Redes de Computadores

Encaminhamento com Base em Inundação

(1) Primeira parte

Objetivos do Capítulo

- O encaminhamento numa rede com a dimensão da Internet é muito complexo pois a mesma tem uma dimensão gigantesca
- Por isso na Internet as diferentes sub-redes usam técnicas de encaminhamento independentes
- No entanto, em redes pequenas, é possível usar soluções mais simples de encaminhamento
- Neste capítulo iremos ver a mais simples de todas, o encaminhamento com base em inundação

Increasingly, people seem to misinterpret complexity as sophistication, which is baffling – the incomprehensible should cause suspicion rather than admiration.

– **Autor:** *Niklaus Wirth*

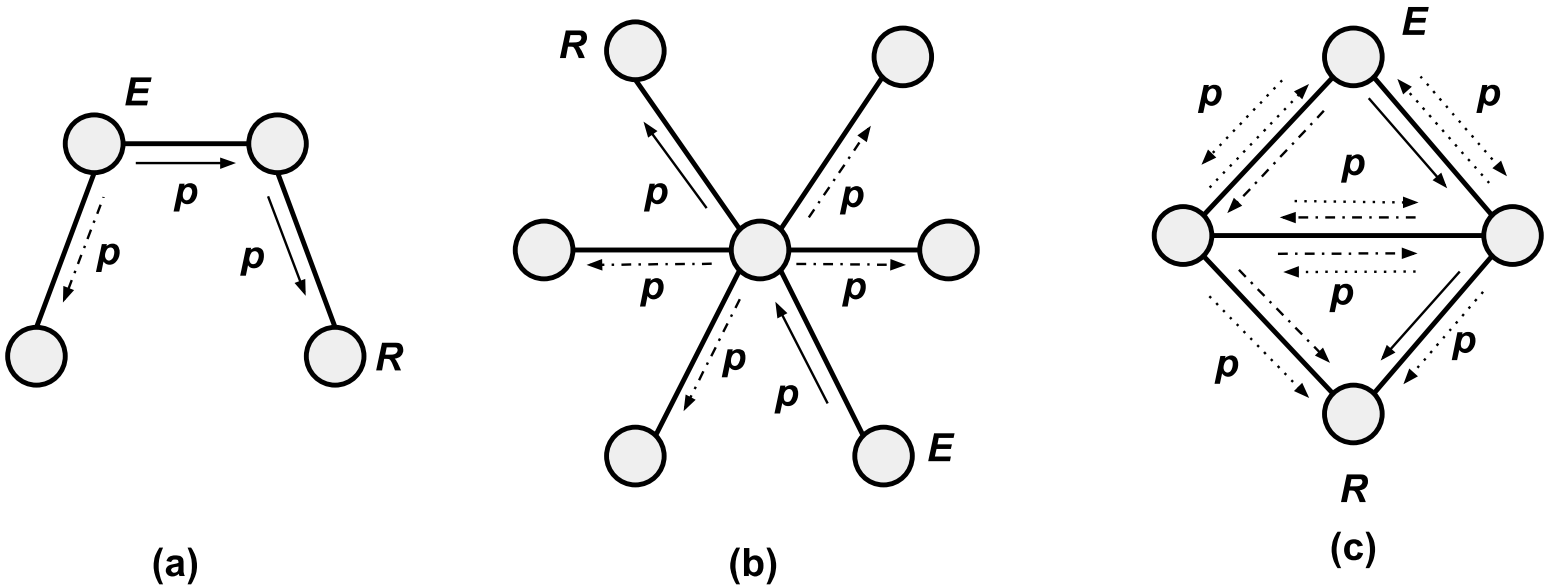
KISS – Keep It Simple, Stupid !

- Admitindo que um comutador não sabe nada sobre a rede ou a localização do nó de destino, qual o algoritmo de encaminhamento mais simples?

