

# Redes de Computadores 1

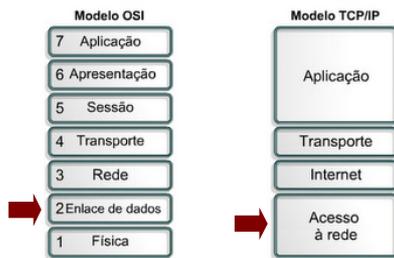
Prof. Miguel Elias Mitre Campista

<http://www.gta.ufrj.br/~miguel>

# Parte IV

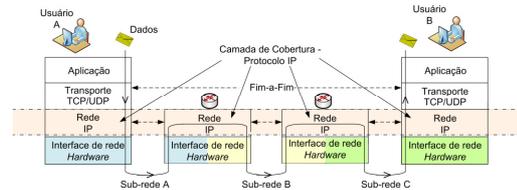
Camada de Enlace: Introdução

## Camada de Enlace



## Rede X Enlace

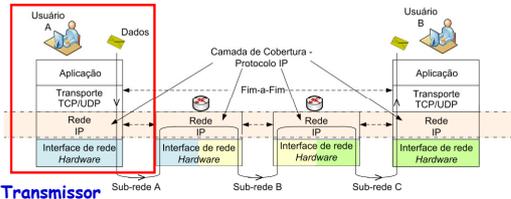
- Protocolos da camada de rede
  - Executados nos sistemas finais e nos roteadores



Transporta segmentos da estação remetente à receptora

## Rede X Enlace

- Protocolos da camada de rede
  - Executados nos sistemas finais e nos roteadores

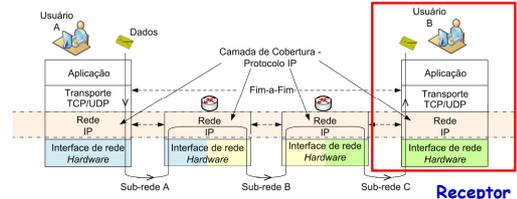


Transmissor

No lado transmissor, encapsula segmentos dentro de datagramas

## Rede X Enlace

- Protocolos da camada de rede
  - Executados nos sistemas finais e nos roteadores

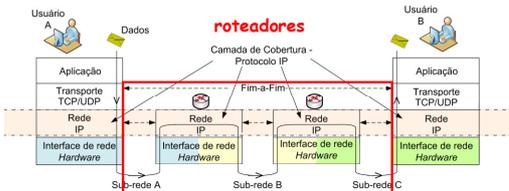


Receptor

No lado receptor, entrega os segmentos para a camada de transporte

## Rede X Enlace

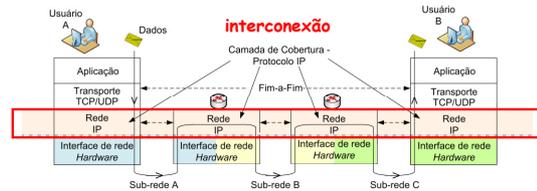
- Protocolos da camada de rede
  - Executados nos sistemas finais e nos roteadores



Roteadores examinam campos de cabeçalho de todos os datagramas IP que passam por eles e decidem o próximo salto a ser seguido

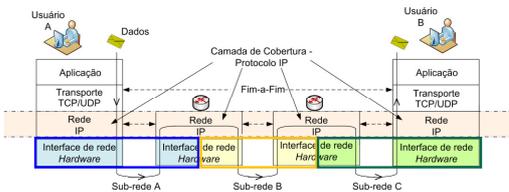
## Rede X Enlace

- Protocolos da camada de rede
  - Executados nos sistemas finais e nos roteadores



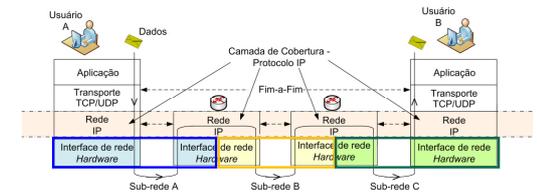
## Rede X Enlace

- Protocolos da camada de rede
  - Executados nos sistemas finais e nos roteadores



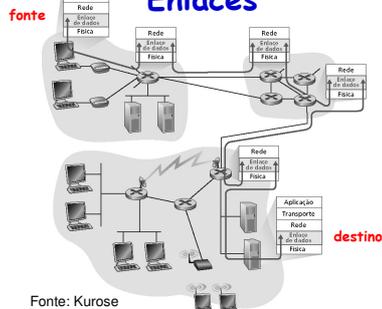
Cada enlace pode ter um protocolo diferente e cada protocolo pode ser utilizado em um enlace com características totalmente diferentes (p.ex. um enlace é cabeado e outro é sem-fio)

- Protocolos da camada de rede
  - Executados nos sistemas finais e nos roteadores



Serviço salto-a-salto, em contraste com o serviço fim-a-fim

## Transmissão de Dados pelos Enlaces



Fonte: Kurose

## Camada de Rede

- Responsável por:
  - Determinar o melhor caminho para o envio dos pacotes
    - É função dos protocolos de roteamento
  - Encaminhar os pacotes até o destino
    - É função do protocolo IP
  - Interconectar redes de diferentes tecnologias
    - É função do protocolo IP

## Camada de Rede

- Responsável por:
  - **Determinar o melhor caminho** para o envio dos pacotes
    - É função dos protocolos de roteamento
  - **Encaminhar** os pacotes até o destino
    - É função do protocolo IP
  - **Interconectar** redes de diferentes tecnologias
    - É função do protocolo IP

Um caminho é composto por um ou mais **enlaces** (*links*)



Como os dados são transmitidos em cada enlace?

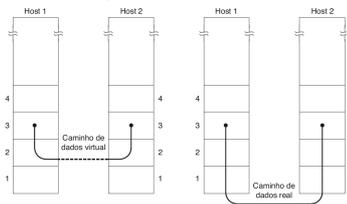
## Camada de Enlace

- Presta serviço para a camada de rede
  - **Serviço básico:** Prover comunicação eficiente e confiável de unidades de informação entre dois nós adjacentes
    - **Unidade de informação:** Pacote de camada de enlace, também chamado de quadro (*frame*)
      - Quadros encapsulam datagramas da camada de rede
    - **Nós adjacentes:** Nós conectados fisicamente por um canal de comunicação, também chamado de enlace
      - Enlace entrega bits ao destinatário na mesma ordem de envio



## Camada de Enlace

- Usa serviços prestados pela camada física
  - Serviço necessário para envio e recebimento de bits pelo canal de comunicação



## Camada de Enlace

- Além do serviço básico...
  - Enquadramento (*framing*)
  - Entrega confiável
  - Controle de fluxo
  - Detecção e correção de erros
  - Transmissão *half-duplex* ou *full-duplex*
  - Controle de acesso ao meio

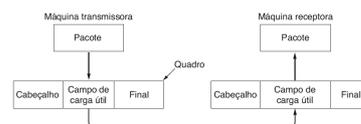
Protocolos de camada de enlace são responsáveis por oferecer tais serviços!

