidade de serviço e ao evitarem o congestionamento dentro da sub-rede:
  - os recursos (por exemplo, *buffers*, largura de banda e ciclos da CPU) podem ser reservados antecipadamente, quando a conexão é estabelecida

- ✔ Os circuitos virtuais têm um problema de vulnerabilidade:
  - se um roteador de um circuito virtual apresentar uma falha e perder sua memória todos os circuitos virtuais que estiverem passando por ele terão de ser interrompidos
  - se um roteador de datagramas ficar fora do ar, somente os usuários cujos pacotes estiverem enfileirados no roteador naquele momento serão afetados
  - a perda de uma linha de comunicação é fatal para os circuitos virtuais que a utilizam
- ✔ Os datagramas permitem que os roteadores equilibrem o tráfego pela sub-rede:
  - as rotas podem ser parcialmente alteradas durante uma longa sequência de transmissões de pacotes

1.7 – Algoritmos de roteamento

- A principal função da camada de rede é rotear pacotes da máquina de origem para a máquina de destino
- Na maioria das sub-redes, os pacotes necessitarão de vários hops para cumprir o trajeto
- Os algoritmos que escolhem as rotas e as estruturas de dados que eles utilizam constituem um dos elementos mais importantes do projeto da camada de rede
- O algoritmo de roteamento é a parte do software da camada de rede responsável pela decisão sobre a linha de saída a ser usada na transmissão do pacote
  - Se a sub-rede utilizar datagramas internamente, essa decisão deverá ser tomada mais uma vez para cada pacote de dados recebido
  - Se a sub-rede utilizar circuitos virtuais internamente, as decisões de roteamento serão tomadas somente quando um novo circuito virtual estiver sendo estabelecido

- ✔ Propriedades que são desejáveis em um algoritmo de roteamento: correção, simplicidade, robustez, estabilidade, equidade e otimização
- ✔ Conflito entre equidade e otimização

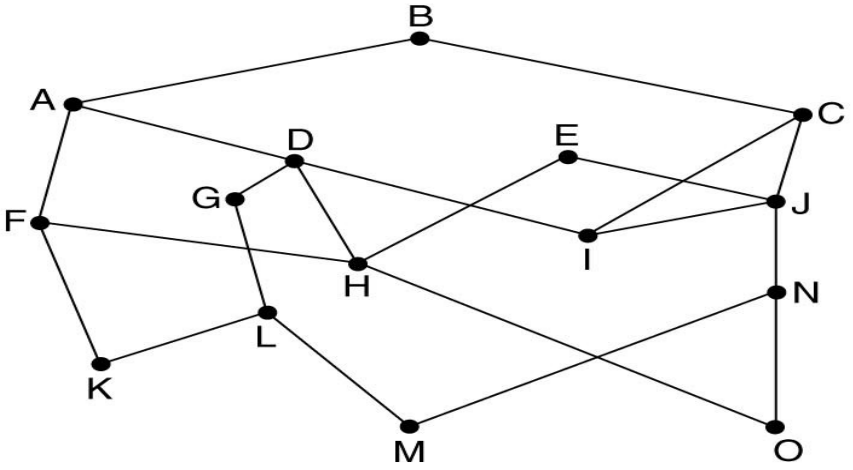


- ✔ O que está se tentando otimizar?
  - minimização do retardo médio de pacote
  - maximização do throughput total da rede
    - operar qualquer sistema de enfileiramento em uma velocidade próxima a de sua capacidade máxima implica um longo retardo de enfileiramento
  - minimizar o número de hops que um pacote deve percorrer

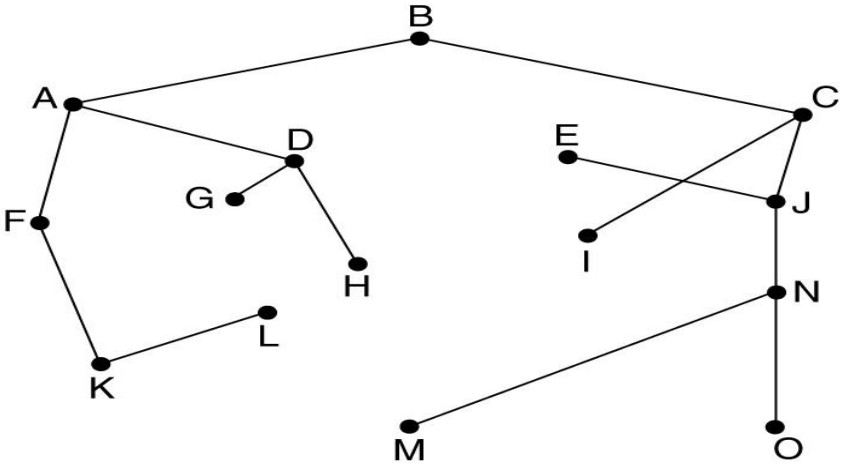
- ✔ Roteamento ***adaptativo* X *não-adaptativo***, ou, roteamento ***dinâmico* X *estático***
- ✔ Os algoritmos adaptativos se diferem em:
  - lugar em que obtêm suas informações
    - e.g.: no local, de roteadores adjacentes ou de todos os roteadores
  - momento em que alteram as rotas
    - e.g.: a cada T segundos, quando a carga se altera ou quando a topologia muda
  - unidade métrica utilizada para a otimização
    - e.g.: distância, número de hops ou tempo de trânsito estimado

Princípio da otimização

- ✔ Esse princípio estabelece que, se o roteador B estiver no caminho ótimo entre o roteador A e o roteador C, o caminho ótimo de B até C também estará na mesma rota
- ✔ Consequência direta do princípio de otimização:
  - o conjunto de rotas ótimas de todas as origens para um determinado destino forma uma árvore com raiz no destino (árvore de escoamento)
    - uma árvore de escoamento não é necessariamente exclusiva; podem existir outras árvores com caminhos de mesmo tamanho
- ✔ O objetivo de todos os algoritmos de roteamento é descobrir e utilizar as árvores de escoamento em todos os roteadores



(a)



(b)

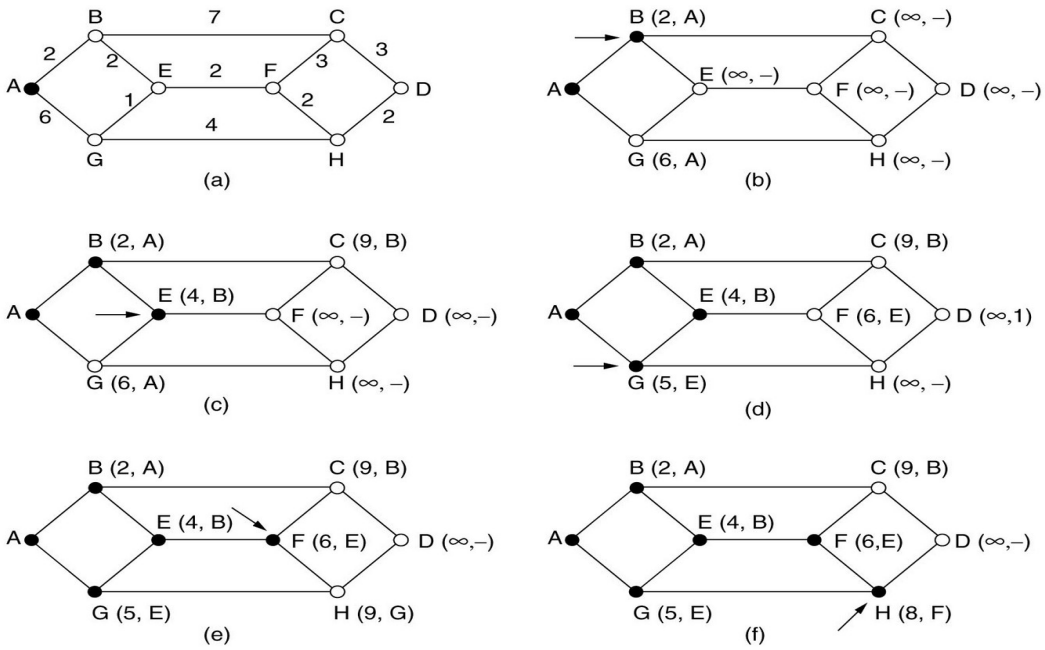
- ✔ Uma árvore de escoamento não contém loops:
  - Cada pacote será entregue dentro de um número finito e limitado de hops.
- ✔ Na prática, enlaces e roteadores podem sair do ar e voltar à atividade durante a operação:
  - Desse modo, diferentes roteadores podem ter ideias distintas sobre a topologia atual

1.7.1 – Roteamento pelo caminho mais curto

- A ideia é criar um grafo da sub-rede, com cada nó do grafo representando um roteador e cada arco indicando uma linha de comunicação (geralmente chamada enlace)
- Para escolher uma rota entre um determinado par de roteadores, o algoritmo simplesmente encontra o caminho “mais curto” entre eles no grafo
- Caminho “mais curto”:
  - Número de hops;
  - Distância geográfica em kilômetros;
  - Retardo médio de enfileiramento e transmissão;
  - Largura de banda;
  - Tráfego médio;
  - Custo de comunicação;
  - Comprimento médio da fila

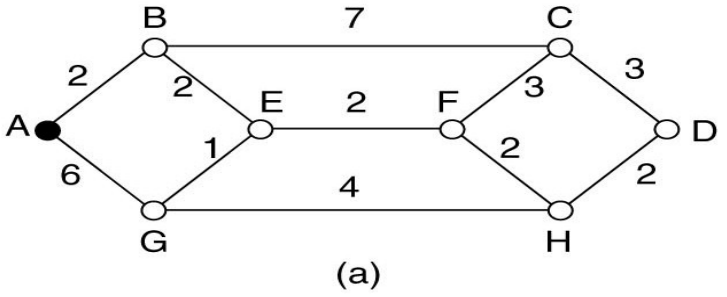
- ✔ Alterando-se a função de ponderação (atribuição de pesos), o algoritmo calcularia o caminho "mais curto" medido de acordo com qualquer critério ou com uma combinação de critérios
- ✔ Existem diversos algoritmos para se calcular o caminho “mais curto” entre dois nós de um grafo. Estudaremos o algoritmo de Dijkstra (1959)

- Cada nó é identificado (entre parênteses) por sua distância a partir do nó de origem ao longo do melhor caminho conhecido

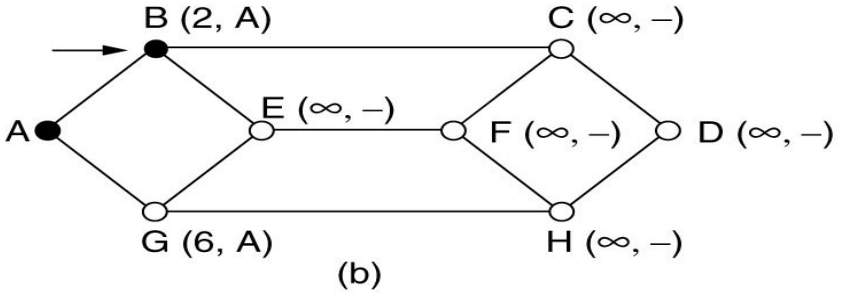


- Inicialmente, nenhum caminho é conhecido; portanto, todos os nós são rotulados com infinito
- À medida que o algoritmo prossegue e os caminhos são encontrados, os rótulos podem mudar, refletindo melhores caminhos
- Um rótulo pode ser provisório ou permanente
- No início, todos são provisórios
- Quando se descobre que um rótulo representa o caminho mais curto possível até a origem desse nó, ele se torna permanente e nunca mais será alterado