- Como o emissor não conhece onde está o destino, envia para todas as interfaces menos pela que recebeu

- Esse algoritmo chama-se algoritmo de **inundação ou *flooding***

Inundação



- A inundação pode introduzir desperdício, ver (a) e (b), ou mesmo redundância, isto é, duplicados (c), sempre que a rede tem ciclos

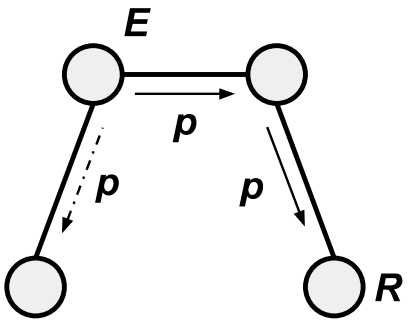
Análise do Algoritmo

- É robusto pois explora todos os caminhos disponíveis
  - a cópia que chega primeiro ao destinatário chega pelo caminho mais curto
  - Mesmo que o destinatário se desloque, localiza-o facilmente
- Implementa implicitamente a difusão (*broadcasting*)
- Não se conhece nenhum outro mais simples
- Defeitos
  - Quando o destinatário é único (*unicasting*) envia mensagens a mais
  - É um método promíscuo pois todos os nós vêm todas as mensagens
  - O mecanismo para deteção de duplicados é pesado nos requisitos de memória e de cabeçalhos
  - Mas o mecanismo de deteção de duplicados pode servir de base à construção de um algoritmo de difusão fiável

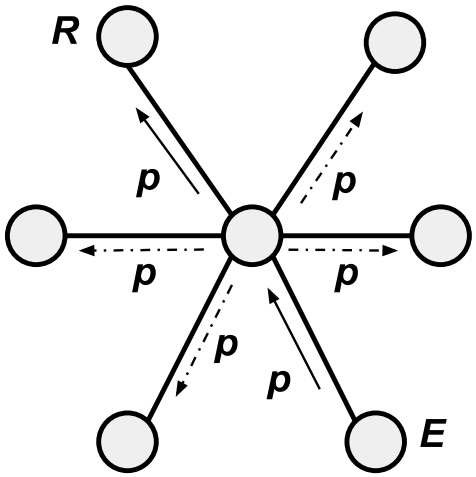
Inundação numa árvore

Mas se a rede tiver a configuração de uma árvore, (a) e (b), a inundação não introduz duplicados.

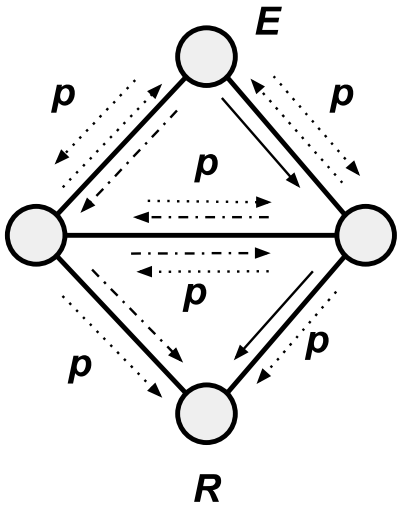
Apenas mensagens inúteis quando se pretende comunicar em *unicasting* (de um emissor para um único recetor)



(a)



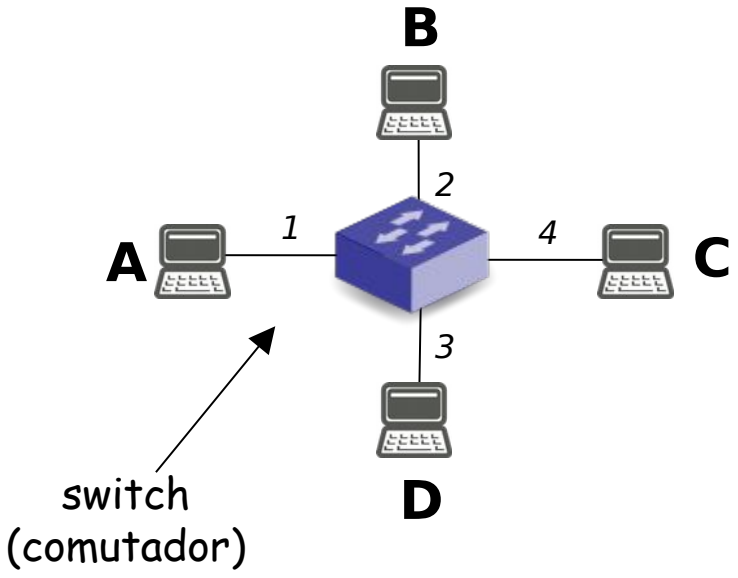
(b)



(c)