## Camada de Enlace

- Para oferecer serviços, os protocolos devem considerar...
  - Tipos diferentes de canais de comunicação:
    - **Canal ponto-a-ponto**
      - Uma estação em cada extremidade
      - Requer controle simples de acesso
        - » Exs.: Redes de acesso domiciliares e redes entre roteadores
    - **Canal de difusão (*broadcast*)**
      - Várias estações conectadas ao mesmo canal
      - Requer controle de acesso ao meio para coordenar as transmissões
        - » Ex. rede sem-fio

## Camada de Enlace

- Dependendo dos serviços e do tipo de canal...
  - Protocolo deve realizar o gerenciamento de quadros
    - Constitui uma tarefa importante para o serviço prestado para a camada de rede
      - Cada quadro possui um **cabeçalho**, um campo de **carga útil** (*datagrama*) e um marcador de **final**



## Protocolos de Camada de Enlace

- Transportam datagramas salto-a-salto em um enlace individual
  - Devem definir quais os serviços são oferecidos à camada de rede, levando em conta:
    - Tipo de canal de comunicação
      - Ponto-a-ponto ou difusão
    - Gerenciamento de quadros
      - Formato dos quadros

## Serviços da Camada de Enlace

## Serviços da Camada de Enlace

- Enquadramento
  - Delimita onde começa e onde termina um quadro
  - Encapsula um datagrama em um quadro
    - Adiciona cabeçalho e fim de quadro (trailer)
- Entrega confiável entre nós adjacentes
  - Similar aos mecanismos da camada de transporte
  - Pouco usada em canais com baixas taxas de erro
    - Ex.: fibra óptica, alguns tipos de pares trançados, etc.
  - Necessária em canais com altas taxas de erro
    - Ex.: canais sem-fio

## Serviços da Camada de Enlace

- Controle de fluxo
  - Compatibilizar taxas de produção e consumo de quadros entre remetentes e receptores
- Detecção de erros
  - Erros são causados por atenuação do sinal e por ruído
    - Receptor detecta presença de erros
      - Sinaliza ao remetente a retransmissão ou descarta o quadro
- Correção de erros
  - Permite que o receptor localize e corrija o(s) erro(s) sem precisar da retransmissão

## Serviços da Camada de Enlace

- Half-duplex e full-duplex
  - Com half-duplex um nó não pode transmitir e receber pacotes ao mesmo tempo
  - Com full-duplex um nó pode transmitir e receber pacotes ao mesmo tempo
- Controle de acesso ao meio
  - Implementa o controle de acesso ao canal se meio for compartilhado
  - Endereços físicos (MAC) são usados nos cabeçalhos dos quadros para identificar origem e destino de quadros em enlaces multiponto
    - Endereços diferentes do endereço IP

## Serviços da Camada de Enlace

- Funções similares: enlace e transporte
  - Para que confiabilidade nas duas camadas?
  - Para que controle de fluxo nas duas camadas?

Camada de transporte: fim-a-fim  
X  
Camada de enlace: salto-a-salto

## Classificação dos Serviços

- Em geral, há três tipos de serviços providos
  - Não orientado a conexões e sem confirmação
  - Não orientado a conexões e com confirmação
  - Orientado a conexões e com confirmação

## Classificação dos Serviços

- **Não orientado a conexões e sem confirmação**
  - Apropriado quando a taxa de erros é baixa
    - **Ethernet**
  - Recuperação de perdas a cargo das camadas superiores
    - **Camada de transporte, por exemplo**
  - Apropriado para tráfego de tempo real
    - **Maior parte das redes locais usa um serviço desse tipo**

## Classificação dos Serviços

- **Não orientado a conexões e com confirmação**
  - Quadros são numerados
  - Usa temporizadores para implementar a confiabilidade
    - **Usado atualmente em redes sem fio (WiFi)**
      - **Essas redes possuem canais não confiáveis**
- **Orientado a conexões e com confirmação**
  - Há estabelecimento de conexão
  - Quadros são numerados
  - Usa temporizadores para implementar a confiabilidade
    - **Usado em redes onde a retransmissão é custosa**
      - **Redes via satélite**

## Classificação dos Serviços

- **Confirmação na camada enlace**
  - Questão de otimização, já que pode ser feita também em camadas superiores
    - **Problema da fragmentação dos pacotes em quadros**
      - **Retransmissão pode levar muito tempo se a confirmação for apenas fim-a-fim e não salto-a-salto**

## Enquadramento

- **Por que é preciso?**
  - Serviço provido pela camada física não garante que o fluxo de bits seja livre de erros
    - **Por exemplo:**
      - **Bits recebidos podem ter valores diferentes dos bits transmitidos**
      - **Número de bits pode até ser maior ou menor do que o número de bits transmitidos!**

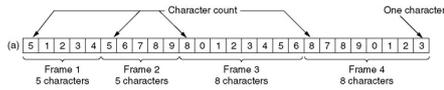
## Enquadramento

- **Camada de enlace:**
  - Divide o fluxo de bits em quadros
    - **Para isso, precisa marcar o início e o fim**
  - Faz verificação de erros por quadro
    - **Deteção e correção de erro, quando possível**
- **Métodos para marcação de início e o fim de quadros**
  - Contagem de caracteres
  - Inserção de bytes de *flags*
  - Inserção de *flags* no início e no final do quadro
  - Etc.

## Enquadramento: Contagem de caracteres

- Usa um campo no cabeçalho para especificar o número de caracteres do quadro

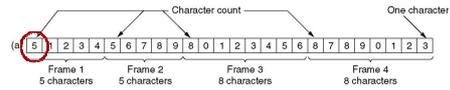
Exemplo de contagem de caracteres (fonte: Tanenbaum)



## Enquadramento: Contagem de caracteres

- Usa um campo no cabeçalho para especificar o número de caracteres do quadro

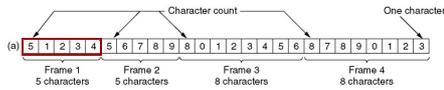
Exemplo de contagem de caracteres (fonte: Tanenbaum)



## Enquadramento: Contagem de caracteres

- Usa um campo no cabeçalho para especificar o número de caracteres do quadro

Exemplo de contagem de caracteres (fonte: Tanenbaum)

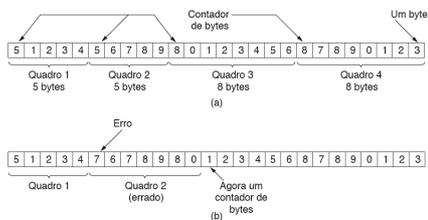


## Enquadramento: Contagem de caracteres

- Problema
  - Contagem pode ser adulterada por um erro de transmissão...
  - Mesmo com a verificação incorreta, destino não sabe onde começa o próximo quadro
  - Solicitação de retransmissão também não adianta
    - Destino não sabe quantos caracteres devem ser ignorados para chegar ao início da retransmissão, já que não há marcação de início e fim de quadro

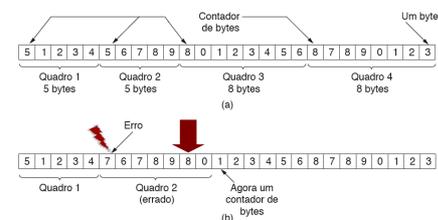
## Enquadramento: Contagem de caracteres