- Desejamos encontrar o caminho mais curto de A até D
- Começamos marcando o nó A como permanente (círculo preenchido)
- Examinamos separadamente cada um dos nós adjacentes a A (o nó ativo), alterando o rótulo de cada um deles para indicar a distância até A
- Sempre que um nó é rotulado novamente, ele também é rotulado com o nó a partir do qual o teste foi feito; assim, podemos reconstruir o caminho final mais tarde
- Verificamos todos os nós provisoriamente rotulados no grafo inteiro e tornamos permanente o nó que tem o menor rótulo. (esse nó passa a ser o novo nó ativo)

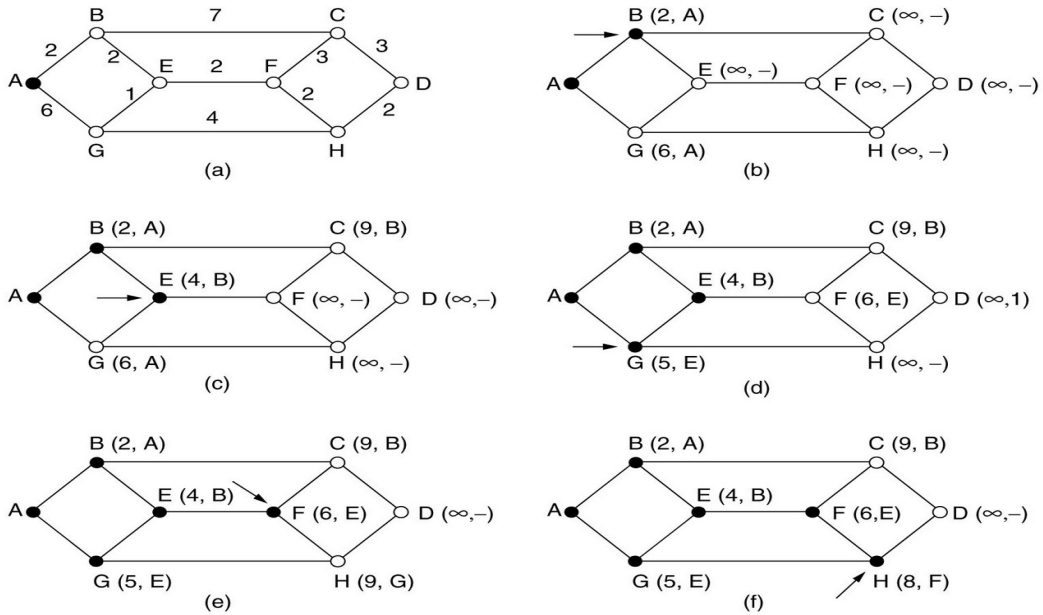


(a)



(b)

- Agora, começamos por B e examinamos todos os nós adjacentes a ele
- Se a soma do rótulo de B e a distância entre B e o nó que está sendo considerado for menor que o rótulo desse nó, teremos um caminho mais curto; portanto, o nó será rotulado novamente.
- Depois que todos os nós adjacentes ao nó ativo tiverem sido inspecionados e os rótulos provisórios tiverem sido alterados na medida do possível, o grafo inteiro será pesquisado até ser encontrado o nó com o rótulo provisório de menor valor. Esse nó passará a ser o nó permanente e se tornará o nó ativo na próxima iteração



1.7.2 – Roteamento por inundação

- ✔ Cada pacote de entrada é enviado para toda linha de saída, exceto para aquela em que chegou
- ✔ Gera uma quantidade infinita de pacotes duplicados
- ✔ Medidas devem ser tomadas para evitar a duplicação:
  - Ter um contador de hops contido no cabeçalho de cada pacote
    - o contador é decrementado em cada hop, com o pacote sendo descartado quando o contador atingir zero
    - o ideal é que o contador de hops seja inicializado com o comprimento do caminho desde a origem até o destino
    - Se não souber o tamanho do caminho, o transmissor poderá inicializar o contador com o valor referente ao pior caso, ou seja, o diâmetro total da sub-rede

- Controlar quais pacotes foram transmitidos por inundação, a fim de evitar transmiti-los uma segunda vez
  - uma forma de conseguir isso é fazer o roteador de origem inserir um número de sequência em cada pacote recebido de seus hosts
  - cada roteador precisará de uma lista por roteador de origem informando quais números de sequência originários desse ponto já foram vistos
  - se houver um pacote de entrada na lista, ele não será transmitido na inundação
  - para evitar que as listas cresçam indefinidamente, cada lista deve ser incrementada de acordo com um contador k, o que significa que todos os números de sequência até k foram vistos
  - quando um pacote for recebido, será fácil verificar se ele é uma cópia; se for, ele será descartado
  - a lista completa abaixo de k não é necessária, visto que k na verdade resume essa lista

- Uma variação um pouco mais prática do algoritmo de inundação é a ***inundação seletiva***
  - Os roteadores não enviam cada pacote de entrada para todas as linhas, apenas para aquelas que provavelmente estão na direção certa
    - não há muita razão para se utilizar uma linha da região leste para transportar um pacote cujo destino seja a região oeste
- Emprego do algoritmo de inundação
  - Em aplicações militares:
    - muitos roteadores podem ser destruídos a qualquer momento
  - Em aplicações de bancos de dados distribuídos:
    - às vezes é necessário atualizar todos os bancos de dados ao mesmo tempo

- Em redes sem fios:
  - todas as mensagens transmitidas por uma estação podem ser recebidas por todas as outras estações dentro de seu alcance de rádio
- Em medidores de desempenho:
  - usado como uma unidade de medida que servirá como base de comparação com outros algoritmos de roteamento
  - o algoritmo de inundação sempre escolhe o caminho mais curto, pois todos os caminhos possíveis são selecionados em paralelo
  - nenhum outro algoritmo é capaz de produzir um retardo de menor duração

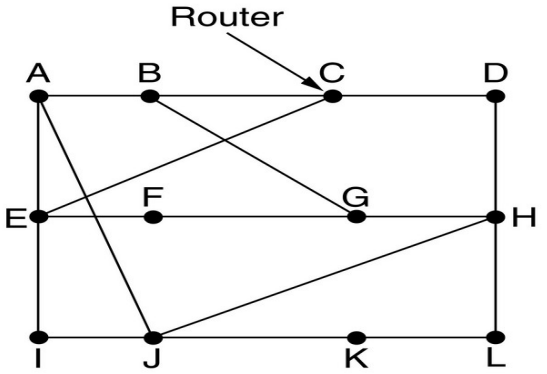
1.7.3 – Roteamento com Vetor de Distância

- Os algoritmos de roteamento com vetor de distância operam fazendo cada roteador manter uma tabela (isto é, um vetor) que fornece a melhor distância conhecida até cada destino e determina qual linha deve ser utilizada para se chegar lá
  - Essas tabelas são atualizadas através da troca de informações com os vizinhos
- Algoritmo original da ARPANET
  - Utilizado no início da Internet com o nome RIP
- A tabela (isto é, um vetor) contém uma entrada para cada um dos roteadores da sub-rede
  - Essa entrada contém duas partes:
    - a linha de saída preferencial a ser utilizada para esse destino
    - uma estimativa do tempo ou da distância até o destino

- A unidade métrica utilizada pode ser:
  - número de hops
  - retardo de tempo em milissegundos
  - número total de pacotes enfileirados no caminho, etc
- Pressupõe-se que o roteador conheça a "distância" até cada um de seus vizinhos
  - se a unidade métrica for o hop, a distância será de apenas um hop
  - se a unidade métrica for o comprimento da fila, o roteador simplesmente examinará cada uma das filas
  - se a unidade métrica for o retardo, o roteador poderá medi-lo diretamente com pacotes ECHO

- ✔ Ex: suponha que a unidade métrica é o retardo e que o roteador conhece o retardo até cada um de seus vizinhos
    - cada roteador envia a cada vizinho uma lista de seus retardos estimados até cada destino
    - o roteador também recebe uma lista semelhante de cada vizinho
    - suponha que uma dessas tabelas tenha acabado de chegar do vizinho X e que *Xi* é a estimativa de X sobre o tempo que ele levará para chegar até o roteador i
    - o roteador sabe que o seu retardo para X é de *m* ms
    - o roteador saberá que pode alcançar o roteador i por meio de X em (*Xi* + *m*) ms
    - efetuando esse cálculo para cada vizinho, um roteador pode descobrir qual estimativa parece ser a melhor
    - o roteador já poderá usar essa estimativa e a linha correspondente em sua nova tabela de roteamento

- Exemplo: cálculo da tabela de roteamento de J
  - a parte (a) da figura mostra uma sub-rede
  - as quatro primeiras colunas da parte (b) mostram os vetores de retardo recebidos dos vizinhos do roteador J



(a)

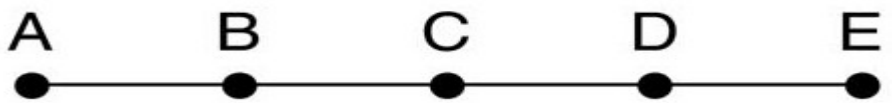
To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

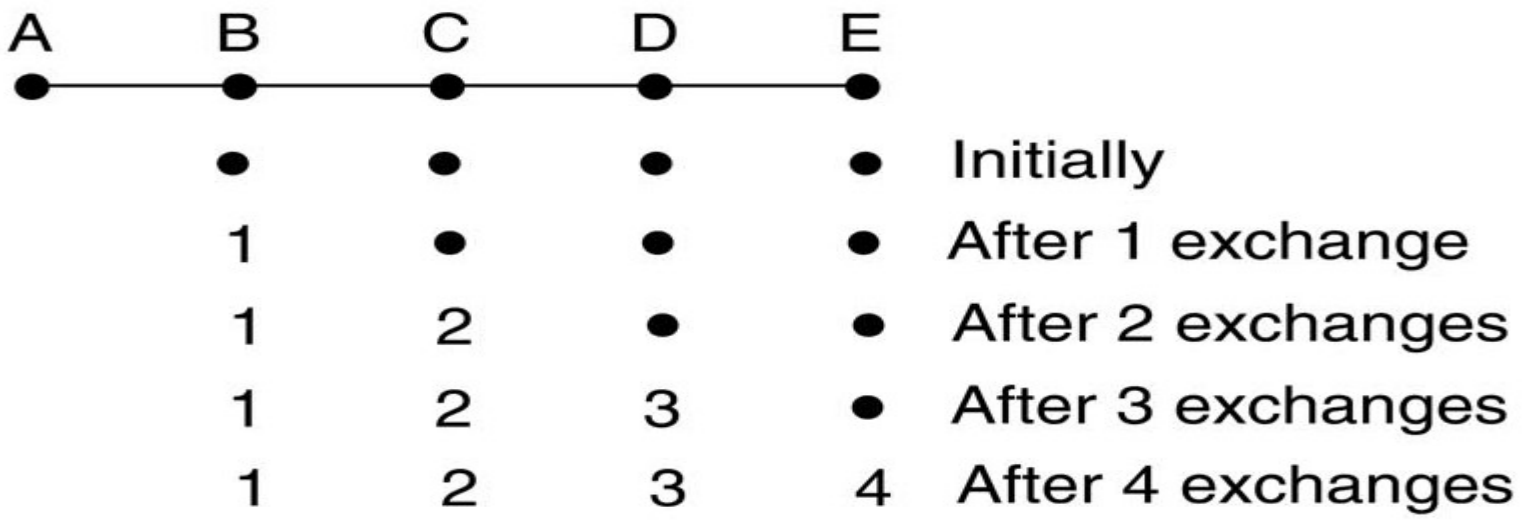
Vectors received from J's four neighbors