Exemplo: *Switches Ethernet*

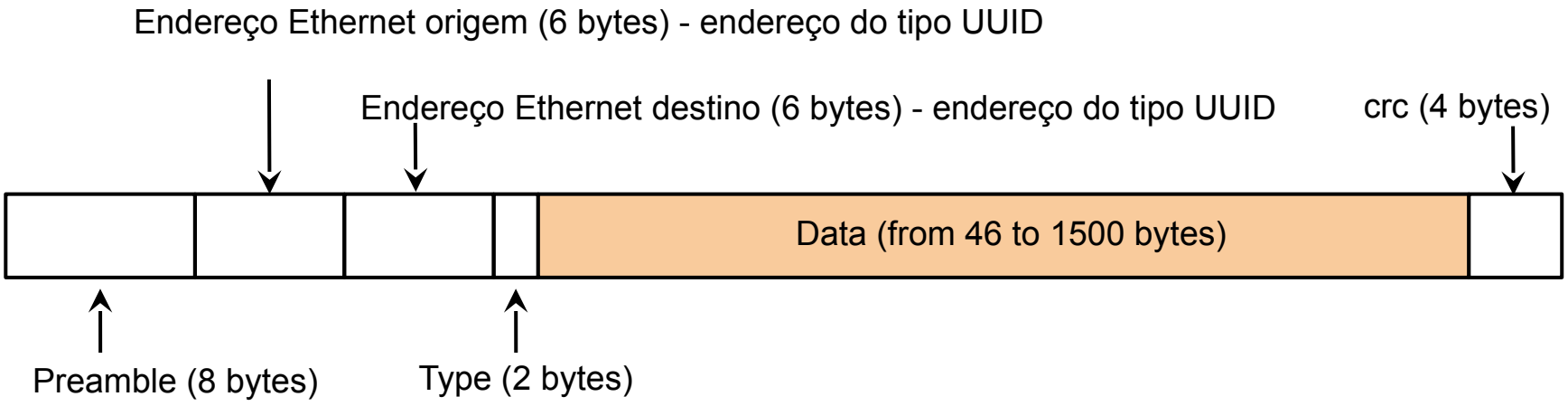
- Computadores com interfaces com ligações dedicadas a portas de um comutador especial designado *switch Ethernet* ou simplesmente *switch*
- O *switch*, como todos os comutadores, faz *store and forward*
- Trabalha com qualquer tipo de endereços desde que cada interface tenha um endereço diferente
- No cenário típico, os computadores têm interfaces ethernet com *MAC addresses* distintos



Switch com 4 interfaces (1,2,3,4)

Ethernet *Frame* Format

Originalmente definida para canais de difusão 802.3



É Possível Fazer Melhor?

- Sim, com aprendizagem pelo caminho inverso
- Quando um *frame* passa, anota-se de que lado está o emissor
- Para a próxima não é necessário enviar para todos os nós

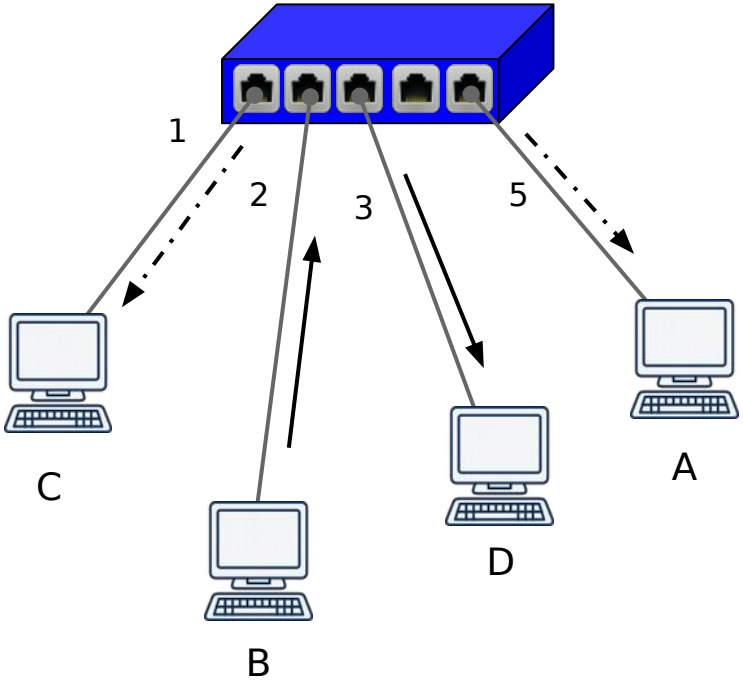
Inundação com auto aprendizagem

- Método usado pelos *switches ethernet* com base numa tabela dita *MAC-Address Table*
  - Cada entrada associa um endereço de nível MAC a uma interface
  - Cada entrada tem associado um TTL
  - Quando o TTL expira (30 a 60 segundos) a entrada é suprimida
- Inicialmente a *MAC-Address Table* está vazia
  - No entanto, sempre que o *switch* recebe um pacote
  - Pode introduzir na tabela o seu endereço MAC de origem e colocá-lo na tabela associado à interface por onde foi recebido