Exemplo de contagem de caracteres com um erro (fonte: Tanenbaum)



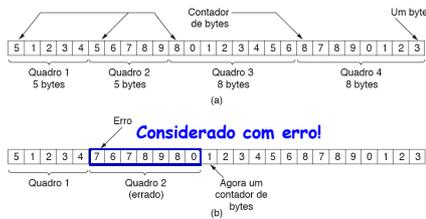
## Enquadramento: Contagem de caracteres

Exemplo de contagem de caracteres com um erro (fonte: Tanenbaum)



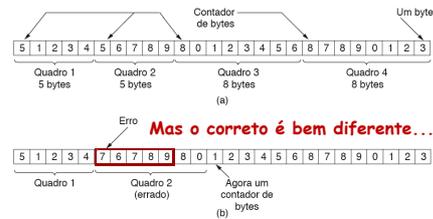
## Enquadramento: Contagem de caracteres

Exemplo de contagem de caracteres com um erro (fonte: Tanenbaum)



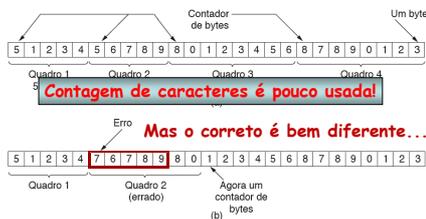
## Enquadramento: Contagem de caracteres

Exemplo de contagem de caracteres com um erro (fonte: Tanenbaum)



## Enquadramento: Contagem de caracteres

Exemplo de contagem de caracteres com um erro (fonte: Tanenbaum)



## Enquadramento: Bytes de flags

- Soluciona o problema de resincronização após um erro
  - Quadro começa e termina com bytes especiais
    - Bytes de flags são usados para delimitar o início e o fim de um quadro

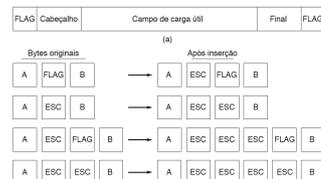


## Enquadramento: Bytes de flags

- Problema:
  - O Payload pode conter os bytes de flags
    - Se for composto por dados binários, sincronismo é perdido
- Solução
  - Transmissor da camada de enlace introduz um caractere de escape especial (ESC) antes de cada byte de flag "acidental" nos dados
    - Técnica chamada de inserção de octetos ou inserção de caracteres
      - Usada no protocolo PPP (Point-to-Point Protocol)

## Enquadramento: Bytes de flags

Sequências de quadros com octetos de flags (fonte: Tanenbaum)



## Enquadramento: Bytes de flags

Sequências de quadros com octetos de flags (fonte: Tanenbaum)



**Problema: Depende do uso de caracteres de 8 bits**

## Enquadramento: Flags iniciais e finais

- Vantagem
  - Dados podem ter um número arbitrário de bits
- Cada quadro começa e termina com uma sequência padrão de bits
  - Ex.: Delimitador → 01111110 (Um 0, seis 1's e outro 0)
  - Transmissor
    - Quando encontra cinco bits 1's consecutivos nos dados, insere um bit 0 após a sequência → Inserção de bits
  - Receptor
    - Ao receber cinco bits 1's seguidos por um bit 0, remove o bit 0

## Enquadramento: Flags iniciais e finais

Exemplo de inserção de bits (fonte: Tanenbaum)

(a) 011011111111111111110010

(b) 01101111011111011111010010

↑

Bits inseridos

(c) 011011111111111111110010

- (a) Dados originais  
 (b) Dados transmitidos  
 (c) Dados recebidos após a remoção dos bits

## Enquadramento: Flags iniciais e finais

Exemplo de inserção de bits (fonte: Tanenbaum)

(a) 011011111111111111110010

(b) 0110111101111011111010010

↑

Bits inseridos

(c) 011011111111111111110010

- (a) Dados originais  
 (b) Dados transmitidos  
 (c) Dados recebidos após a remoção dos bits

## Enquadramento: Flags iniciais e finais

Exemplo de inserção de bits (fonte: Tanenbaum)

(a) 011011111111111111110010

(b) 0110111101110111010010

↑

Bits inseridos

(c) 011011111111111111110010

- (a) Dados originais  
 (b) Dados transmitidos  
 (c) Dados recebidos após a remoção dos bits

## Enquadramento: Flags iniciais e finais

- Se o receptor perder a sincronização
  - Basta procurar pelo padrão de bits

## Controle de Fluxo

- Um nó quer transmitir a uma taxa maior do que a taxa que o outro pode receber
  - Uma das máquinas é mais lenta que a outra
    - Ex. Computador de última geração X smartphone
- Duas abordagens mais comuns
  - Controle de fluxo baseado em realimentação
    - Abordagem mais usada
  - Controle de fluxo baseado na velocidade
    - Mecanismo interno limita a velocidade dos dados
    - Não usa realimentação do receptor
    - Não é utilizado na camada de enlace

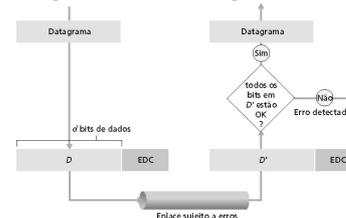
## Controle de Erros

- Como garantir que os quadros enviados foram recebidos ordenadamente?
  - Mais comum é dar ao transmissor alguma realimentação sobre o que está se passando do outro lado
    - Reconhecimentos positivos e negativos
  - Além disso, usam-se temporizadores
    - Espera pela confirmação durante um tempo
  - Números de sequência também são usados
    - Várias cópias do mesmo quadro podem ser recebidas
      - Ex.: Reconhecimentos perdidos

## Deteção e Correção de Erros

- Ocorrem no nível de bits
  - Erros de transmissão frequentes
    - Loops locais e enlaces sem fio
  - Erros tendem a ocorrer em rajadas
    - Vantagem: Podem danificar poucos quadros
    - Desvantagem: Dificultam a correção dos erros
- Usam informações redundantes
  - Para detectar e corrigir erros

## Deteção e Correção de Erros



EDC: bits de Deteção e Correção de Erros (redundância)

D : dados protegidos por verificação de erros, podem incluir alguns campos do cabeçalho

→ Quanto maior o campo EDC melhor será a capacidade de deteção e correção de erros

## Deteção e Correção de Erros



EDC: bits de Deteção e Correção de Erros (redundância)  
 D : dados protegidos por verificação de erros, podem incluir alguns campos do cabeçalho

→ Quanto maior o campo EDC melhor será a capacidade de deteção e correção de erros

## Deteção e Correção de Erros

- Códigos de **deteção** de erros
  - Bons para enlaces confiáveis
    - Retransmissão "mais confiável"
    - Ex.: enlaces de fibra
- Códigos de **correção** de erros
  - Bons para enlaces pouco confiáveis
    - Retransmissão pode ainda conter erros
    - Ex.: enlaces sem-fio