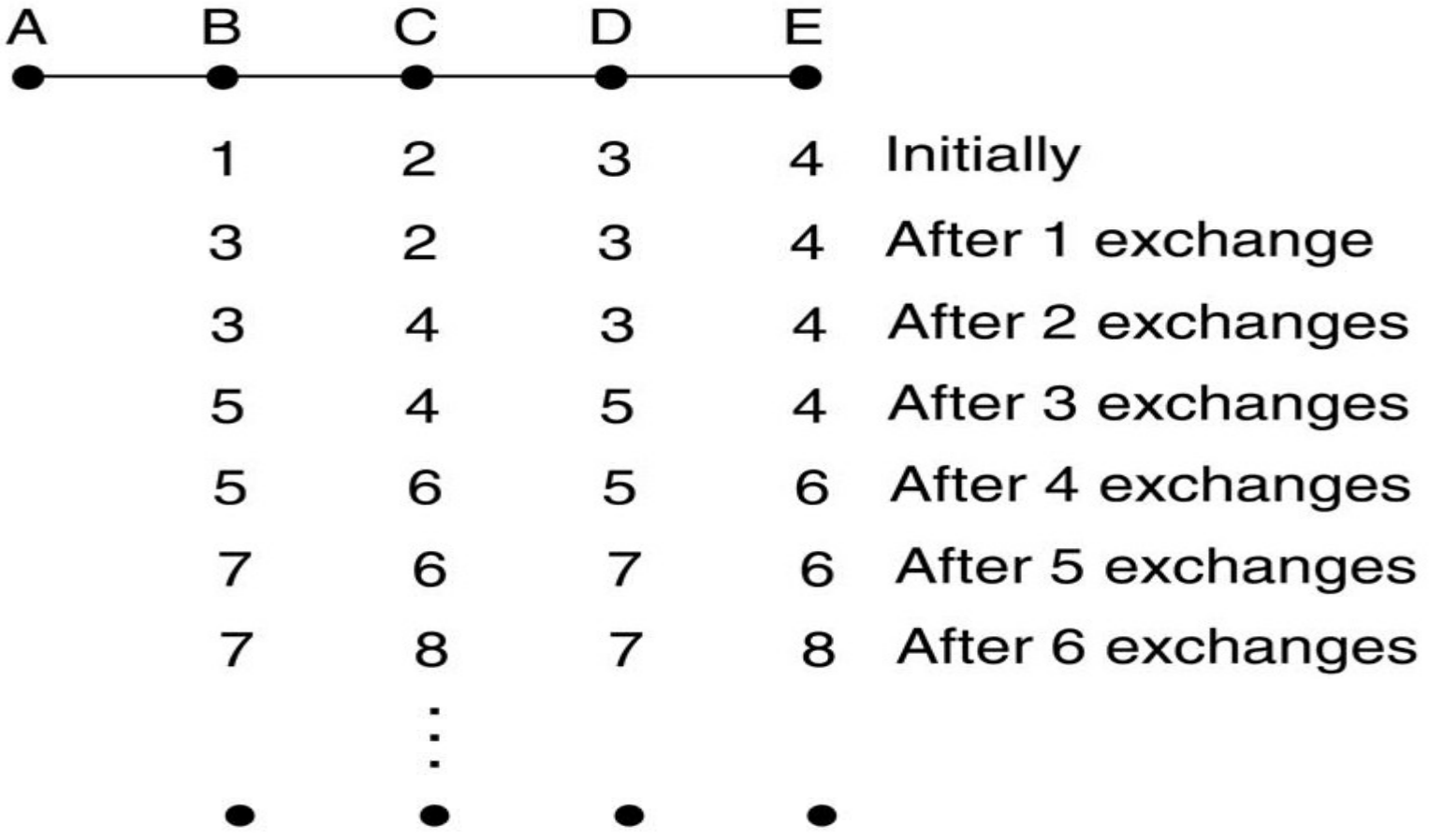
(b)

O problema da contagem até infinito

- Problema na prática: converge lentamente para a tabela correta definitiva
  - reage com rapidez a boas notícias, mas reage devagar a más notícias
- Exemplo: considere uma sub-rede de cinco nós (linear) na qual a unidade métrica para calcular o retardo é o número de hops.
- Suponha que A inicialmente esteja inativo e que todos os outros roteadores saibam disso. De repente A fica ativo




A
B
C
D
E
Initially
1
After 1 exchange
1 2
After 2 exchanges
1 2 3
After 3 exchanges
1 2 3 4
After 4 exchanges

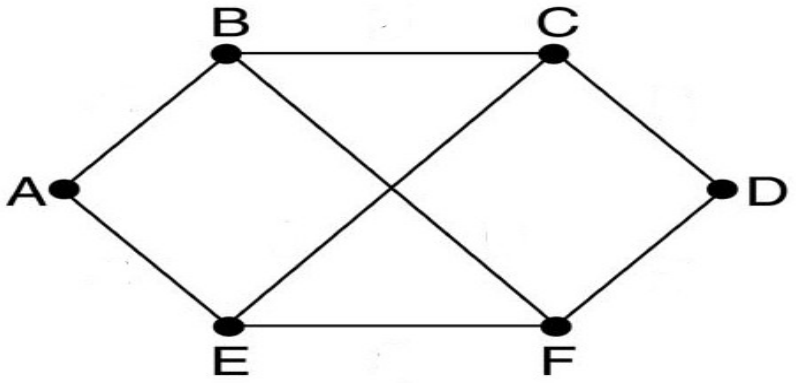

A
B
C
D
E
1 2 3 4 Initially
3 2 3 4 After 1 exchange
3 4 3 4 After 2 exchanges
5 4 5 4 After 3 exchanges
5 6 5 6 After 4 exchanges
7 6 7 6 After 5 exchanges
7 8 7 8 After 6 exchanges

✔ Exercício:

Considere a sub-rede da figura abaixo. O roteamento com vetor de distância é usado e os vetores a seguir acabaram de entrar no roteador C:

- de B: (5,0,8,12,6,2)
- de D: (16,12,6,0,9,10)
- de E: (7,6,3,9,0,4)

Os retardos medidos para B, D e E são 6, 3 e 5, respectivamente. Qual a nova tabela de roteamento de C? Forneça a linha de saída a ser usada e o retardo esperado

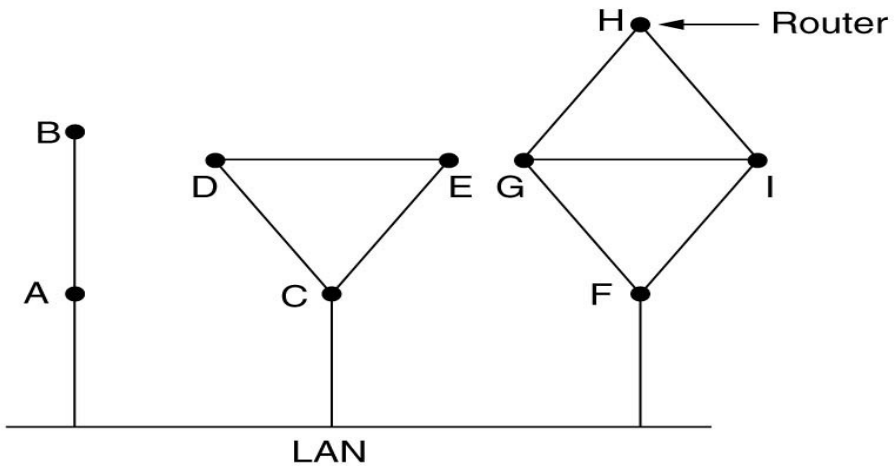


1.7.4 – Roteamento por Estado de Enlace

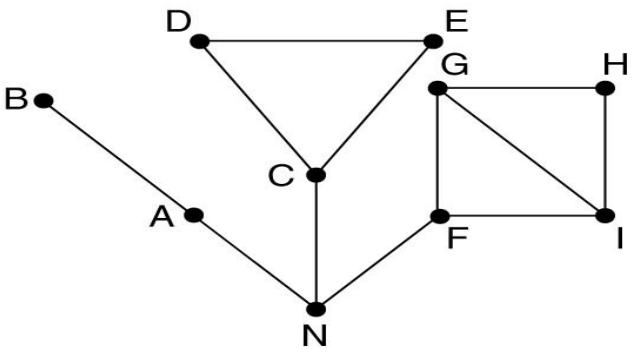
- Substituiu o algoritmo Vetor de Distância na ARPANET/Internet
  - Como no início todas as linhas tinham 56 kbps, a largura de banda das linhas não era importante
  - Vetor de Distância convergia lentamente
- O princípio de funcionamento é:
  - 1. Cada roteador deve descobrir seus vizinhos e aprender seus endereços de rede
  - 2. Cada roteador deve medir o retardo ou o custo até cada um de seus vizinhos
  - 3. Cada roteador deve criar um pacote que informe tudo o que ele acabou de aprender
  - 4. Cada roteador de enviar esse pacote a todos os outros roteadores
  - 5. Cada roteador deve calcular o caminho mais curto até cada um dos outros roteadores

Conhecendo os vizinhos

- Quando um roteador é inicializado, sua primeira tarefa é aprender quem são seus vizinhos
  - Esse objetivo é alcançado enviando-se um pacote HELLO especial em cada linha ponto a ponto
  - O roteador da outra extremidade deve enviar de volta uma resposta, informando quem é
  - Esses nomes devem ser globalmente exclusivos
- Quando dois ou mais roteadores estão conectados por uma LAN, a situação é um pouco mais complicada



(a)

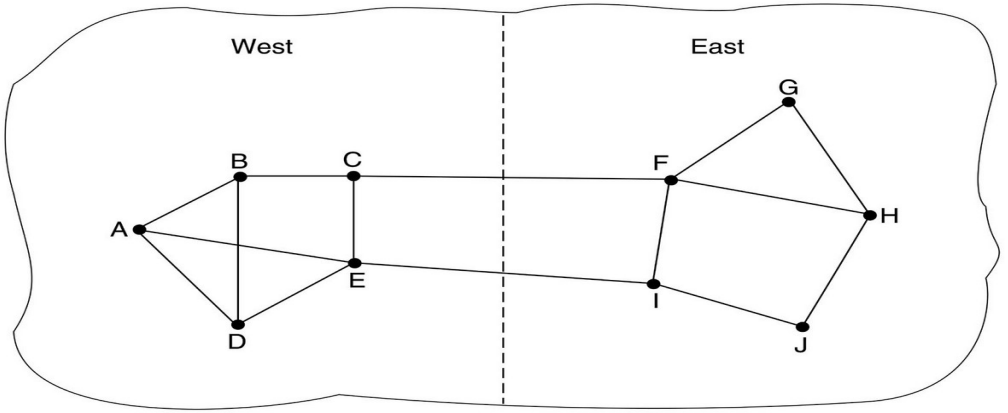


(b)

Como medir o custo da linha

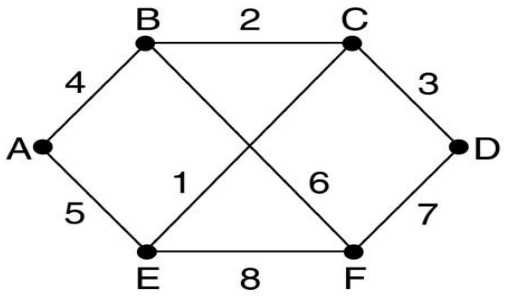
- O algoritmo exige que cada roteador conheça o retardo para cada um de seus vizinhos (ou pelo menos tenha uma boa estimativa de qual seja ele)
  - Enviando um pacote ECHO pela linha, e o outro lado deve transmitir de volta imediatamente
  - Mede-se o tempo de ida e volta e divide-se por dois
    - Pressupõe-se que os retardos são simétricos
  - Para se melhorar a estimativa, deve-se realizar o teste várias vezes e usar a média obtida
- A carga deve ou não ser levada em consideração ao ser medido o retardo?