Aprendizagem Pelo Caminho Inverso

Address	Interface	TTL
B	2	120

Address	Interface	TTL
B	2	115
D	3	120



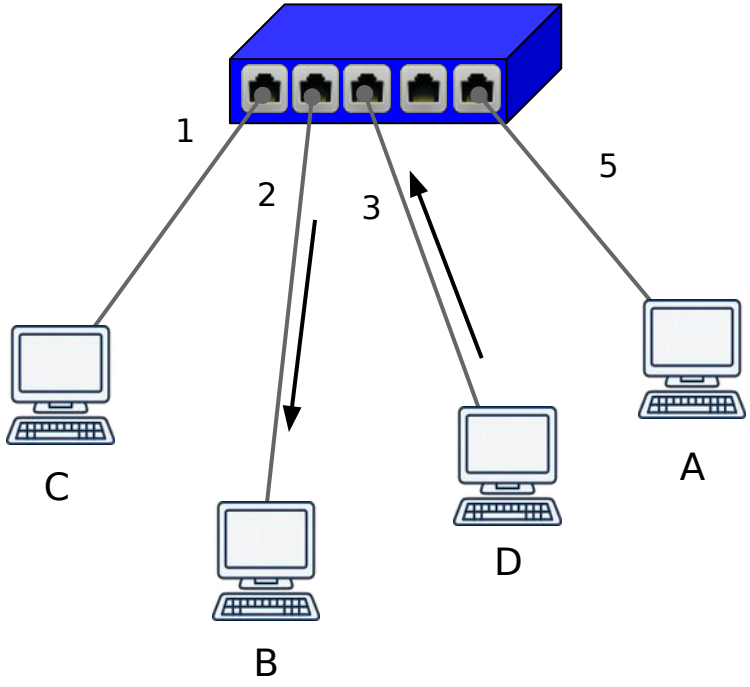
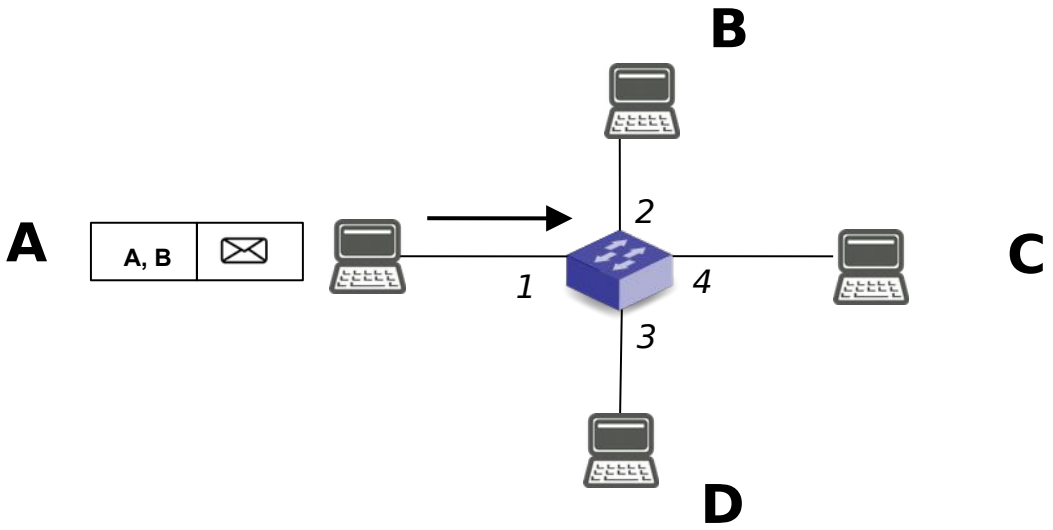