## Detecção e Correção de Erros

- Quadro com  $m$  bits de dados e  $r$  bits de redundância
  - Tamanho total  $n$  bits  $\rightarrow n = m + r$
- A unidade de  $n$  bits é chamada **palavra** de código de  $n$  bits
- Número de posições de bits que duas palavras diferem entre si é chamado de **distância** (de Hamming)
  - 0011 e 0000 tem distância igual a 2

## Detecção e Correção de Erros

- Se duas palavras de código estiverem a uma distância  $d$  uma da outra...
  - É necessário corrigir  $d$  erros para converter uma na outra
- Em geral, todas as  $2^m$  mensagens de dados são válidas
  - Porém, nem todas as  $2^n$  palavras de código são usadas
- Cálculo da distância de Hamming do código completo:
  - Elabora-se uma lista contendo todas as palavras válidas e localiza-se as duas cuja distância é mínima
    - Essa distância é a distância de Hamming

## Detecção e Correção de Erros

- Detecção e correção de erros **dependem da distância de Hamming** do código completo
  - Para detectar  $x$  erros  $\rightarrow$  código de distância  $d = x + 1$ 
    - Não há como  $x$  erros de bits transformarem uma palavra de código válida em outra válida
  - Para corrigir  $x$  erros  $\rightarrow$  código de distância  $d = 2x + 1$ 
    - Palavras de código válidas estarão tão distantes que, mesmo com  $d$  alterações, a palavra de código original continuará mais próxima do que qualquer outra

## Detecção e Correção de Erros

- Exemplo
  - Código contendo as seguintes palavras: 0000000000, 0000011111, 1111100000 e 1111111111
    - Distância igual a 5
  - Se  $d = 5$ , pode corrigir erros duplos ( $d = 2x + 1 \rightarrow x = 2$ )
    - Se detecta 0000000111 (é um erro duplo)
      - Original deve ser 0000011111
  - Em compensação, com  $d = 5$ , não se pode corrigir erros triplos, apenas detectá-los
    - Se detecta 0000000111 e 0000000000 foi transmitido
      - Erro não é corrigido de maneira correta...

## Verificação de Paridade

- Código simples de detecção de erros
- Bit de paridade acrescentado aos dados
  - Escolhido de forma que o número de 1's da palavra de código seja par ou ímpar
- Receptor conta quantos 1's a palavra possui
  - Se for usada paridade par e número de 1's for ímpar
    - Ocorreu um número ímpar de erros
  - Se número de erros for par
    - Erros não são detectados
- Código com um único bit de paridade tem distância 2
  - Pode detectar erros isolados

## Verificação de Paridade

- Exemplo
  - 1011010 enviado com paridade par  $\rightarrow$  10110100
  - 1011010 enviado com paridade ímpar  $\rightarrow$  10110101
- Problema: Erros ocorrem geralmente em rajada
  - Paridade com um bit não é suficiente
  - Solução:
    - Aumenta-se o número de bits de paridade

## Verificação de Paridade

- Paridade bidimensional
  - Pode detectar e corrigir erros isolados
  - Pode detectar erros duplos

Faz paridade de linha, de coluna e de bits de paridade

## Exemplo de paridade bidimensional (fonte: Kurose)



Nenhum erro					Erro de bit único corrigível				
1	0	1	0	1	1	0	1	0	1
1	1	1	1	0	1	0	1	1	0
0	1	1	0	1	0	1	1	0	1
0	0	1	0	1	0	0	1	0	1

Erro de paridade

## Código de Hamming

- Bits da palavra de código são numerados consecutivamente
  - Primeiro bit (Bit 1) na extremidade esquerda
- Bits que são potências de 2 são bits de verificação
- Outros bits são preenchidos com os  $m$  bits de dados

## Código de Hamming

- Bits de verificação calculados usando a paridade par ou ímpar dos bits que verificam
  - Exemplo com código de mais de 12 bits
    - Bit 1 → paridade dos bits 3, 5, 7, 9, 11, ...
    - Bit 2 → paridade dos bits 3, 6, 7, 10, 11, ...
    - Bit 4 → paridade dos bits 5, 6, 7, 12, ...
    - Bit 8 → paridade dos bits 9, 10, 11, 12, ...
  - Decomposição
    - 11 = 1 + 2 + 8, logo tanto o bit 1, quanto o 2 e o 8 consideram o bit 11 para cálculo da sua paridade...
    - 3 = 1 + 2, usa o bit 1 e 2
    - 9 = 1 + 8, usa o bit 1 e 8
  - Exemplo: palavra de 11 bits, 1001000 → 00110010000

## Código de Hamming

- Exemplo: Código com palavra de 11 bits
  - Mensagem de 7 bits (# bits - # potências de 2): 1001000

			1		0	0	1		0	0	0
posição	1	2	3	4	5	6	7	8	9	10	11

- Calculando a paridade
  - Bit 1 → paridade de 3, 5, 7, 9, 11
  - Bit 2 → paridade de 3, 6, 7, 10, 11
  - Bit 4 → paridade de 5, 6, 7
  - Bit 8 → paridade de 9, 10, 11

	0	0	1	1	0	0	1	0	0	0	0
posição	1	2	3	4	5	6	7	8	9	10	11

## Código de Hamming

- Receptor
  - Inicializa um contador com zero (contador = 0)
  - Examina cada bit de verificação  $k$  ( $k=1, 2, 4, 8, \dots$ ) para confirmar se a paridade está correta
    - Caso não esteja,  $k$  é somado ao valor do contador

## Código de Hamming

- Receptor
  - Contador indica zero após o exame de todos os bits de verificação (contador = 0)
    - Palavra aceita como válida
  - Se o contador não é zero (contador  $\neq$  0)
    - Ele contém o número do bit errado
      - Ex.: Se os bits de verificação 1, 2 e 8 estiverem incorretos, o bit invertido será igual a 11 (contador = 1 + 2 + 8 = 11)
      - O bit 11 é o único verificado por 1, 2 e 8 ao mesmo tempo!

## Código de Hamming

- Só pode corrigir erros simples
  - E se os erros fossem em rajada?
    - Sequência de  $k$  palavras consecutivas é organizada como uma matriz, com uma palavra de código por linha
    - Em vez de transmitir os dados uma palavra de código por vez, da esquerda para a direita, transmite-se uma coluna por vez, começando pela coluna mais à esquerda

## Código de Hamming

- Só pode corrigir erros simples
  - E se os erros fossem em rajada?
    - Receptor reconstrói a matriz, uma coluna por vez
    - Se ocorrer um erro em rajada com a extensão  $k$ , no máximo um bit de cada uma das  $k$  palavras de código será afetado
      - Bloco é restaurado, pois os erros se tornam  $k$  erros simples!

