CAPÍTULO

5 CAMADA DE ENLACE DE DADOS: REDES COM FIOS

Nos Capítulos 2 a 4, discutimos as camadas de aplicação, transporte e rede. Neste capítulo e no seguinte, discutiremos a camada de enlace de dados. A pilha de protocolos TCP/IP não especifica qualquer protocolo nas camadas de enlace de dados e físicas; estas são os territórios das redes que, quando conectadas, formam a Internet. Conforme discutimos no Capítulo 1, essas redes, com ou sem fios, fornecem serviços para as três camadas superiores da pilha de protocolos TCP/IP. É um primeiro indício de que existem vários protocolos padronizados no mercado atualmente. Por isso, discutimos a camada de enlace de dados em dois capítulos: neste, discutimos os conceitos gerais envolvidos dando foco às redes com fios, também conhecidas como redes cabeadas ou guiadas; no próximo, discutiremos as redes sem fios. Este capítulo está dividido em sete seções:

- Na primeira seção, introduzimos o conceito de nós e enlaces, discutimos os tipos de enlaces e mostramos como a camada de enlace de dados é, na realidade, dividida em duas subcamadas: Controle de Enlace de Dados (DLC – Data Link Control) e Controle de Acesso ao Meio (MAC – Media Access Control).
- Na segunda seção, discutimos a subcamada de controle de enlace de dados (DLC) da camada de enlace de dados e explicamos os serviços fornecidos pela subcamada, como enquadramento, controle de fluxo e de erros, e detecção de erros.
- Na terceira seção, discutimos a subcamada de controle de acesso ao meio (MAC) da camada de enlace de dados. Explicamos diferentes abordagens usadas por ela, como acesso aleatório, acesso controlado e canalização.
- Na quarta seção, discutimos o endereçamento na camada de enlace e como o endereço da camada de enlace de um nó pode ser encontrado usando o Protocolo de Resolução de Endereços (ARP – Address Resolution Protocol).
- Na quinta seção, apresentamos as LANs com fios e em particular a Ethernet, o protocolo de LAN dominante atualmente. Navegamos pelas diferentes gerações da Ethernet e mostramos como ela evoluiu.
- Na sexta seção, discutimos outras redes com fios encontradas na Internet atual, como as redes ponto a ponto e comutadas.
- Na sétima seção, discutimos os dispositivos de conexão utilizados nas três camadas mais baixas da pilha de protocolos TCP/IP, como *hubs*, *switches* da camada de enlace e roteadores.

5.1 INTRODUÇÃO

No Capítulo 4, aprendemos que a comunicação na camada de rede é *host* a *host*. Embora possa ser fragmentado e remontado, um datagrama é uma unidade de dados enviada de um *host*, localizado

em algum lugar do mundo, para outro *host*, em algum outro lugar no mundo. A Internet, porém, é uma combinação de redes ligadas entre si por meio de dispositivos de conexão (roteadores ou *switches*). Se um datagrama precisar viajar de uma estação para outra, precisará passar por tais redes.

A Figura 5.1 mostra a comunicação entre Alice e Bob, usando o mesmo cenário que usamos nos últimos três capítulos. A comunicação na camada de enlace de dados, entretanto, é composta por cinco conexões lógicas distintas entre as camadas de enlace de dados no caminho de uma estação à outra.