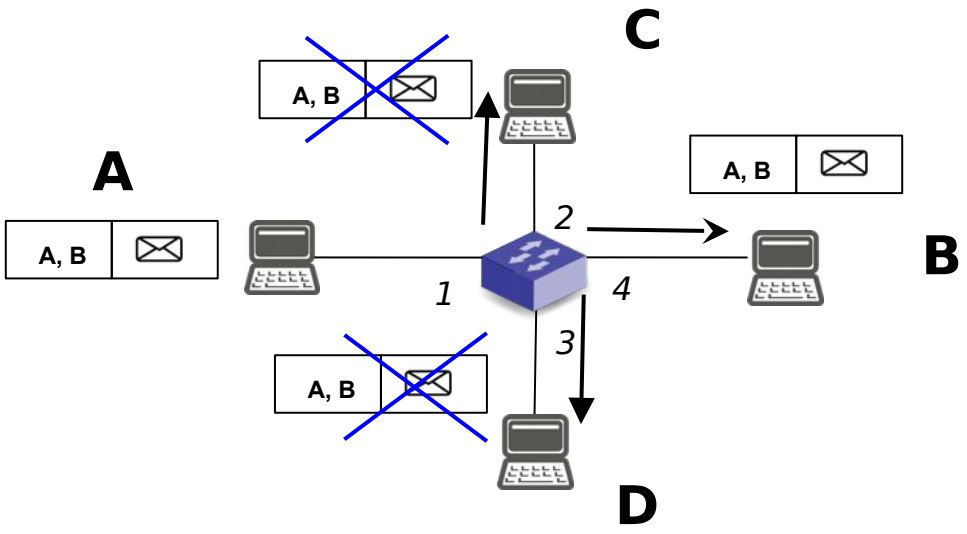
Tabela de Auto-Aprendizagem

- Quando um *frame* chega
  - Obter o endereço origem
  - Associá-lo à interface de entrada
  - Memorizar a associação na tabela de auto-aprendizagem
  - Usar um TTL para esquecer associações muito antigas



Fazer Inundação por Defeito

- Quando um *frame* chega e não se conhece o destino, fazer inundação
  - Enviar uma cópia exceto pela interface de onde veio
  - Felizmente não vai ser necessário fazê-lo com frequência

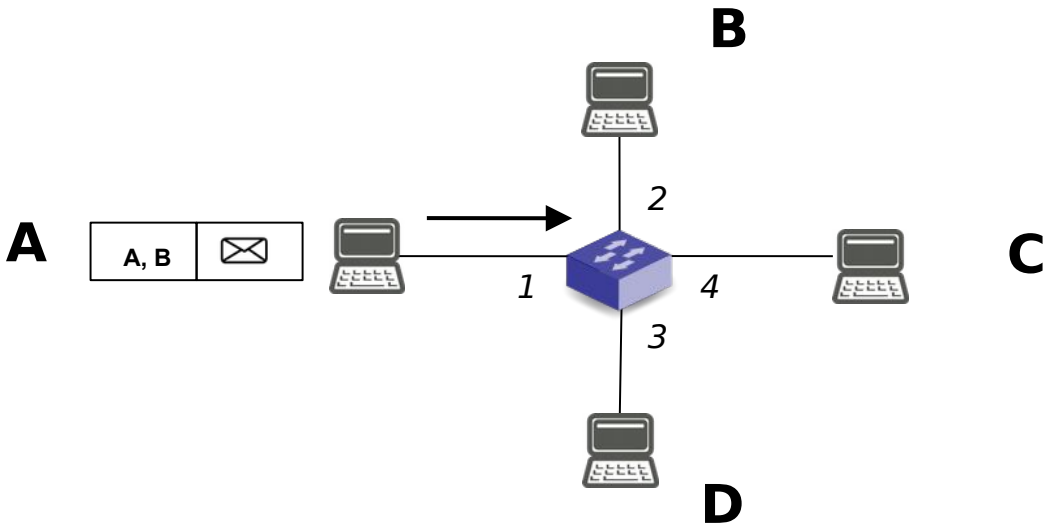


Exemplo: *A* envia para B

Mac Address table
(inicialmente vazia)