Exemplo para corrigir erros em rajada (fonte: Tanenbaum)

Caracteres	ASCII	Bits de paridade
{ k palavras	H	1001000
	a	1100001
	m	1101101
	m	1101101
	i	1101001
	n	1101110
	g	1100111
	o	0100000
	c	1100011
	o	1101111
d	1100100	
e	1100101	

Bits de paridade
00110010000
10111001001
11101010101
11101010101
01101011001
01101010110
01111001111
10011000000
11111000011
10101011111
11111001100
00111000101

Primeira coluna a ser enviada      Última coluna a ser enviada

## Soma de Verificação (Checksum)

- Método simples
  - Normalmente implementado em *software*
- Bits de dados tratados como uma sequência de números inteiros de  $k$  bits
- Soma-se esses números inteiros (em complemento a 1) e usa-se o total como bits de detecção de erros

## Soma de Verificação (Checksum)

- Receptor pode recalculer o *checksum* e compará-lo com o transmitido
  - Se diferente  $\rightarrow$  Erro!
- Usado no TCP, no UDP e no IP

## Soma de Verificação (Checksum)

- Observação
  - Ao adicionar números, o transbordo (vai um) do bit mais significativo deve ser adicionado ao resultado
- Exemplo: adição de dois inteiros de 16-bits

```

      1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
      1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
      -----
transbordo ① 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
      soma 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
soma de
verificação 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
    
```

EEL878: Redes de Computadores 1 – Del-Poli/UFRJ

Professor Miguel Campista

## Soma de Verificação (Checksum)

- Observação
  - Ao adicionar números, o transbordo (vai um) do bit mais significativo deve ser adicionado ao resultado
- Exemplo: adição de dois inteiros de 16-bits

soma + soma de verificação = 1111111111111111 →  
Correto!

```

      ① 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
      soma 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
soma de
verificação 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
    
```

EEL878: Redes de Computadores 1 – Del-Poli/UFRJ

Professor Miguel Campista

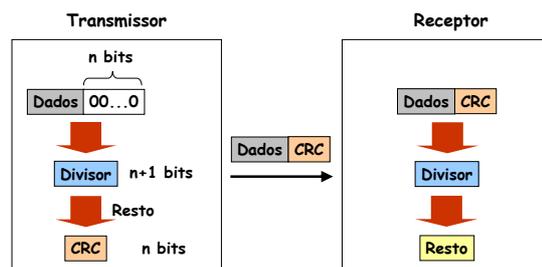
## CRC

- Código de redundância cíclica (*Cyclic Redundancy Check*) ou código polinomial
- Mais complexo
  - Geralmente implementado em hardware
- Trata uma sequência de bits como representações de polinômios com coeficientes 0 e 1

EEL878: Redes de Computadores 1 – Del-Poli/UFRJ

Professor Miguel Campista

## CRC



Se o Resto for zero, significa que os dados são aceitos

EEL878: Redes de Computadores 1 – Del-Poli/UFRJ

Professor Miguel Campista

## CRC

- Quadro de  $k$  bits  $\rightarrow k$  termos, de  $x_{k-1}$  até  $x_0$ 
  - Polinômio de grau  $k-1$
- Aritmética polinomial feita em módulo 2, sem transportes para adição nem empréstimos para subtração
  - Adição e subtração são idênticas à operação ou-exclusivo

EEL878: Redes de Computadores 1 – Del-Poli/UFRJ

Professor Miguel Campista

## CRC

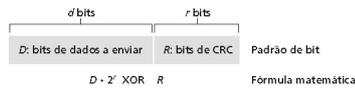
- Dados (D): Sequência de coeficientes de um polinômio (D)
  - É escolhido um polinômio Gerador (G)
    - grau(G) =  $r+1$  bits
  - Divide-se (módulo 2) o polinômio  $D \cdot 2^r$  por G
  - Acrescenta-se o resto (R) a D
  - Observa-se que, por construção, a nova sequência  $\langle D, R \rangle$ 
    - É exatamente divisível por G
  - Receptor conhece G, divide  $\langle D, R \rangle$  por G
    - Caso o resto seja diferente de zero  $\rightarrow$  Erro!

EEL878: Redes de Computadores 1 – Del-Poli/UFRJ

Professor Miguel Campista

## CRC

- Pode-se detectar erros em rajadas
  - Menores do que  $r+1$  bits
- Largamente usado na prática em redes
  - Ex.: ATM e HDLC



## CRC

Queremos:

$$D \cdot 2^r \text{ XOR } R = nG$$

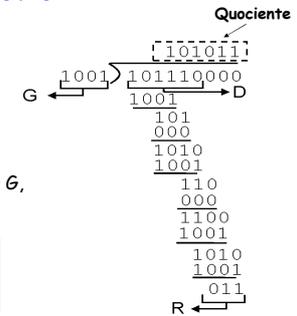
De forma equivalente:

$$D \cdot 2^r = nG \text{ XOR } R$$

de forma equivalente:

se dividirmos  $D \cdot 2^r$  por  $G$ ,  
queremos o resto  $R$

$$R = \text{resto} \left( \frac{D \cdot 2^r}{G} \right)$$



## CRC

- Transmissor e receptor devem concordar em relação ao uso de um polinômio gerador  $G(x)$ 
  - Tanto o bit de mais alta ordem quanto o bit de mais baixa ordem devem ser iguais a 1
  - Polinômio gerador pode ser escolhido de acordo com a probabilidade de ocorrerem erros
- Quadro de  $m$  bits corresponde a  $M(x)$
- $M(x)$  tem de ser de maior grau do que  $G(x)$

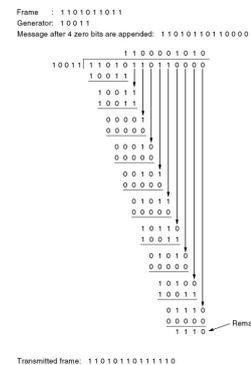
## CRC

- CRC acrescentado ao final do quadro de forma que o quadro verificado seja divisível por  $G(x)$ 
  - Sequência de verificação de quadro (*Frame Check Sequence - FCS*)
- Ao receber o quadro verificado, o receptor tentará dividi-lo por  $G(x)$ 
  - Se o resto é diferente de zero → Erro!

## CRC

- Algoritmo
  - Seja  $r$  o grau de  $G(x)$
  - Acrescente  $r$  bits à extremidade de mais baixa ordem de  $M(x)$ 
    - Polinômio  $x^r M(x)$
  - Divida a sequência de bits correspondente a  $x^r M(x)$  pela sequência correspondente por  $G(x)$
  - Subtraia o resto da sequência correspondente a  $x^r M(x)$
  - Resultado é o quadro verificado que deverá ser transmitido
    - Polinômio  $T(x)$

### Exemplo de cálculo de CRC (fonte: Tanenbaum)



## CRC

- Caso haja erros
  - Recebido  $T(x) + E(x)$ 
    - Cada bit 1 de  $E(x)$  corresponde a um bit que foi invertido
    - Se houver  $k$  bits 1 em  $E(x)$  significa que ocorreram  $k$  erros de bits simples
    - Único erro em rajada é caracterizado por um bit 1 inicial, uma mistura de bits 0 e 1 e um bit 1 final
  - Receptor divide o quadro com verificação por  $G(x)$ 
    - $E(x)/G(x)$ 
      - Erros que correspondem a polinômios contendo  $G(x)$  como fator são ignorados
      - Todos os outros são detectados