- Para levar a carga em conta:
  - o timer encarregado de medir o tempo de ida e volta deve ser iniciado quando o pacote ECHO for enfileirado
  - quando um roteador tiver de escolher entre duas linhas com a mesma largura de banda, sendo que uma está mais carregada que a outra, o roteador irá considerar a rota sobre a linha não carregada um caminho mais “curto”
  - essa opção resultará em melhor desempenho
- Para ***não*** levar a carga em conta:
  - o timer deve ser iniciado quando o pacote ECHO atingir o início da fila
  - evita oscilação da tabela de roteamento



Como criar pacotes de estado de enlace

- De posse das informações necessárias para a troca, a próxima etapa é cada roteador criar um pacote que contenha todos os dados:
  - identidade do transmissor
  - número de sequência
  - idade
  - lista de vizinhos, com o retardo referente a cada vizinho
- Facilidade para serem criados, dificuldade para saber *quando* devem ser criados



(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)

Distribuição dos pacotes de estado de enlace

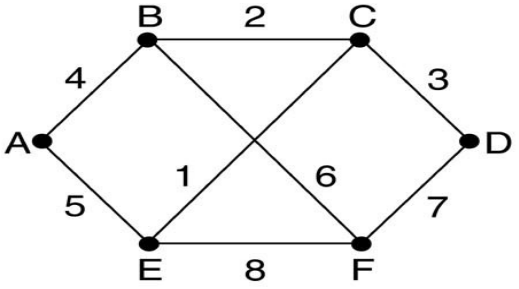
- À medida que os pacotes são distribuídos e instalados, os roteadores que obtiverem os primeiros pacotes mudarão suas rotas
  - os diferentes roteadores talvez estejam usando diferentes versões da topologia
  - isso poderá levar a inconsistências, loops, máquinas inacessíveis e outros problemas
- A ideia fundamental é usar o algoritmo de inundação para distribuir os pacotes de estado de enlace
  - cada pacote contém um número de sequência que é incrementado para cada novo pacote enviado

- os roteadores controlam todos os pares (roteador de origem, sequência) que veem
- quando é recebido, o novo pacote de estado de enlace é conferido na lista de pacotes já verificados
  - se for novo, ele será encaminhado a todas as linhas, exceto à linha por onde chegou
  - se for uma cópia, o pacote será descartado
  - se um pacote recebido tiver com número de sequência mais baixo que o mais alto número de sequência detectado até o momento, ele será rejeitado e considerado obsoleto

- Esse procedimento apresenta alguns problemas:
  - 1. Se os números de sequência se repetirem, a confusão tomará conta
    - solução: usar um número de sequência de 32 bits
    - Ex: com um pacote de estado de enlace por segundo, seriam necessários 137 anos para um número se repetir
  - 2. Se um roteador apresentar falha, ele perderá o controle de seu número de sequência
    - Ex: se ele começar de novo em 0, o pacote seguinte será rejeitado por ser considerado uma cópia
  - 3. Se um número de sequência for adulterado
    - Ex: se o número 65.540 for recebido no lugar do número 4 (um erro de 1 bit), os pacotes de 5 a 65.540 serão rejeitados como obsoletos

- A solução para todos esses problemas é incluir a idade de cada pacote após o número de sequência e decrementá-la uma vez por segundo
  - Quando a idade atingir zero, as informações desse roteador serão descartadas
  - O campo idade também é decrementado por cada roteador durante o processo inicial de inundação
- Quando um pacote de estado de enlace chega a um roteador para inundação, ele não é imediatamente enfileirado para transmissão
  - Em vez disso, ele é colocado em uma área de retenção para aguardar um pouco
  - Se outro pacote de estado de enlace da mesma origem chegar antes da transmissão do primeiro pacote, seus números de sequência serão comparados
    - se forem iguais, a cópia será descartada
    - se forem diferentes, o mais antigo será descartado
- Para evitar erros nas linhas entre dois roteadores, todos os pacotes de estado de enlace são confirmados

- Quando uma linha ficar ociosa, a área de retenção será varrida sequencialmente, a fim de se selecionar um pacote ou uma confirmação a enviar
- Exemplo: A estrutura de dados utilizada pelo roteador B da sub-rede
  - Cada linha corresponde a um pacote de estado de enlace recém-chegado, mas ainda não totalmente processado
  - A tabela registra a origem do pacote, seu número de sequência e idade, e os dados correspondentes
  - Além disso, há flags de transmissão e confirmação para cada uma das três linhas de B (para A, C e F, respectivamente)
    - Os flags de transmissão significam que o pacote deve ser enviado na linha indicada
    - Os flags de confirmação significam que ele deve ser confirmado na linha indicada



(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)

			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	

Como calcular as novas rotas

- Uma vez que um roteador tenha acumulado um conjunto completo de pacotes de estado de enlace, ela poderá criar o grafo da sub-rede completo
  - Na verdade, todo enlace será representado duas vezes, uma vez em cada sentido
- O algoritmo de Dijkstra pode ser executado no local com a finalidade de criar o caminho mais curto até todos os destinos possíveis
  - os resultados desse algoritmo podem ser instalados nas tabelas de roteamento
- No caso de uma sub-rede com n roteadores, cada qual com k vizinhos, a memória necessária para armazenar os dados de entrada é proporcional a kn

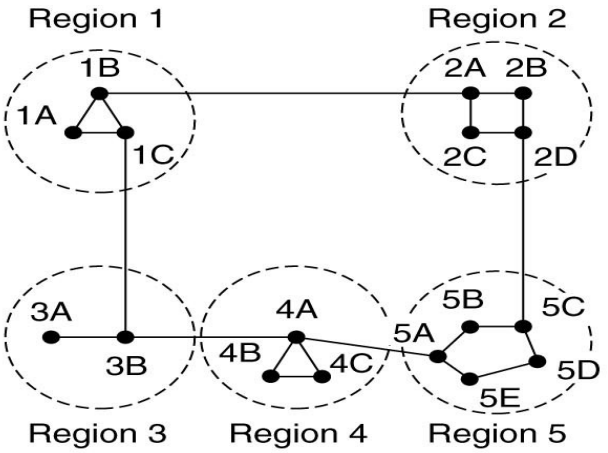
- Problemas com o hardware ou com o software podem causar grandes complicações com esse algoritmo (e também com outros)
- Exemplos:
  - se um roteador alegar ter uma linha que na realidade não tem, ou esquecer uma linha que tem
  - se um roteador deixar de encaminhar pacotes ou danificá-los enquanto os encaminhar
  - se a memória do roteador se esgotar ou se ele calcular o roteamento incorretamente
- À medida que a sub-rede crescer até a faixa de dezenas ou centenas de milhares de nós, a probabilidade de algum roteador falhar ocasionalmente deixará de ser desprezível
  - Perlman (1988) analisa em detalhes esses problemas e suas soluções

1.7.5 – Roteamento Hierárquico

- ✔ À medida que as redes aumentam de tamanho, as tabelas de roteamento dos roteadores crescem proporcionalmente
- ✔ Roteadores exigem mais:
  - memória -> tabelas cada vez maiores
  - CPU -> dedicar mais tempo de processamento
  - largura de banda -> mais envio de relatórios de status
- ✔ A rede pode crescer até o ponto em que deixará de ser viável cada roteador ter uma entrada correspondente a cada outro roteador
  - o roteamento terá de ser feito de forma hierárquica, como na rede telefônica

- Roteamento hierárquico -> os roteadores serão divididos em regiões
  - cada roteador conhece todos os detalhes sobre como rotear pacotes para destinos dentro de sua própria região
  - não conhecem nada sobre a estrutura interna de outras regiões
- Quando diferentes redes estão interconectadas, cada uma é vista como uma região separada
  - os roteadores de uma rede ficam liberados da necessidade de conhecer a estrutura topológica das outras redes
- Em redes muito grandes -> uma hierarquia de dois ou mais níveis
  - provavelmente será necessário reunir as regiões em agrupamentos, os agrupamentos em zonas, as zonas em grupos etc., até faltarem nomes para os agregados

- ✔ Exemplo de uma hierarquia de vários níveis: pacote sendo roteado de Berkeley, na Califórnia, até Malindi, no Quênia
    - o roteador de Berkeley conheceria a topologia detalhada da Califórnia, mas enviaria todo o tráfego de fora do estado para o roteador de Los Angeles
    - o roteador de Los Angeles seria capaz de rotear o tráfego para outros roteadores domésticos, mas enviaria todo o tráfego destinado a outros países para Nova York
    - o roteador de Nova York seria programado de modo a direcionar todo o tráfego para o roteador, do país de destino, que é responsável pelo tratamento do tráfego vindo do exterior, no nosso caso, em Nairóbi.
    - por fim, o pacote seguiria seu caminho descendente pela árvore no Quênia até chegar a Malindi



(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)

- Quantos níveis a hierarquia deve ter?
- Exemplo: considere uma sub-rede com 720 roteadores
  - se não houver hierarquia, cada roteador precisará de 720 entradas na tabela de roteamento
  - se a sub-rede for particionada em 24 regiões de 30 roteadores cada uma, cada roteador precisará de 30 entradas locais e mais 23 entradas remotas, total de 53 entradas
  - se for escolhida uma hierarquia de três níveis com oito agrupamentos, cada um deles contendo 9 regiões de 10 roteadores, cada roteador precisará de 10 entradas para roteadores locais, 8 entradas para roteamento até outras regiões dentro de seu próprio agrupamento e 7 entradas para agrupamentos distantes, total de 25 entradas
- Kamoun e Kleinrock (1979) descobriram que o número ótimo de níveis para uma sub-rede com N roteadores é
  ln(N), exigindo um total de e.ln(N) entradas por roteador
- Eles também demonstraram que o aumento na extensão do caminho médio efetivo causado pelo roteamento hierárquico é suficientemente pequeno e aceitável

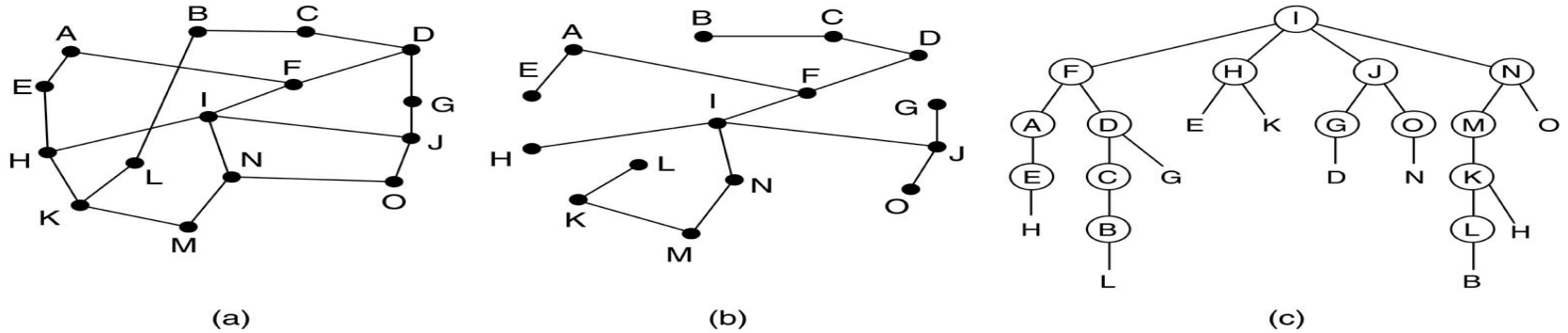
1.7.6 – Roteamento por Difusão

- O envio de um pacote a todos os destinos simultaneamente é chamado **difusão** (***broadcasting***)
- Um primeiro método de difusão permite à origem enviar um pacote específico a cada destino
  - o método não só desperdiça largura de banda como também exige que a origem tenha uma lista completa de todos os destinos
    - na prática, essa pode ser a única possibilidade. No entanto, essa maneira é a menos desejável
- Um segundo método de difusão seria a **inundação**
  - o problema da inundação como técnica de difusão é o mesmo problema que ela tem como um algoritmo de roteamento ponto a ponto:
    - gera pacotes demais e consome largura de banda em excesso.