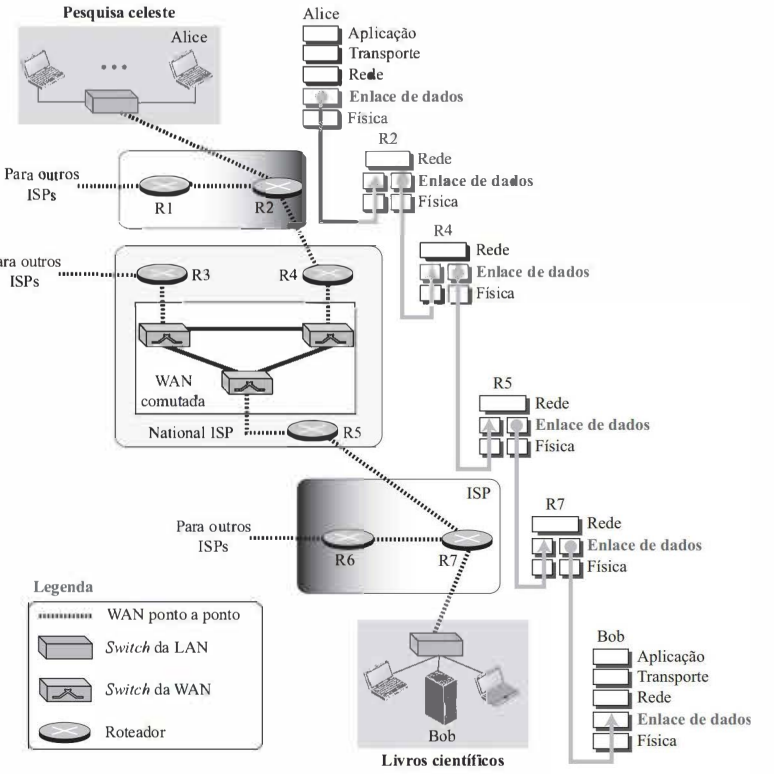


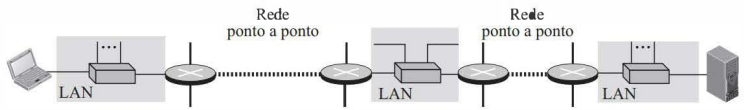
Figura 5.1 Comunicação na camada de enlace de dados.

A camada de enlace de dados do computador de Alice se comunica com a camada de enlace de dados do roteador R2. A camada de enlace de dados do roteador R2 se comunica com a camada de enlace de dados do roteador R4, e assim por diante. Finalmente, a camada de enlace de dados do roteador R7 se comunica com a camada de enlace de dados do computador de Bob. Apenas uma camada de enlace de dados atua tanto na origem como no destino, mas duas camadas de enlace de

dados atuam em cada roteador. A razão é que os computadores de Alice e Bob estão conectados a uma única rede; cada roteador, porém, recebe os dados de entrada vindos de uma rede e envia os dados de saída para outra rede.

5.1.1 Nós e enlaces

Embora a comunicação nas camadas de aplicação, transporte e rede sejam fim a fim, a comunicação na camada de enlace de dados é nó a nó, ou salto a salto. Como vimos nos capítulos anteriores, uma unidade de dados de um ponto na Internet precisa passar por muitas redes (LANs e WANs) para chegar a outro ponto. Essas LANs e WANs são conectadas por roteadores. É comum denominar as duas estações finais e os roteadores como **nós** e as redes entre eles como **enlaces**. Na Figura 5.2, mostramos uma representação simples de enlaces e nós quando o caminho da unidade de dados consiste em apenas seis nós.



(*a*) Uma pequena parte da Internet

(*b*) Nós e enlaces

Figura 5.2 Nós e enlaces.

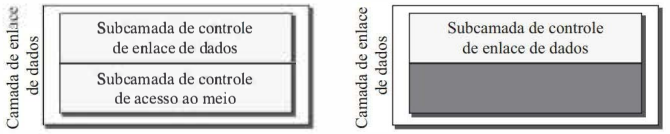
O primeiro nó é a estação de origem; o último é a estação de destino. Os outros quatro nós são quatro roteadores. O primeiro, o terceiro e o quinto enlaces representam três LANs; o segundo e o quarto enlaces representam as duas WANs.

5.1.2 Dois tipos de enlaces

Apesar de dois nós estarem fisicamente conectados por um meio de transmissão como um cabo ou o ar, precisamos lembrar que a camada de enlace de dados controla a forma como o meio é usado. Podemos ter uma camada de enlace de dados que utiliza toda a capacidade do meio; e também uma que utiliza apenas uma parte da capacidade do enlace. Em outras palavras, podemos ter um *enlace ponto a ponto* ou um *enlace de broadcast*. Em um enlace ponto a ponto, o enlace é dedicado aos dois dispositivos; já em um enlace de *broadcast*, o enlace é compartilhado entre vários pares de dispositivos. Por exemplo, quando dois amigos usam os tradicionais telefones fixos para conversar, eles estão usando um enlace ponto a ponto; quando usam seus celulares, eles estão usando um enlace de *broadcast* (o ar é compartilhado por diversos usuários de telefones celulares).