## CRC

- Caso haja erros
  - Se houver um único erro de bit
    - $E(x) = x^i$ ,  $i$  determina o bit incorreto
  - Se contiver dois ou mais termos,  $G(x)$  nunca dividirá  $E(x) \rightarrow$  todos os erros de um único bit serão detectados
  - Se houver dois erros isolados de um único bit
    - $E(x) = x^i + x^j$ ,  $i > j$
    - $E(x) = x^i(x^{i-j} + 1)$
    - Se  $G(x)$  não é divisível por  $x$ , para detectar erros duplos  $G(x)$  não deve dividir  $x^k + 1$  para qualquer  $k$  até o valor máximo de  $i - j$
    - Ex.:  $x^{15} + x^{14} + 1$  não dividirá  $x^k + 1$  para  $k$  menor que 32768

## CRC

- Caso haja erros
  - Se houver um número ímpar de bits com erro
    - $E(x)$  conterá um número ímpar de termos
    - Nenhum polinômio com um número ímpar de bits com erros terá  $x + 1$  como fator
    - Ao tornar  $x + 1$  um fator de  $G(x)$ , pode-se detectar todos os erros que consistem em um número ímpar de bits invertidos

## CRC

- Caso haja erros
  - Se houver erros em rajada de tamanho  $k \leq r$ 
    - $E(x) = x^i(x^{k-1} + \dots + 1)$ ,  $i$  determina a distância entre a extremidade direita da rajada e a extremidade esquerda do quadro recebido
    - Se contiver um termo  $x^0$ ,  $G(x)$  não terá  $x^i$  como fator, logo se o grau de  $x^{k-1} + \dots + 1$  for menor que o grau de  $G(x)$ , o resto nunca poderá ser zero
    - Pode-se detectar todos os erros em rajada de tamanho  $\leq r$
  - Outras análises sem garantia de detecção dos erros

## CRC

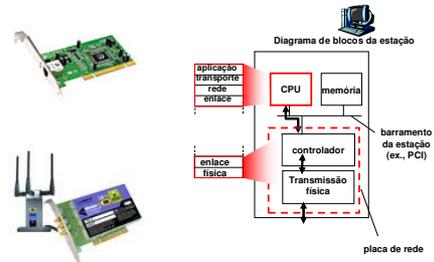
- Usado em diversos padrões de redes locais e metropolitanas
- Exemplo de  $G(x)$  do IEEE 802
  - $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$

## Implementação

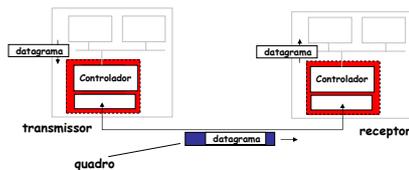
## Implementação

- A camada de enlace é implementada por cada um dos nós da rede
  - Cada um pode implementar uma tecnologia
- É implementada no "adaptador" (*Network Interface Card* - NIC)
  - Exs: placa Ethernet, cartão PCMCIA, cartão 802.11
  - Também implementa a camada física
  - Está conectado ao barramento de sistema do nó
    - Ou integrada na placa mãe
  - É uma combinação de hardware, software e firmware

## Implementação



## Comunicação entre Adaptadores



- Lado transmissor
  - Encapsula o datagrama em um quadro
  - Adiciona bits de verificação de erro, transferência confiável de dados, controle de fluxo, etc.
- Lado receptor
  - Verifica erros, transporte confiável, controle de fluxo, etc.
  - Extrai o datagrama, passa-o para o nó receptor

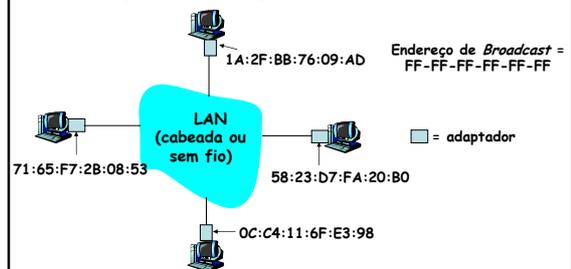
## Endereçamento

## Endereços MAC

- Endereço IP de 32 bits
  - Endereços da camada de rede
  - Usado para levar o datagrama à sub-rede IP destino
- Endereço MAC (ou LAN, ou físico, ou Ethernet)
  - Leva o datagrama de uma interface até outra interface conectada fisicamente (na mesma rede)
  - Possui 48 bits (para a maioria das redes)
    - Representados por 12 dígitos hexadecimais agrupados 2 a 2 (Ex.: 1A:2F:BB:76:09:AD)
  - Gravado na ROM do adaptador ou configurado por software

## Endereços MAC

Cada adaptador na LAN possui um endereço MAC único



## Endereços MAC

- Alocação de endereços MAC gerenciada pelo IEEE
- Um fabricante compra uma parte do espaço de endereços
  - Garantia de unicidade
- Analogia:
  - Endereço MAC
    - Como número do CPF
  - Endereço IP
    - Como endereço postal (CEP)

## Endereços MAC

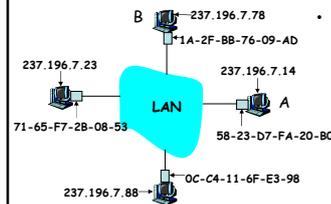
- Endereço MAC tem estrutura linear
  - Portabilidade
    - É possível mover um cartão LAN de uma LAN para outra
- Endereço IP hierárquico NÃO é portátil
  - Requer IP móvel, por exemplo
  - Depende da sub-rede IP à qual o nó está conectado

## ARP

- Protocolo de resolução de endereços (*Address Resolution Protocol*)
  - Descrito na RFC 826
- Faz a tradução de endereços IP para endereços MAC da maioria das redes IEEE 802
  - Executado dentro da sub-rede
- Cada nó (estação ou roteador) possui uma tabela ARP
  - Contém endereço IP, endereço MAC e TTL
  - Tabela ARP construída automaticamente

## ARP

Como obter o endereço MAC de B a partir do endereço IP de B?



- Cada nó de uma LAN possui uma tabela ARP
- Tabela ARP: mapeamento de endereços IP/MAC para alguns nós da LAN <endereço IP; endereço MAC; TTL>
  - TTL (*Time To Live*): tempo a partir do qual o mapeamento de endereços será esquecido (valor típico de 20 min)

## Funcionamento do ARP na Mesma Rede

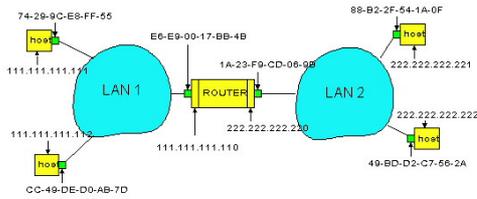
- **A** deseja enviar datagrama para **B**, mas o endereço MAC de **B** não está na tabela ARP...
- Para descobrir o endereço MAC de **B**, **A** difunde um pacote de solicitação ARP com o endereço IP de **B**
  - Endereço MAC destino = FF-FF-FF-FF-FF-FF
  - Todas as máquinas na LAN recebem a consulta do ARP
- **B** então recebe o pacote ARP com a solicitação e responde a **A** com o seu endereço MAC
  - Quadro de resposta é enviado para o endereço MAC (*unicast*) de **A**