MAC addr	interface	TTL
A	1	60

B é desconhecido, *flood*

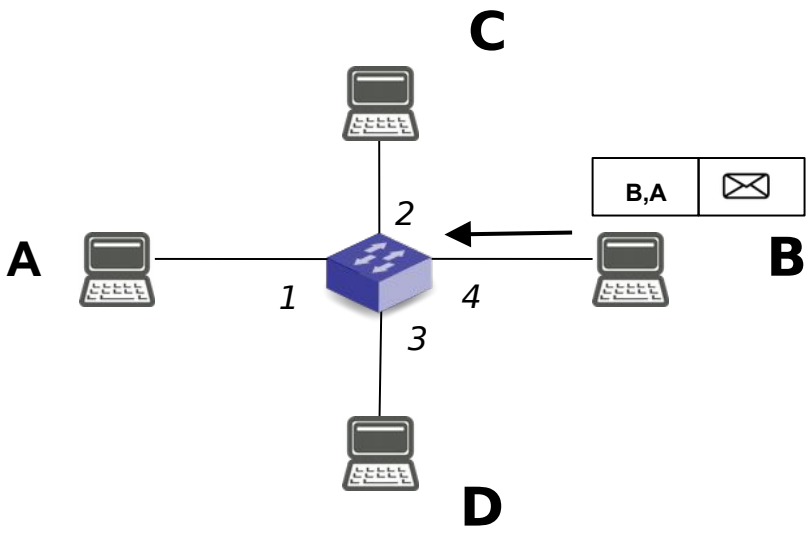


B responde a A

Mac Address table

MAC addr	interface	TTL
A	1	59
B	4	60

A é conhecido, enviar via a interface 1



Algoritmo

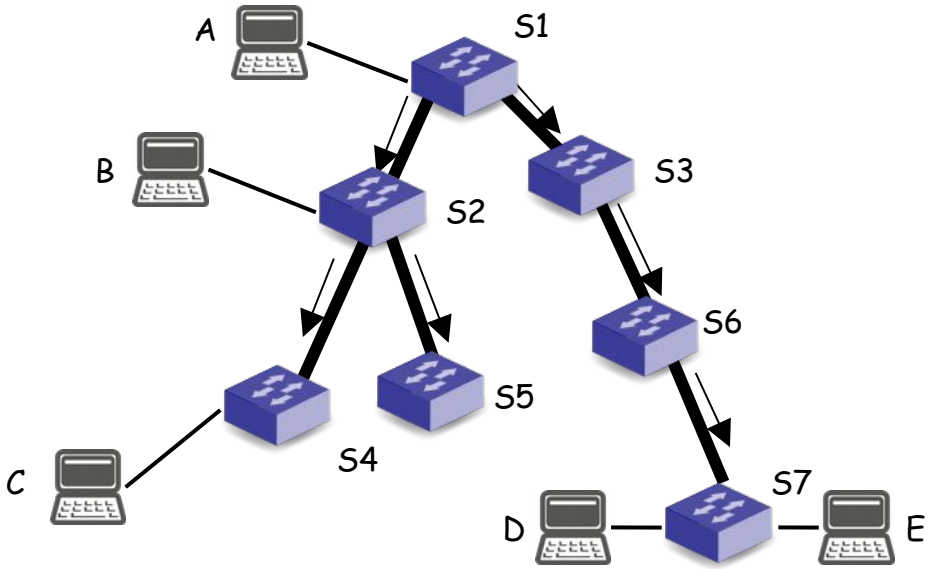
```
TTL= 60 // for example

processPacket ( packet p, interface in ) {
      macTable.put(p.getOrigin(), in, TTL)
      if ( p.getDestination == self.getID() ) {
            // packet p got to its destination
            locally process packet p
            return
            interface out = macTable.get(p.getDestination)
            if ( out == null ) flood(p)
            else if ( out != in ) out.send(p)
            // else ignore p
      }
}
```

Temporizadores

- Quando se introduzem entradas nas tabelas (*Mac Address Tables*) usam-se temporizadores que controlam quanto tempo é que cada entrada pode lá ficar (se não for refrescada)
  - Isso evita que as tabelas cresçam eternamente
  - Permite que as tabelas sejam mais pequenas que o número total de computadores existentes
  - Será que o mesmo endereço MAC pode estar associado a diferentes portas?
  - A que porta está associado o endereço de difusão? (FF FF FF FF FF FF)