5.1.3 Duas subcamadas

Para entender melhor a funcionalidade da camada de enlace e os serviços prestados por ela, podemos dividir a camada de enlace de dados em duas subcamadas: *Controle de Dados de Enlace* (DLC – Data Link Control) e *Controle de Acesso ao Meio* (MAC – Media Access Control). Isto não é incomum porque, como veremos neste e no próximo capítulo, atualmente os protocolos de LAN usam a mesma estratégia. A subcamada de controle de enlace de dados lida com todas as questões comuns tanto aos enlaces ponto a ponto como aos de *broadcast*; a subcamada de controle de acesso ao meio lida apenas com questões específicas dos enlaces de *broadcast*. Ou seja, separamos esses dois tipos de enlaces da camada de enlace de dados, conforme mostra a Figura 5.3.



(*a*) Camada de enlace de dados de um enlace de *broadcast*

(*b*) Camada de enlace de dados de um enlace ponto a ponto

Figura 5.3 Dividindo a camada de enlace de dados em duas subcamadas.

Neste capítulo, começaremos discutindo a subcamada de controle de enlace de dados, que é comum a ambos os tipos de enlaces. Em seguida, discutiremos a subcamada de controle de acesso ao meio, usada somente nos enlaces de *broadcast*. Após a discussão sobre essas duas subcamadas, discutiremos um protocolo pertencente a cada categoria.

5.2 CONTROLE DE ENLACE DE DADOS

O controle de enlace de dados lida com procedimentos para a comunicação entre dois nós adjacentes – comunicação nó a nó – não importando se o enlace é dedicado ou de *broadcast*. As funções do **Controle de Enlace de Dados** (DLC – Link Data Control) incluem *enquadramento*, *controle de fluxo e de erro*, e detecção e correção de erros. Nesta seção, primeiramente discutimos o enquadramento, ou como organizar os *bits* que são transportados pela camada física. Em seguida, o controle de fluxo e de erro. Técnicas para a detecção de erros são abordadas no fim desta seção.

5.2.1 Enquadramento

A transmissão de dados na camada física consiste em mover *bits* na forma de um sinal, indo da origem até o destino. A camada física fornece mecanismos de sincronização de *bits* para garantir que o emissor e o receptor usem as mesmas durações e temporização de *bits*. Discutiremos a camada física no Capítulo 7.

A camada de enlace de dados, por outro lado, precisa agrupar os *bits* em quadros, de modo que cada quadro seja distinguível um do outro. Nosso sistema postal aplica um tipo de enquadramento. O simples ato de inserir uma carta em um envelope separa uma parte da informação da outra; o envelope serve como delimitador. Além disso, cada envelope define os endereços do remetente e do destinatário, algo necessário porque o transporte realizado pelo sistema postal é do tipo muitos para muitos.

O *enquadramento* na camada de enlace de dados define que uma mensagem vai de uma origem para um destino por meio da adição de um endereço de remetente e de um de destinatário. Este determina para onde o pacote deve ir; o endereço do remetente ajuda o destinatário a confirmar a recepção dos dados.

Embora a mensagem completa possa ser acondicionada em um quadro, isto normalmente não é feito. Uma razão é que tal quadro poderia ficar muito grande, tornando os mecanismos de controle de fluxo e de erros muito ineficientes. Quando uma mensagem é transportada em um quadro muito grande, mesmo o erro em um único *bit* exigiria a retransmissão do quadro todo. Quando uma mensagem é dividida em quadros menores, um erro em um único *bit* afeta apenas o quadro pequeno.

Comprimento dos quadros

Os quadros podem ter um tamanho fixo ou variável. No *enquadramento de tamanho fixo*, não é necessário definir os limites dos quadros: o próprio tamanho pode ser usado como um delimitador. Um exemplo de rede que adota esse tipo de enquadramento são as WANs ATM, que utilizam quadros de tamanho fixo denominados *células*. Discutiremos o ATM no final deste capítulo.

O cerne da nossa discussão neste capítulo refere-se ao *enquadramento de tamanho variável*, que é, de modo geral, usado em LANs. Nele, precisamos de algum mecanismo para definir o final de um quadro e o início do próximo. Historicamente, duas abordagens foram criadas com esse propósito: uma abordagem orientada a caracteres e outra orientada a *bits*.