## Funcionamento do ARP na Mesma Rede

- Um *cache* (salva) o par de endereços IP-para-MAC na sua tabela ARP até que a informação expire
  - É "*soft state*"
    - Informação que expira a menos que seja renovada
  - Um nó pode responder a uma requisição com um endereço MAC que conheça
    - Não necessariamente o próprio nó de destino
- ARP é "*plug-and-play*"
  - Os nós criam suas tabelas ARP sem a intervenção do administrador da rede

## Funcionamento entre Redes Diferentes

- Envio de datagrama de **A** para **B** através de **R**
- O Roteador **R** possui duas tabelas ARP
  - Uma para cada rede local



EEL878: Redes de Computadores 1 – Del-Poli/UFRJ Professor Miguel Campista

## Funcionamento entre Redes Diferentes

- **A** cria o datagrama com endereço IP de fonte **A** e de destino **B**
- **A** consulta a tabela de roteamento e obtém **R** como próximo salto
- **A** usa o ARP para obter o endereço MAC de **R**
- **A** cria um quadro com endereço MAC de destino **R** e o datagrama de **A** para **B** na carga útil
- Adaptador de **A** envia o quadro para **R**
- Adaptador de **R** recebe o quadro

EEL878: Redes de Computadores 1 – Del-Poli/UFRJ

Professor Miguel Campista

## Funcionamento entre Redes Diferentes

- **R** remove o datagrama IP do quadro Ethernet e verifica que é destinado a **B**
- **R** consulta a tabela de roteamento
- **R** usa o ARP para obter o endereço MAC de **B**
- **R** cria o quadro contendo o datagrama de **A** para **B**
- Adaptador de **R** envia o quadro para **B**
- Adaptador de **B** recebe o quadro

EEL878: Redes de Computadores 1 – Del-Poli/UFRJ

Professor Miguel Campista

## Ferramentas ARP

- Saída do `tcpdump`

```
[root@masq-gw]# tcpdump -i eth0 \(( arp \)
tcpdump: listening on eth0
0:80:c8:f8:4a:51 ff:ff:ff:ff:ff:ff 42: arp who-has 192.168.99.254 tell 192.168.99.35
0:80:c8:f8:5c:73 0:80:c8:f8:4a:51 60: arp reply 192.168.99.254 is-at 0:80:c8:f8:5c:73
```

EEL878: Redes de Computadores 1 – Del-Poli/UFRJ

Professor Miguel Campista

## Ferramentas ARP

- Ferramenta `arp`
  - Mostra a tabela ARP de uma estação

```
[miguel@tijuca ~]#arp
Address      HWtype  HWaddress      Flags Mask  Iface
sono.gta.ufrj.br  ether   38:60:77:72:84:e8  C          br0
niteroi.gta.ufrj.br  ether   4c:72:b9:b0:df:19  C          br0
ramos.gta.ufrj.br   ether   00:1c:c0:91:36:63  C          br0
inga.gta.ufrj.br    ether   00:1c:c0:1c:b0:31  C          br0
grajau.gta.ufrj.br  ether   70:71:bc:e2:ee:7e  C          br0
```

EEL878: Redes de Computadores 1 – Del-Poli/UFRJ

Professor Miguel Campista

## Ferramentas ARP

- Arping
  - "Ping" da camada de enlace

```
ARPING 192.168.0.1 from 192.168.0.10 eth0
Unicast reply from 192.168.0.1 [00:01:80:38:F7:4C] 0.510ms
Unicast reply from 192.168.0.1 [00:01:80:38:F7:4C] 0.601ms
Unicast reply from 192.168.0.1 [00:01:80:38:F7:4C] 0.610ms
Unicast reply from 192.168.0.1 [00:01:80:38:F7:4C] 0.605ms
Sent 4 probes (1 broadcast(s))
Received 4 response(s)
```

EEL878: Redes de Computadores 1 – Del-Poli/UFRJ

Professor Miguel Campista

## Elementos de Interconexão

## Repetidor

- **Nível físico**
- Tem um número pequeno de interfaces
- **Tarefas executadas:**
  - Recebe
  - Conformar (recupera a forma do sinal original)
  - Amplifica
  - Retransmite os bits de uma interface para todas as outras

## Hub

- **Nível físico**
- É um repetidor
- **Repete** os bits de uma porta para **todas** as outras
- Segmentos da rede formam um **único domínio de colisão**
  - Domínio de colisão é uma única rede com CSMA/CD na qual haverá colisão se duas estações da rede transmitirem ao mesmo tempo

## Hub

- **Geralmente não pode conectar segmentos da rede operando em diferentes taxas**
  - Esse caso poderia ser implementado usando dois hubs que operam em velocidades diferentes conectados internamente por um comutador de duas portas

## Ponte (bridge)

- **Nível de enlace**
- Tem um pequeno número de interfaces
- Usam o endereço MAC de destino para encaminhar e filtrar quadros
- Cada **segmento** de rede é um **domínio de colisão separado**
- Pode conectar segmentos diferentes de rede, mesmo que cada uma opere a uma taxa diferente

## Ponte (bridge)

- **Conceitos**
  - Filtragem
    - Capacidade da ponte decidir se um quadro será repassado para alguma interface ou descartado
  - Repasse
    - Capacidade de determinar as interfaces para as quais um quadro deve ser repassado e fazê-lo

## Ponte (*bridge*)

- Tabela de comutação usada no repasse
  - Se o endereço de destino está na tabela e a interface não é a mesma de onde veio, transmite para a interface correspondente
  - Se o endereço de destino está na tabela e a interface é a mesma de onde veio, descarta
    - Significa que o destino está na mesma rede da origem
  - Se o endereço de destino não está na tabela, transmite em todas as interfaces exceto a interface de onde veio

## Ponte (*bridge*)

- Possui a característica de aprendizagem automática
  - Construção automática da tabela de comutação
    - Cada quadro que passa pela ponte é examinado e são colocados na tabela o endereço fonte, a interface de onde veio o quadro e o tempo do registro na tabela
    - Registros expiram
      - Ex.: Pode-se trocar uma estação de lugar

## Comutador (*switch*)

- Nível enlace
- Pontes de alto desempenho e múltiplas interfaces
  - Tem um maior número de interfaces que os repetidores e as pontes
- Atualmente a maioria é utilizada para acesso dedicado
  - Uma única estação por domínio de colisão

## Comutador (*switch*)

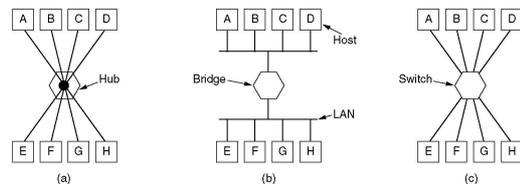
- Usam o endereço MAC de destino para encaminhar e filtrar quadros
- Cada segmento de rede é um domínio de colisão separado
- Pode conectar segmentos diferentes de rede, mesmo que cada uma opere a uma taxa diferente

## Comutador (*switch*)

- Pode trabalhar em *full-duplex*
- Comutação de quadros
  - Quadro sempre é expedido pela mesma saída, decidida uma vez por todas quando da aceitação de trocar dados