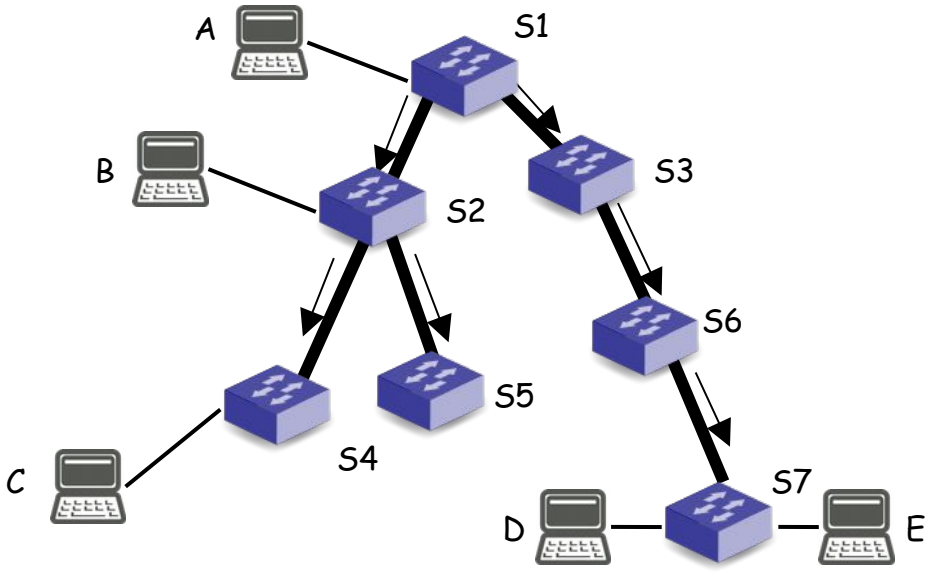
O Algoritmo Escala

- Quando o computador *A* envia um *frame*, caso o destino não seja conhecido, o *frame* é difundido por inundação e todos os *switches* passam a localizar o emissor



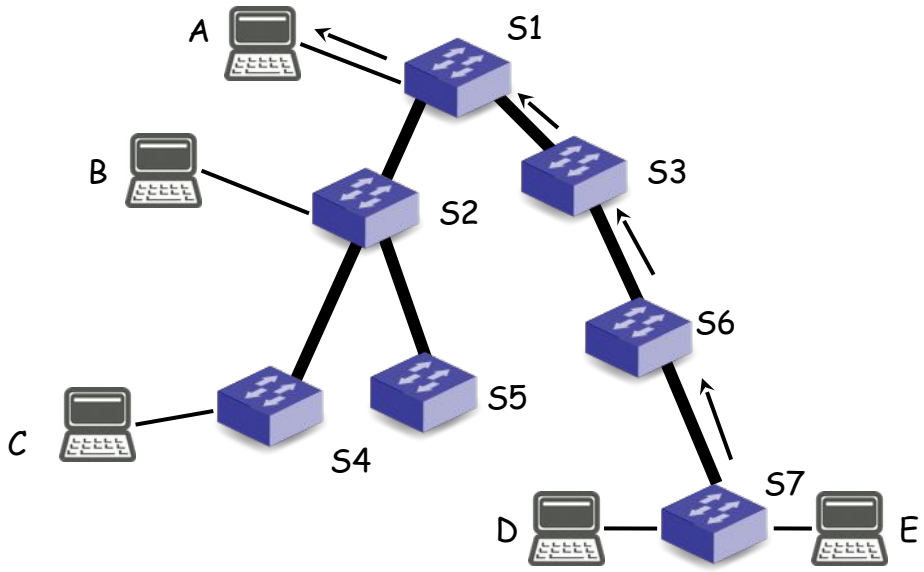
Exemplo: *A* envia a E

- Se o computador *A* envia um *frame* para E, como o destino é desconhecido, o *frame* é difundido por inundação, mas todos os *switches* passam a localizar *A*