Enquadramento orientado a caracteres

No *enquadramento orientado a caracteres* (ou *orientado a bytes*), os dados transportados são caracteres de 8 *bits* provenientes de algum sistema de codificação, tal como ASCII (ver o Apêndice A no *site* do livro). O cabeçalho, que normalmente transporta os endereços de origem e de destino, bem como outras informações de controle, como o rodapé (*trailer*), que transporta *bits* redundantes usados para detecção de erros, também são múltiplos de 8 *bits*. Para separar um quadro do outro, um **marcador** de 8 *bits* (1 *byte*) é adicionado ao início e ao fim de cada quadro. O marcador, composto por caracteres especiais dependentes do protocolo, sinaliza o início e o fim de um quadro. A Figura 5.4 mostra o formato de um quadro em um protocolo orientado a caracteres.

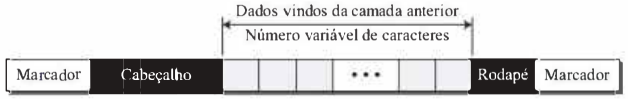


Figura 5.4 Um quadro em um protocolo orientado a caracteres.

O enquadramento orientado a caracteres era bastante popular quando as camadas de enlace de dados trocavam somente texto entre elas. O marcador podia ser qualquer caractere que não fosse usado na comunicação usando texto. Atualmente, entretanto, enviamos outros tipos de informação, como gráficos, áudio e vídeo, de modo que qualquer sequência usada como marcador pode também ser parte da informação. Se isto acontecer, o receptor, ao se deparar com essa sequência no meio dos dados, pensa que atingiu o fim do quadro. Para corrigir esse problema, uma estratégia de **preenchimento de *byte*** foi adicionada ao enquadramento orientado a caracteres. Na técnica de preenchimento de *byte* (ou preenchimento de caracteres), um *byte* especial é adicionado à seção de dados do quadro sempre que ele apresentar um caractere com a mesma sequência de *bits* que o marcador. A seção de dados é preenchida com um *byte* adicional, normalmente denominado *caractere de escape* (ESC) e consiste em uma sequência de *bits* predefinida. Sempre que o receptor encontra o caractere ESC, ele o remove da seção de dados e trata o próximo caractere como dados e não como um marcador de fim de quadro.

O preenchimento de *byte* por meio de um caractere de escape permite a presença de marcadores na seção de dados do quadro, mas cria outro problema. O que acontece se o texto contiver um ou mais caracteres de escape seguidos por um *byte* idêntico a um marcador? O receptor remove o

caractere de escape, mas mantém o *byte* seguinte, que é incorretamente interpretado como o fim do quadro. Para resolver esse problema, os caracteres de escape que fazem parte do texto também devem ser identificados com outro caractere de escape. Em outras palavras, se o caractere de escape fizer parte do texto, um caractere extra é adicionado para indicar que o segundo deles faz parte do texto. A Figura 5.5 ilustra essa situação.

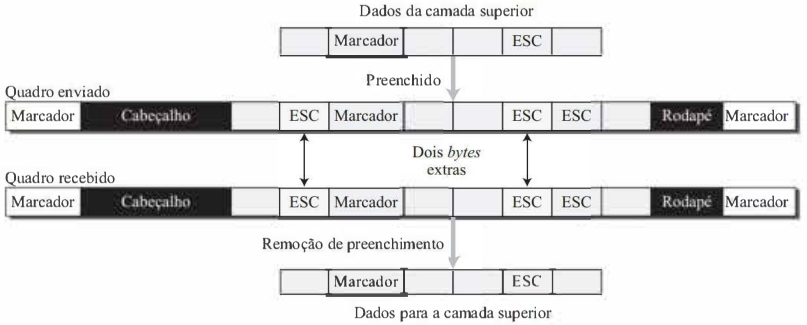


Figura 5.5 Preenchimento e remoção do preenchimento de *byte*.

Preenchimento de *byte* refere-se ao processo de adicionar um *byte* extra sempre que houver um marcador ou um caractere de escape no texto.

Protocolos orientados a caracteres apresentam outro problema no que se refere à comunicação de dados. Os sistemas universais de codificação em uso atualmente, como o Unicode, usam caracteres de 16 ou 32 *bits* que conflitam com caracteres de 8 *bits*. Podemos dizer que, em geral, a tendência tem se tornado a utilização de protocolos orientados a *bits* que discutiremos a seguir.