- Um terceiro método de difusão: **o roteamento para vários destinos** -> cada pacote conterá uma lista de destinos ou um mapa de bits indicando os destinos desejados
  - quando um pacote chega a um roteador, este verifica todos os destinos para determinar o conjunto de linhas de saída que serão necessárias
  - o roteador gera uma nova cópia do pacote para cada linha de saída a ser utilizada e inclui em cada pacote somente os destinos que vão usar a linha
    - o conjunto de destinos é particionado entre as linhas de saída
  - após um número suficiente de hops, cada pacote transportará somente um destino e poderá ser tratado como um pacote normal

- Um quarto método de difusão faz uso explícito da árvore de escoamento para o roteador que inicia a difusão
  - se cada roteador souber quais de suas linhas pertencem à árvore de escoamento, ele poderá copiar um pacote de difusão de entrada em todas as linhas da árvore de escoamento, exceto aquela em que o pacote chegou
  - esse método faz excelente uso da largura de banda, gerando o número mínimo absoluto de pacotes necessários para realizar essa tarefa
  - o único problema é que cada roteador deve ter conhecimento de sua árvore de escoamento

- Um quinto método de difusão (**encaminhamento pelo caminho inverso**) é uma tentativa de aproximação com o comportamento do quarto método, mesmo quando os roteadores nada sabem sobre árvores de escoamento
  - quando um pacote de difusão chega a um roteador, o roteador verifica se o pacote chegou pela linha que normalmente é utilizada para o envio de pacotes à origem da difusão
    - em caso positivo, há uma excelente possibilidade de que o pacote de difusão tenha seguido a melhor rota a partir do roteador e seja, portanto, a primeira cópia a chegar ao roteador
    - o roteador encaminhará cópias do pacote para todas as linhas, exceto aquela por onde ele chegou
    - em caso negativo, se o pacote de difusão tiver chegado em uma linha diferente da preferencial para alcançar a origem, ele será descartado como uma provável duplicata



- No primeiro hop, I envia pacotes para F, H, J e N
  - Cada um desses pacotes chega ao caminho preferencial para I (supondo-se que o caminho preferencial acompanhe a árvore de escoamento) e é então indicado por um círculo em torno da letra
- No segundo hop, são gerados oito pacotes, dois por cada um dos roteadores que receberam um pacote no primeiro hop
  - todos os oito pacotes chegam a roteadores não visitados anteriormente, e cinco deles chegam ao longo da linha preferencial
- No terceiro hop, seis pacotes são gerados e somente três chegam pelo caminho preferencial (em C, E e K); os outros são duplicatas
- Depois de cinco hops e 24 pacotes, a difusão termina, em comparação com quatro hops e 14 pacotes que haveria se a árvore de escoamento fosse seguida exatamente

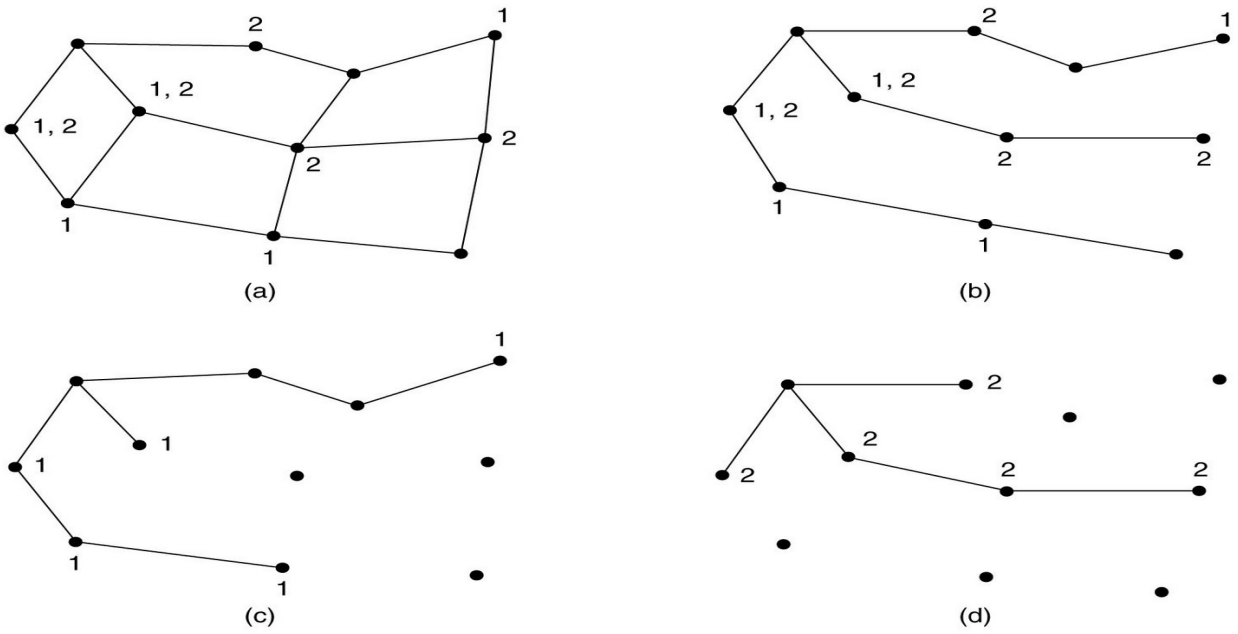
- ✔ A principal vantagem do **encaminhamento pelo caminho inverso** é que ele é ao mesmo tempo razoavelmente eficiente e fácil de implementar
- ✔ Não têm o overhead de uma lista de destino ou um mapa de bits em cada pacote de difusão, como ocorre na estratégia de endereçamento para vários destinos
- ✔ Ele também não requer nenhum mecanismo especial para interromper o processo, como é o caso do algoritmo de inundação

1.7.7 – Roteamento por Multidifusão

- Muitas vezes é necessário que um host envie uma mensagem a todos os outros membros do grupo
  - se o grupo for pequeno, ele poderá simplesmente enviar a cada um dos outros membros uma mensagem ponto a ponto
  - se o grupo for grande, essa estratégia se tornará dispendiosa
  - às vezes, a difusão pode ser utilizada; no entanto, o uso da difusão para informar a 1000 máquinas de uma rede com um milhão de nós é ineficiente, porque a maioria dos receptores não está interessada na mensagem (ou, pior ainda, os receptores estão definitivamente interessados, mas não veem a mensagem)
- **Multidifusão** é um meio para enviar mensagens a grupos bem definidos que têm um tamanho numericamente grande, mas que são pequenos em comparação com a rede como um todo

- A mulidifusão exige o gerenciamento de grupos
  - será preciso usar algum método para criar e destruir grupos, e para permitir que os hosts entrem e saiam de grupos
- Um host ao se associar a um grupo, ele informará seu roteador desse fato
  - é importante que os roteadores saibam quais de seus hosts pertencem a cada um dos grupos
  - os hosts devem informar seus roteadores sobre alterações na associação a grupos, ou então os roteadores terão de consultar seus hosts periodicamente
  - os roteadores devem ficar sabendo quais de seus hosts estão em cada um dos grupos
  - os roteadores darão essa informação a seus vizinhos, e assim a informação se propagará pela sub-rede

- ✔ Cada roteador calcula uma árvore de escoamento que engloba todos os outros roteadores da sub-rede
- ✔ Quando um processo envia um pacote de multidifusão a um grupo, o primeiro roteador examina sua árvore de escoamento e a poda, removendo todas as linhas que não levam a hosts que são membros do grupo



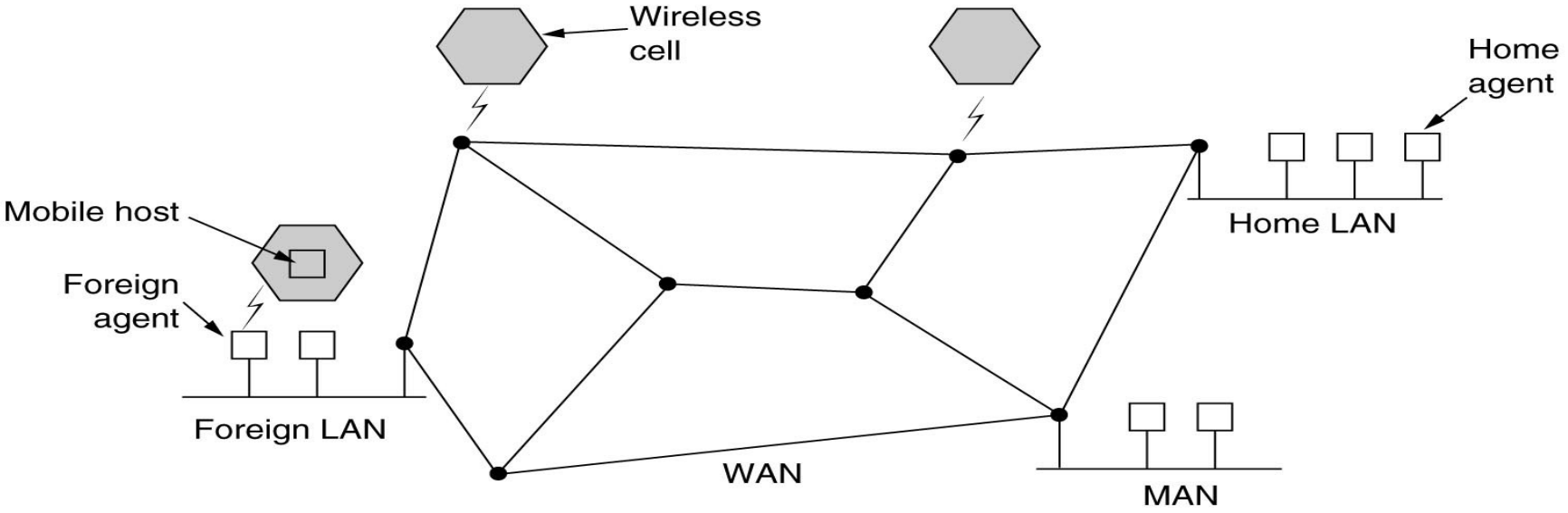
- ✔ Existem vários métodos que podem ser usados para podar a árvore de escoamento
- ✔ O mais simples pode ser usado se o roteamento por estado de enlace for empregado e cada roteador estiver ciente da topologia completa, inclusive de quais hosts pertencem a cada um dos grupos
  - a árvore de escoamento pode ser podada, começando pela extremidade de cada caminho, seguindo em direção à raiz e removendo todos os roteadores que não pertencem ao grupo em questão

- Quando se emprega o roteamento com vetor de distância, é necessário utilizar uma outra estratégia de poda
  - o algoritmo básico é o encaminhamento pelo caminho inverso
    - sempre que um roteador sem hosts interessados em um grupo específico e sem conexões para outros roteadores recebe uma mensagem de multidifusão relacionada a esse grupo, ele responde com uma mensagem PRUNE, informando ao transmissor que este não deve enviar mais mensagens de multidifusão para esse grupo
    - quando um roteador sem membros de grupos entre seus hosts recebe tais mensagens em todas as suas linhas, ele também pode responder com uma mensagem PRUNE
  - uma desvantagem potencial desse algoritmo é que ele é mal dimensionado para redes grandes
    - suponha que uma rede tenha n grupos, cada qual com uma média de m membros
    - para cada grupo, devem ser armazenadas m árvores de amplitudes podadas, perfazendo um total de mn árvores
    - quando há muitos grupos grandes, é necessário um espaço de armazenamento considerável para armazenar todas as árvores