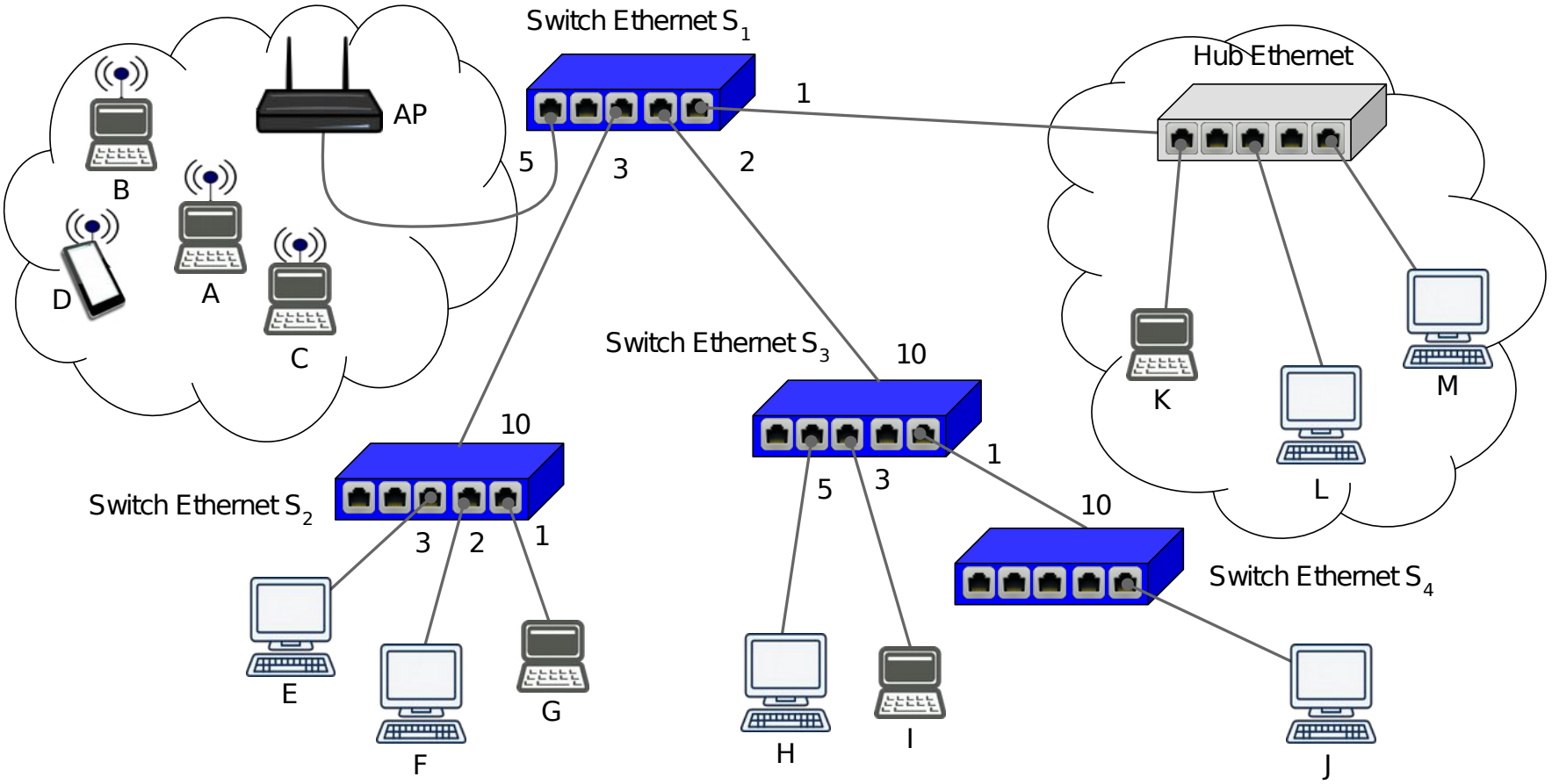


E responde a A

- Se o computador E responde a A, o caminho direto para A já é conhecido



Configuração Típica



Análise e conclusões

- Numa rede estruturada em árvore e em que seja realista utilizar inundação, é possível encaminhar mensagens sem necessidade de nenhuma parametrização prévia. É equivalente ao funcionamento de um canal único baseado em difusão
  - O método suporta até que os computadores se desloquem (sejam móveis)
  - Exige-se apenas que todos os endereços sejam diferentes
  - E que fazer inundação com alguma frequência seja realista e que a rede não tenha falhas (os canais não avariam !)
- Será realista fazê-lo numa rede com a configuração de uma malha e caminhos redundantes?
  - Com vários milhares de computadores?
  - Cobrindo um país ou mesmo do mundo?
- A resposta é NÃO !

Conclusões

- Uma das propriedades interessantes de qualquer solução de um problema de engenharia é a simplicidade
- Fazer encaminhamento de pacotes usando difusão é muito simples
  - Não requer que os comutadores necessitem de muita informação
  - Permite aos comutadores adaptarem-se automaticamente
- Introduzir otimização por auto aprendizagem pelo caminho inverso torna o método ainda mais atrativo
- Infelizmente
  - Só é realista em redes organizadas em árvore
  - Pode não ser muito escalável sempre que a rede tem muitos computadores ou esta é de grande dimensão