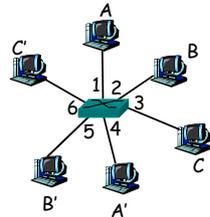
## Elementos de Interconexão

Hub, ponte e comutador (fonte: Tanenbaum)



## Comutador: Múltiplas Transmissões Simultâneas

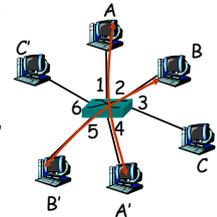
- Estações têm conexão direta e dedicada para o comutador
- Os comutadores armazenam quadros



Comutador com seis interfaces (1, 2, 3, 4, 5, 6)

## Comutador: Múltiplas Transmissões Simultâneas

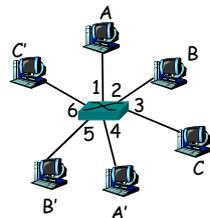
- O protocolo Ethernet é usado em cada enlace de entrada, mas não há colisões: *full duplex*
  - Cada enlace é o seu próprio domínio de colisão
- Comutação: A-para-A' e B-para-B' simultaneamente, sem colisões
  - Isto não é possível com hubs



Comutador com seis interfaces (1, 2, 3, 4, 5, 6)

## Comutador: Tabela de Comutação

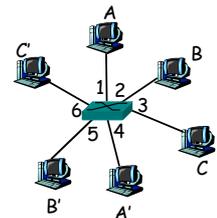
- Como é que o comutador sabe que A' é alcançável através da interface 4, e que B' é alcançável a partir da interface 5?
  - Cada comutador possui uma **tabela de comutação**, cada entrada contém:
    - endereço MAC da estação, interface para alcançar a estação, carimbo de tempo
  - Similar a uma tabela de roteamento



Comutador com seis interfaces (1, 2, 3, 4, 5, 6)

## Comutador: Tabela de Comutação

- Como são criadas e mantidas as entradas na tabela de comutação?
  - Há algo como um protocolo de roteamento?



Comutador com seis interfaces (1, 2, 3, 4, 5, 6)

## Comutador: autoaprendizagem

- Comutador **aprende** quais estações podem ser alcançados através de quais interfaces
  - Quando um quadro é recebido, o comutador "aprende" a localização do transmissor: segmento LAN de entrada
  - registra o par transmissor/localização na tabela de comutação

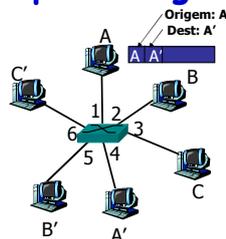


Tabela de comutação (inicialmente vazia)

end MAC	interface	TTL
A	1	60

## Filtragem e Encaminhamento

- Quando um comutador recebe um quadro:**
- registra o enlace associado com a estação transmissora
  - indexa a tabela de comutação usando o endereço MAC do destino
  - if entrada encontrada para o destino
    - then{
      - if dest estiver no segmento de onde veio o quadro
        - then descarta o quadro
        - else repassa o quadro na interface indicada
    - }
      - else usa inundação ← Repassa o quadro para todas as demais interfaces exceto aquela em que o quadro foi recebido

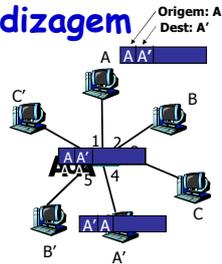
## Autoaprendizagem

- Destino do quadro desconhecido:

**inundação**

- Local do destino A conhecido:

**transmissão seletiva**

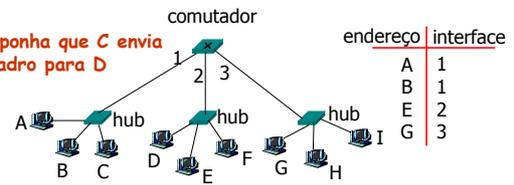


end MAC	interface	TTL
A	1	60
A'	4	60

Tabela de comutação (inicialmente vazia)

## Exemplo com Computador

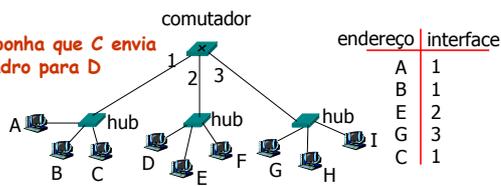
Suponha que C envia quadro para D



- Computador recebe o quadro vindo de C
  - Anota na tabela de comutação que C está na interface 1
  - Dado que D não se encontra na tabela, encaminha o quadro para as demais interfaces: 2 e 3
- Quadro é recebido por D

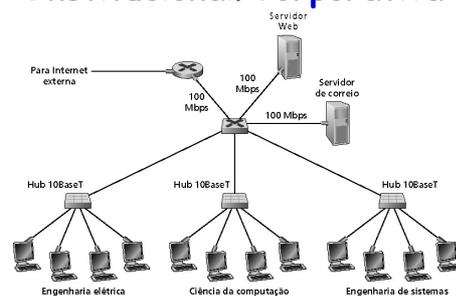
## Exemplo com Computador

Suponha que C envia quadro para D



- Computador recebe o quadro vindo de D
  - Anota na tabela de comutação que D está na interface 2
- Dado que C está na tabela, encaminha o quadro apenas na interface 1
  - Quadro é recebido por C

## Rede Institucional/Corporativa



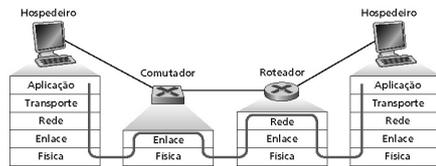
## Roteador

- Nível de rede
- Roteamento de pacote
  - Endereço do destinatário
  - Escolha da melhor saída no momento da decisão

## Computadores x Roteadores

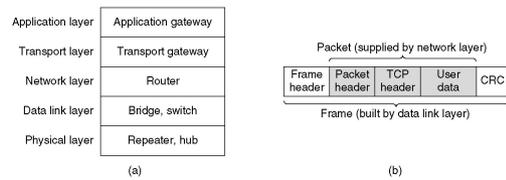
- Ambos são dispositivos do tipo armazena-e-repassa
  - Roteadores → dispositivos da camada de rede
    - Examinam os cabeçalhos dessa camada
  - Computadores → dispositivos da camada de enlace
- Roteadores
  - Mantêm tabelas de roteamento, implementam algoritmos de roteamento
- Computadores
  - Mantêm tabelas de comutação, implementam filtragem, algoritmos de aprendizagem

## Comutadores x Roteadores



## Elementos de Interconexão

Elementos (fonte: Tanenbaum)



## Material Utilizado

- Notas de aula do Prof. Igor Monteiro Moraes, disponíveis em <http://www2.ic.uff.br/~igor/cursos/redespq>

## Leitura Recomendada

- Capítulo 3 do Livro "Computer Networks", Andrew S. Tanenbaum e David J. Wetherall, 5a. Edição, Editora Pearson, 2011
- Capítulo 5 do Livro "Computer Networking: A Top Down Approach", 5a. Ed., Jim Kurose and Keith Ross, Editora Pearson, 2010