- Um projeto alternativo utiliza **árvores baseadas no núcleo**
  - é calculada uma única árvore de escoamento por grupo, com a raiz (o núcleo) próxima ao centro do grupo
  - para enviar uma mensagem de multidifusão, um host a envia ao núcleo, que então faz a multidifusão ao longo da árvore de escoamento
  - embora essa árvore não seja ótima para todas as origens, a redução dos custos de armazenamento de m árvores para uma única árvore por grupo é uma economia importante

1.7.8 – Roteamento para hosts móveis

- Antes de rotear um pacote para um host móvel, primeiro a rede precisa localizá-lo
- Os hosts que nunca se movem são chamados ***estacionários***
  - estão conectados à rede por fios de cobre ou fibra ópticas
- Os hosts ***migrantes*** são hosts estacionários que se deslocam de um local fixo para outro de tempos em tempos, mas que utilizam a rede apenas quando estão fisicamente conectados a ela
- Os hosts ***visitantes*** realmente utilizam seus computadores em trânsito e querem manter suas conexões à medida que se deslocam
- Hosts ***móveis*** serão considerados os hosts que estão fora de suas bases e que ainda querem se manter conectados (migrantes e visitantes)


Wireless cell
Home agent
Mobile host
Foreign agent
Home LAN
Foreign LAN
WAN
MAN

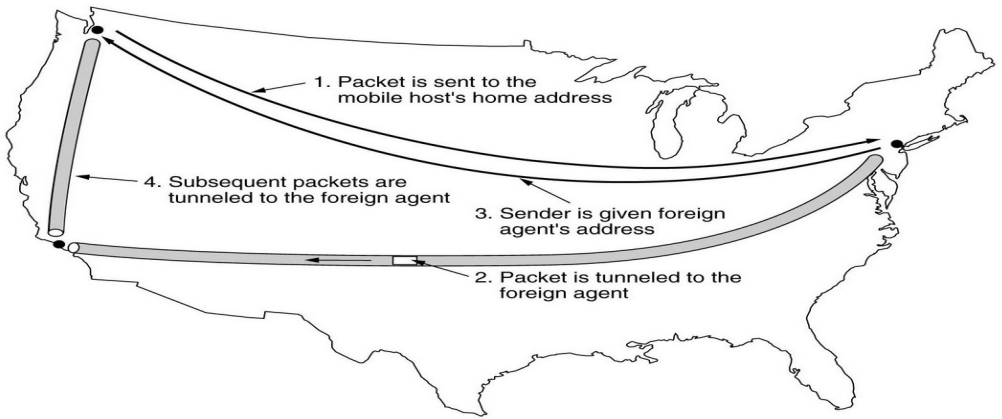
- Partimos do princípio de que todos os hosts têm um local inicial permanente, que nunca muda
- Os hosts também têm um endereço local permanente que pode ser usado para determinar seus locais iniciais
- O objetivo do roteamento em sistemas com hosts móveis é tornar possível o envio de pacotes a hosts móveis que estejam usando seus endereços locais, onde quer que eles estejam
- O grande problema é localizá-los
- ***Agentes externos*** → processos que controlam todos os hosts móveis que visitam a área
- ***Agente local*** → controla os hosts cuja base se encontra na área, mas no momento que estão visitando outra área
- Quando um novo host entra em uma área, ele deve se registrar com o agente externo dessa área

- Procedimento de registro:
  - 1. Periodicamente, cada agente externo transmite um pacote anunciando sua existência e seu endereço
    - um host móvel recém-chegado pode aguardar uma dessas mensagens
    - se nenhuma mensagem chegar rápido o suficiente, o host móvel poderá transmitir (por difusão) um pacote com a mensagem: “Há algum agente externo por aí?”
  - 2. O host móvel se registra com o agente externo, fornecendo seu endereço local, o endereço atual da camada de enlace de dados e algumas informações de segurança
  - 3. O agente externo entra em contato o agente local do host móvel e diz: “Um de seus hosts está por aqui.”
    - a mensagem do agente externo para o agente local contém o endereço de rede do agente externo
    - a mensagem contém ainda as informações de segurança, a fim de convencer o agente local de que o host móvel realmente está lá

- 4. O agente local examina as informações de segurança, que contêm um timbre de hora, para provar que foi gerado há alguns segundos
  - se tudo estiver correto, o agente local diz ao agente externo para prosseguir
- 5. Quando o agente externo obtém a confirmação do agente local, ele cria uma entrada em suas tabelas e informa ao host móvel que agora ele está registrado

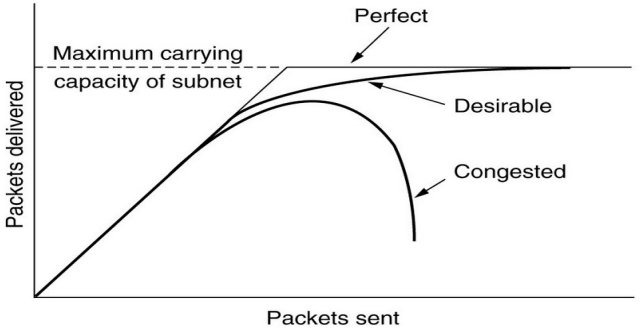
✔ Quando um host deixa uma área, isso também deve ser anunciado para permitir o cancelamento do registro

- muitos usuários desligam seus computadores abruptamente quando terminam de usá-los



1.8 – Algoritmos de controle de congestionamento

- ***Congestionamento*** → situação em que há pacotes em excesso na sub-rede (ou em parte da sub-rede) causando a diminuição do desempenho
  - Quando o número de pacotes enviados na sub-rede pelos hosts está dentro de sua capacidade de transporte, eles são todos entregues (exceto alguns que sofram com erros de transmissão), e o número entregue é proporcional ao número enviado
  - Quando o tráfego aumenta muito, os roteadores já não são capazes de suportá-lo e começam a perder pacotes
    - No caso de tráfego muito intenso, o desempenho entra em colapso total, e quase nenhum pacote é entregue

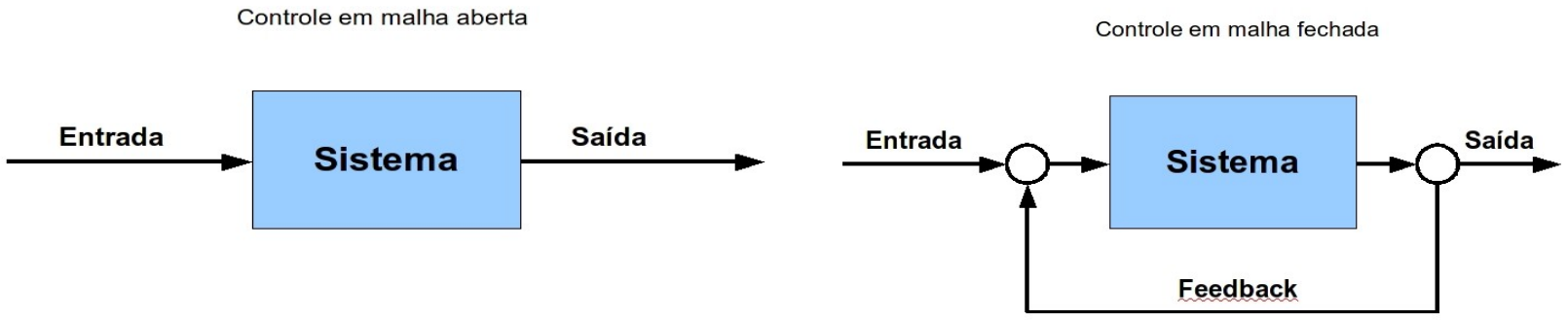


- O congestionamento pode ser causado por diversos fatores:
  - se os fluxos de pacotes começarem a chegar repentinamente em três ou quatro linhas de entrada e todas precisarem da mesma linha de saída, haverá uma fila
    - se a memória for insuficiente para conter todos eles, os pacotes se perderão
    - a inclusão de mais memória ajudará até certo ponto
    - e se os roteadores tiverem um volume infinito de memória?
    - o congestionamento piorará pois, no momento em que os pacotes chegarem ao início da fila, eles já terão sido temporizados (repetidamente) e as duplicatas já terão sido enviadas
  - processadores lentos também podem causar congestionamento
    - se as CPUs dos roteadores forem lentas na execução de tarefas administrativas (enfileiramento de buffers, atualização de tabelas etc.), poderão surgir filas, mesmo que haja capacidade de linha suficiente
  - linhas de baixa largura de banda também podem causar congestionamento

- A atualização das linhas sem atualização dos processadores, ou vice-versa, ajuda um pouco, mas geralmente empurra o gargalo para outro lugar
  - o problema real é uma incompatibilidade entre partes do sistema
  - esse problema persistirá até que todos os componentes estejam em equilíbrio
- Diferença entre controle de congestionamento e controle de fluxo
  - o controle de congestionamento é uma questão global, envolvendo o comportamento de todos os hosts, de todos os roteadores, de operações de repasse dentro dos roteadores e de todos os outros fatores que tendem a reduzir a capacidade de transporte da sub-rede
  - o controle de fluxo se baseia no tráfego ponto a ponto entre um transmissor e um receptor, garantindo que um transmissor rápido não possa transmitir dados continuamente com maior rapidez do que o receptor é capaz de absorver

1.8.1 – Princípios gerais do controle de congestionamento

- ✔ Pode ser analisado de acordo com os princípios da Teoria de Controle
  - essa abordagem nos leva à divisão de todas as soluções em dois grupos: loops abertos e loops fechados



- As soluções para malha aberta tentam resolver o problema com um bom projeto, para tentar garantir que os problemas não ocorrerão
  - uma vez que o sistema esteja em funcionamento, não serão feitas correções que afetem processos em andamento
- Ferramentas para controle de congestionamento que utilizam malha aberta:
  - decidir quando aceitar mais tráfego
  - decidir quais pacotes devem ser descartados e quando isso deve ser feito
  - programar decisões em vários pontos da rede
- Tudo isso tem em comum o fato de que as decisões serão tomadas sem levar em conta o estado atual da rede

- As ferramentas para malha fechada se baseiam no conceito de um loop de feedback
- Essa estratégia tem três partes, quando aplicada ao controle de congestionamento:
  - 1. Monitorar o sistema para detectar quando e onde ocorre congestionamento
  - 2. Enviar essas informações para lugares onde alguma providência possa ser tomada
  - 3. Ajustar a operação do sistema para corrigir o problema.
- Várias unidades métricas podem ser usadas para monitorar a sub-rede quanto à ocorrência de congestionamentos
  - percentagem de todos os pacotes descartados por falta de espaço em buffer
  - média dos comprimentos de fila
  - número de pacotes interrompidos por alcançarem o tempo limite e que são retransmitidos
  - retardo médio de pacotes
  - desvio padrão do retardo de pacotes

- A segunda etapa do loop de feedback é transferir informações sobre congestionamento do ponto em que o fenômeno é detectado para o ponto em que algo pode ser feito em relação a ele
  - a solução mais óbvia é o roteador detectar o congestionamento com a finalidade de enviar um pacote à origem ou às origens de tráfego, anunciando o problema
    - esses pacotes extras aumentam a carga exatamente no momento em que a sub-rede está congestionada
  - pode-se também reservar um bit ou um campo em todos os pacotes para que os roteadores o preencham sempre que o congestionamento superar algum limite inicial
    - quando detecta esse estado de congestionamento, o roteador preenche o campo em todos os pacotes de saída, a fim de alertar os vizinhos
  - pode-se também fazer com que os hosts ou roteadores enviem pacotes de sondagem periodicamente para perguntar de forma explícita sobre o congestionamento
    - essa informação pode ser usada para rotear o tráfego em áreas problemáticas

- Em todos os esquemas de feedback, a esperança é que o conhecimento do congestionamento faça com que os hosts tomem as providências necessárias para reduzi-lo
- A escala de tempo deve ser ajustada com cuidado
  - se todas as vezes que dois pacotes chegarem em sequência um roteador gritar PARE e toda vez que um roteador ficar ocioso por 20 s ele gritar VÁ, o sistema oscilará muito e nunca convergirá
  - se ele aguardar 30 minutos para ter certeza antes de comunicar algo, o mecanismo de controle de congestionamento reagirá muito lentamente
- Obter o tempo correto é uma questão não trivial
- A presença de congestionamento significa que a carga é (temporariamente) maior do que os recursos (de uma parte do sistema) podem manipular. Há duas soluções:
  - aumentar os recursos
  - diminuir a carga

- Aumentando os recursos:
  - usar linhas extras para aumentar temporariamente a largura de banda entre determinados pontos
  - aumento da potência de transmissão em satélites quase sempre proporciona largura de banda mais alta
  - divisão do tráfego em várias rotas em vez de sempre utilizar a melhor rota
  - roteadores sobressalentes que podem ser ativados para oferecer maior capacidade
- Diminuindo a carga:
  - negar o serviço a alguns usuários
  - piorar a qualidade do serviço para alguns ou para todos os usuários
  - fazer os usuários programarem suas necessidades de um modo mais previsível

1.8.2 – Políticas de prevenção de congestionamento

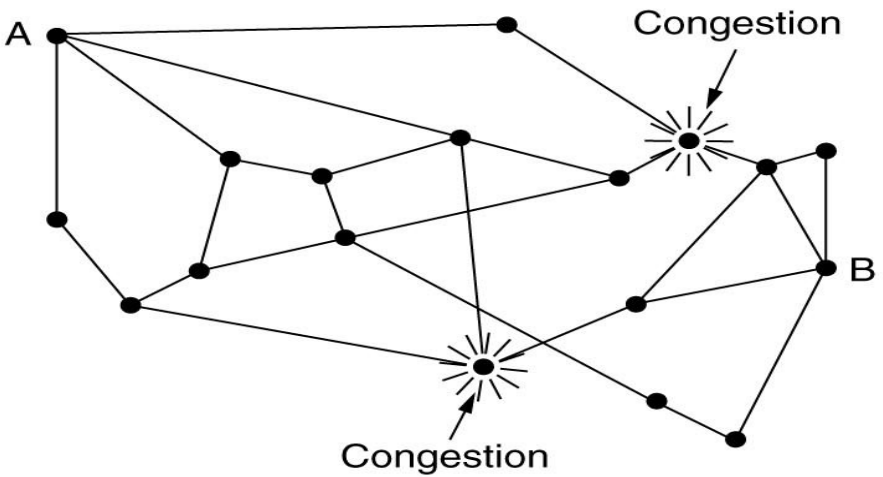
- ✔ Políticas na camada de enlace:
  - ***política de retransmissão*** → trata da rapidez com que um transmissor chega ao timeout
    - um transmissor que chega ao timeout com rapidez e retransmite todos os pacotes pendentes usando go back n irá impor uma carga mais pesada sobre o sistema do que um transmissor lento que utilize retransmissão seletiva
  - ***política de cache fora de ordem*** → armazenamento em buffer
    - se os receptores costumam descartar todos os pacotes fora da ordem, esses pacotes terão de ser retransmitidos mais tarde, criando carga extra
  - ***política de confirmação***
    - se cada pacote for confirmado imediatamente, os pacotes de confirmação irão gerar tráfego extra
    - se as confirmações forem guardadas para serem transmitidas por carona sobre o tráfego inverso, poderão ocorrer interrupções e retransmissões extras
  - ***política de controle de fluxo***
    - uma janela pequena reduz a taxa de dados

- ✔ Políticas na camada de rede
  - **circuitos virtuais versus datagramas**
    - muitos algoritmos de controle de congestionamento só funcionam com sub-redes de circuitos virtuais
  - **política de serviços e de enfileiramento de pacotes**
    - os roteadores têm uma fila por linha de entrada, uma fila por linha de saída ou os dois
    - essa estratégia está relacionada à ordem em que os pacotes são processados. e.g.: por rodízio ou baseada na prioridade
  - **política de descarte**
    - é a regra que informa qual pacote será descartado quando não houver espaço
  - **algoritmo de roteamento**
    - evitar o congestionamento espalhando o tráfego por todas as linhas
  - **gerenciamento da duração do pacote →** lida com o tempo de duração de um pacote antes de ser descartado
    - se esse tempo for muito longo, os pacotes perdidos poderão atrapalhar o funcionamento por muito tempo
    - se o tempo for muito curto os pacotes poderão chegar ao timeout antes de alcançarem seu destino

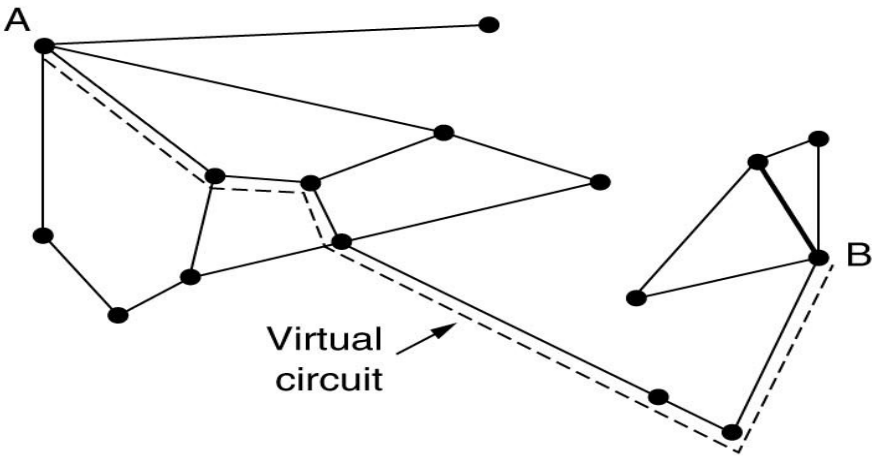
- Políticas na camada de transporte
  - na camada de transporte, surgem as mesmas questões que ocorrem na camada de enlace de dados
  - é mais difícil determinar o intervalo de timeout, porque o tempo de trânsito na rede é menos previsível que o tempo de trânsito sobre um fio entre dois roteadores
    - se o intervalo de timeout for curto demais, serão enviados pacotes extras desnecessariamente
    - se ele for muito longo, o congestionamento será reduzido, mas o tempo de resposta será sacrificado sempre que um pacote for perdido

1.8.3 - Controle de congestionamento em sub-redes de circuitos virtuais

- ***Controle de admissão*** → uma vez que o congestionamento tenha dado alguma indicação de sua existência, nenhum outro circuito virtual será estabelecido até que o problema tenha passado
  - todas as tentativas de estabelecer novas conexões da camada de transporte falharão
  - fácil de ser executada
  - no sistema telefônico, quando um switch fica sobrecarregado, o controle de admissão também é acionado, e não são fornecidos sinais de discagem
- Uma estratégia alternativa é permitir novos circuitos virtuais, mas rotear com cuidado todos os novos circuitos virtuais em áreas problemáticas



(a)



(b)

- Outra estratégia é negociar um acordo entre o host e a sub-rede quando um circuito virtual for configurado
  - normalmente, esse acordo especifica o volume e a formatação do tráfego, a qualidade de serviço exigida e outros parâmetros
  - para manter essa parte do acordo, a sub-rede reservará recursos ao longo do caminho quando o circuito for configurado
    - esses recursos podem incluir espaço em tabelas e buffers nos roteadores, além de largura de banda nas linhas
  - dessa maneira, é improvável que ocorra congestionamento nos novos circuitos virtuais, pois todos os recursos necessários estarão disponíveis
  - esse tipo de reserva sempre poderá ser feito todo o tempo como procedimento de operação padrão ou somente quando a sub-rede estiver congestionada
    - uma desvantagem de fazer isso o tempo todo é a tendência a desperdiçar recursos
  - o preço desse controle de congestionamento é a largura de banda não utilizada ou seja, desperdiçada

1.8.4 - Controle do congestionamento em sub-redes de datagramas

- ✔ Cada roteador pode monitorar a utilização de suas linhas de saída e de outros recursos
  - o roteador pode associar a cada linha uma variável real, u, cujo valor entre 0,0 e 1,0 reflete a utilização recente dessa linha
  - para manter uma boa estimativa de u, uma amostra da utilização instantânea da linha, f (0 ou 1), pode ser obtida periodicamente, sendo u atualizada de acordo com:

$$u_{nova} = a u_{antiga} + (1 - a) f$$

  - sempre que u ultrapassa o limite, a linha de saída entra em um estado de "advertência"
  - cada pacote recém-chegado é conferido para sabermos se sua linha de saída encontra-se em estado de advertência
    - se estiver, alguma ação será adotada

O bit de advertência

- O estado de advertência é assinalado ativando-se um bit especial no cabeçalho do pacote
  - Quando o pacote chega a seu destino, a entidade de transporte copia o bit na próxima confirmação a ser enviada de volta à origem e, em seguida, a origem interrompe o tráfego
  - Enquanto estiver no estado de advertência, o roteador continua a definir o bit de advertência, e isso significa que a origem continua a receber informações com o bit ativado
  - A origem monitora a fração de confirmações com o bit ativado e ajusta sua velocidade de transmissão de acordo com ele
  - Enquanto os bits de advertência continuam a fluir, a origem continua a diminuir sua taxa de transmissão
  - Quando diminui a velocidade de chegada das confirmações, a origem aumenta sua taxa de transmissão
  - Observe que, considerando que cada roteador ao longo do caminho pode ativar o bit de advertência, o tráfego só aumenta quando nenhum roteador tiver problemas

Pacotes reguladores

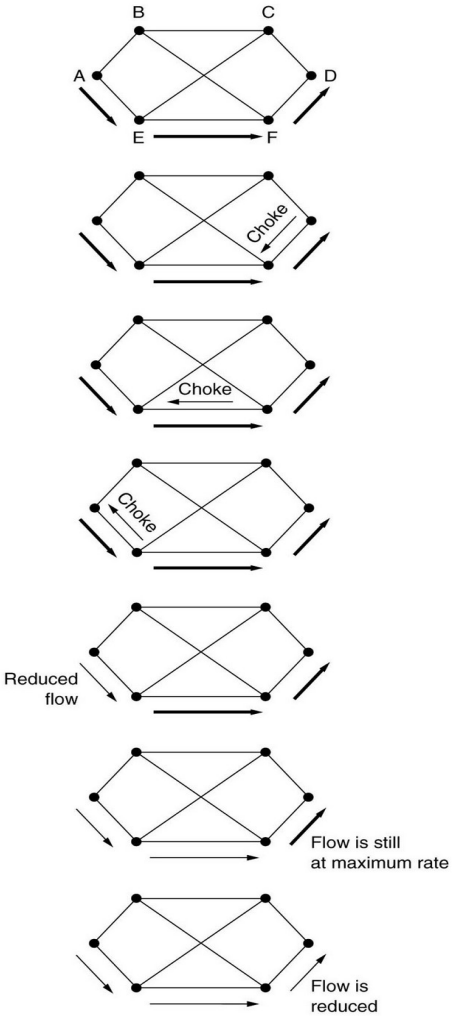
- ✔ O roteador enviará um pacote regulador ao host de origem, informando o destino encontrado no pacote
- ✔ O pacote original é marcado (um bit de cabeçalho é ativado) para que ele não venha a gerar mais pacotes reguladores ao longo do caminho
- ✔ Ao receber o pacote regulador, o host de origem é obrigado a reduzir em X% o tráfego enviado ao destino especificado
- ✔ Como outros pacotes com o mesmo destino provavelmente já estarão a caminho e irão gerar ainda mais pacotes reguladores, o host deve ignorar os pacotes reguladores que se refiram a esse destino por um intervalo de tempo fixo

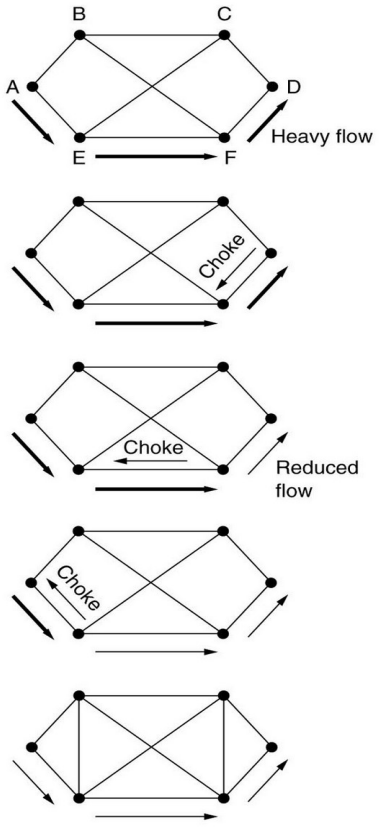
- ✔ Depois que esse tempo houver expirado, o host esperará mais pacotes reguladores para outro intervalo
  - se chegar um pacote, a linha ainda estará congestionada
    - o host reduzirá o fluxo ainda mais e começará novamente a ignorar pacotes reguladores
  - se não chegar nenhum outro pacote regulador durante o período de escuta
    - o host poderá aumentar o fluxo mais uma vez
- ✔ O feedback implícito desse protocolo pode ajudar a evitar congestionamento sem estrangular o fluxo, a menos que ocorra algum problema

- Os hosts reduzem o tráfego ajustando seus parâmetros de orientação como, por exemplo, o tamanho de sua janela
  - em geral, o primeiro pacote regulador faz a taxa de dados se reduzir a 0,50 de seu valor anterior
  - o seguinte causa uma redução de 0,25 e assim por diante
  - os aumentos são feitos em incrementos menores para impedir que voltem rapidamente a ocorrer congestionamentos
- Os roteadores podem manter diversos valores limites
  - dependendo do valor limite que tiver sido ultrapassado, o pacote regulador poderá conter uma advertência suave, uma advertência severa ou um ultimato
- Os roteadores podem usar comprimentos de filas ou buffers, em vez de linhas como sinal para ativar o processo
  - também é possível utilizar com essa métrica a mesma ponderação exponencial que foi usada com u

Pacotes reguladores hop a hop

- ✔ Em altas velocidades ou em longas distâncias, o envio de um pacote regulador para os hosts de origem não funciona bem porque a reação é muito lenta
- ✔ Uma abordagem alternativa é fazer com que o pacote regulador tenha efeito a cada hop pelo qual passar
- ✔ O efeito líquido desse esquema hop a hop é oferecer alívio rápido no ponto de congestionamento, ao preço de aumentar o consumo de buffers
- ✔ Dessa maneira, o congestionamento pode ser cortado pela raiz sem perda de pacotes


B
C
A
D
E
F
Choke
Choke
Choke
Reduced
flow
Flow is still
at maximum rate
Flow is
reduced


B
C
A
D
E
F
Heavy flow
Choke
Choke
Reduced
flow
Choke

1.8.5 – Escoamento de carga

- ✔ Considerado a artilharia pesada: quando nenhum dos anteriores tiver resolvido o problema
- ✔ O escoamento de carga acontece quando os roteadores estão sendo inundados por pacotes que não podem manipular: eles simplesmente devem descartá-los
- ✔ Um roteador que está sendo sufocado com pacotes pode simplesmente selecionar ao acaso aqueles que deverão ser descartados
- ✔ Ou, o pacote a ser descartado pode depender das aplicações em execução
  - transferência de arquivos → um pacote antigo vale mais que um novo
  - multimídia → um pacote novo é mais importante que um antigo
- ✔ “antigo é melhor que novo” → política do vinho
- ✔ “novo é melhor que antigo” → política do leite

- Numa política de descarte inteligente, as aplicações devem marcar seus pacotes por classes de prioridade
  - quando os pacotes tiverem de ser descartados, os roteadores poderão descartar primeiro os pacotes da classe mais baixa, depois os da classe mais baixa seguinte e assim por diante
    - É claro que, a menos que exista algum incentivo especial para marcar os pacotes com algo diferente de MUITO IMPORTANTE — NUNCA DESCARTAR, ninguém o fará
- O incentivo poderia vir sob a forma de dinheiro:
  - transmissão dos pacotes de baixa prioridade mais econômica que o envio dos pacotes de alta prioridade
  - uma alternativa seria permitir que os transmissores enviassem pacotes de alta prioridade em condições de carga leve mas, à medida que a carga aumentasse, eles seriam descartados, encorajando assim os usuários a interromperem a transmissão desses quadros
  - permitir que os hosts excedam os limites especificados no circuito virtual
    - ficam sujeitos à condição de que todo o tráfego em excesso seja marcado como tráfego de baixa prioridade

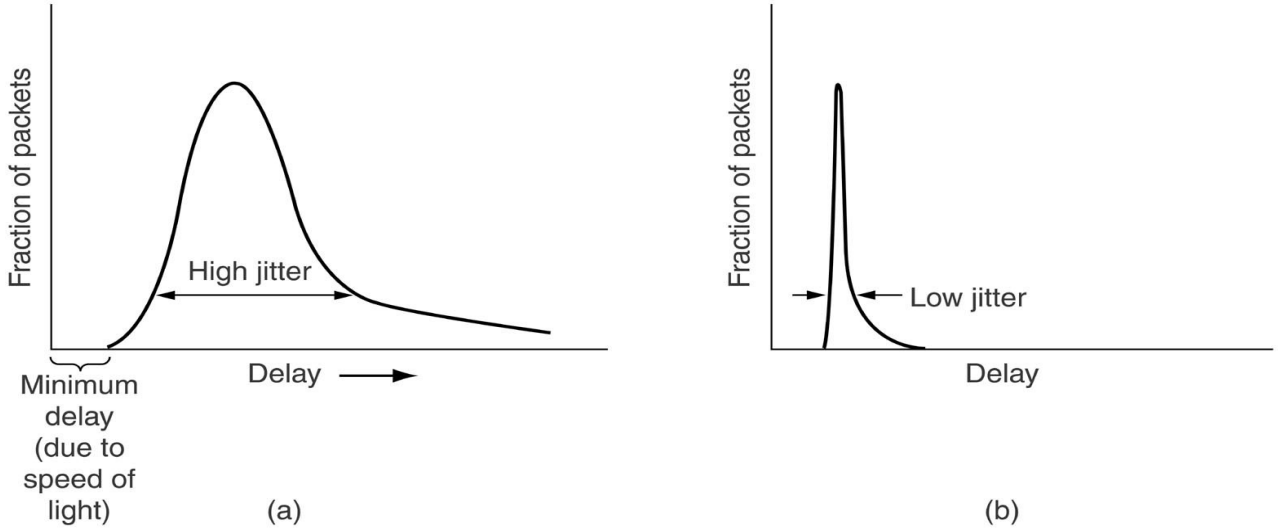
Detecção aleatória prematura

- ✔ Lidar com o congestionamento logo que se inicie é mais eficaz do que permitir que o congestionamento se consolide e depois tentar lidar com ele
  - descartar pacotes antes que todo o espaço dos buffers realmente se esgote
  - em alguns protocolos de transporte (inclusive o TCP), a resposta à perda de pacotes é tornar a origem mais lenta
    - o TCP foi projetado para redes fisicamente conectadas e essas redes são muito confiáveis
    - a perda de pacotes se deve muito mais ao estouro de buffers do que a erros de transmissão.
  - a ideia consiste em ter tempo para empreender alguma ação antes de ser tarde demais
  - para determinar quando começar a descartar pacotes:
    - os roteadores mantêm uma média atualizada dos comprimentos de suas filas
    - quando o comprimento médio da fila em alguma linha excede um limite, considera-se que a linha está congestionada e é executada uma ação para solucionar o problema

- O roteador provavelmente não poderá detectar a origem que está causando a maior parte do problema
  - ele escolhe um pacote ao acaso na fila que disparou a ação
- Como o roteador deve fornecer informações à fonte sobre o problema?
  - enviar um pacote regulador, conforme descrito antes
    - um problema que surge é que ela impõe ainda mais carga à rede já congestionada
  - uma estratégia diferente é simplesmente descartar o pacote selecionado e não informar sobre o fato
    - a origem notará a falta de confirmação e executará alguma ação
    - como ela sabe que a perda de pacotes em geral é causada pelo congestionamento e por descartes, a origem responderá diminuindo a taxa de transmissão, em vez de continuar a tentar transmitir com maior intensidade
    - essa forma implícita de feedback só funciona quando as origens respondem à perda de pacotes reduzindo sua taxa de transmissão
    - Em redes sem fios, nas quais a maioria das perdas se deve ao ruído no enlace aéreo, essa abordagem não pode ser usada

1.8.6 – Controle de flutuação

- Para aplicações como a transmissão de áudio e vídeo, não importa muito se os pacotes demorarem 20 ms ou 30 ms para serem entregues, desde que o tempo em trânsito seja constante
- A variação (isto é, o desvio padrão) nos tempos de chegada de pacotes é chamada flutuação (jitter)
  - e.g.: uma flutuação elevada, na qual alguns pacotes demoram 20 ms e outros demoram 30 ms para chegar, resultará em uma qualidade irregular do som ou do filme
  - e.g: um acordo em que 99% dos pacotes fossem entregues com um retardo no intervalo de 24,5 ms a 25,5 ms poderia ser aceitável
- O valor médio escolhido deve ser viável
  - deve levar em conta o tempo de trânsito na velocidade da luz e o retardo mínimo na passagem pelos roteadores, e deixar uma pequena folga para alguns retardos inevitáveis



- A flutuação pode ser limitada pelo cálculo do tempo de trânsito esperado para cada hop ao longo do caminho
  - quando um pacote chega a um roteador, este verifica se o pacote está adiantado ou atrasado em sua programação
  - essas informações são armazenadas no pacote e atualizadas a cada hop
    - se estiver adiantado, o pacote será retido um tempo suficiente para que seja sincronizado
    - se estiver atrasado, o roteador tentará enviá-lo rapidamente (furando fila)

- O algoritmo pode sempre escolher o pacote que estiver mais atrasado para ser enviado
  - pacotes que estão adiantados na programação têm sua taxa de transmissão reduzida
  - pacotes que estão atrasados são acelerados
  - em ambos os casos, essa ação reduz o volume de flutuação
- Em algumas aplicações, como vídeo por demanda, a flutuação pode ser eliminada pelo armazenamento em buffer no receptor
- Para outras aplicações que exigem interação em tempo real entre pessoas, como telefonia da Internet e videoconferência, o retardo inerente do armazenamento em buffer não é aceitável