Enquadramento orientado a *bits*

No *enquadramento orientado a bits*, a seção de dados de um quadro consiste em uma sequência de *bits* que pode ser interpretada pela camada superior como texto, gráficos, vídeo, áudio e assim por diante. No entanto, além de cabeçalhos (e possíveis rodapés), ainda precisamos de um delimitador para separar um quadro de outro. A maioria dos protocolos usa um marcador especial de 8 *bits*, 01111110, como o delimitador para determinar o início e o fim do quadro, conforme mostra a Figura 5.6.

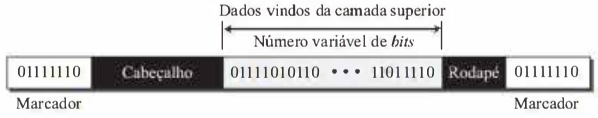


Figura 5.6 Um quadro em um protocolo orientado a *bits*.

Esse marcador pode criar o mesmo tipo de problema que vimos nos protocolos orientados a caracteres. Isto é, se o marcador aparecer nos dados, precisamos ser capazes de informar o receptor de que este não é o fim do quadro. Fazemos isso por meio da adição de um único *bit* (em vez de 1 *byte*) para evitar que essa sequência de *bits* seja confundida com um marcador. A estratégia é denominada **preenchimento de *bit***. No preenchimento de *bit*, caso seja encontrado um *bit* 0 seguido de cinco *bits* 1 consecutivos, um *bit* 0 extra é adicionado, que acaba sendo removido da seção de dados pelo receptor. Observe que o *bit* extra é adicionado após o aparecimento de um 0 seguido por cinco 1s, independentemente do valor do *bit* seguinte. Isto garante que a sequência do marcador não aparecerá inadvertidamente no quadro.

O preenchimento de *bits* refere-se ao processo de adicionar um 0 extra sempre que os dados contiverem cinco 1s consecutivos após um 0, de modo que o receptor não confunda a sequência 0111110 com um marcador.

A Figura 5.7 mostra os processos de preenchimento de *bit* no remetente e de remoção do *bit* no receptor. Perceba que, mesmo se tivermos um 0 após cinco 1s, ainda adicionamos um 0, removido pelo receptor.

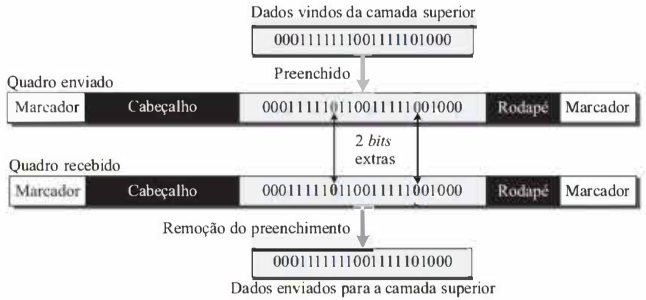


Figura 5.7 Preenchimento e remoção do preenchimento de *bit*.

Isto significa que, se uma sequência semelhante a um marcador 01111110 aparecer nos dados, ela será modificada para 011111010 (preenchida) e não será confundida com um marcador pelo receptor. O marcador de fato, 01111110, não é preenchido pelo remetente, mas é reconhecido pelo receptor.

5.2.2 Controle de fluxo e erros

Definimos o controle de fluxo e de erros no Capítulo 3. Uma das responsabilidades da subcamada de controle de enlace de dados é o controle de fluxo e erros na camada de enlace de dados.

Controle de fluxo

Conforme discutimos na camada de transporte (Capítulo 3), o controle de fluxo coordena a quantidade de dados que pode ser enviada antes que uma confirmação seja recebida. Na camada de enlace de dados, o controle de fluxo é uma das tarefas da subcamada de controle de enlace de dados. O

conceito de controle de fluxo na camada de enlace de dados segue o mesmo princípio que discutimos para a camada de transporte. Embora o controle de fluxo na camada de transporte seja fim a fim (*host* a *host*), o controle de fluxo na camada de enlace de dados é nó a nó, ao longo do enlace.

Controle de erros

O controle de erros consiste tanto na detecção como na correção de erros; ele permite que o receptor informe o remetente de quaisquer quadros perdidos ou danificados durante a transmissão e também que coordene a retransmissão dos quadros pelo remetente. Na camada de enlace de dados, o termo **controle de erros** refere-se principalmente aos métodos de detecção de erros e de retransmissão. O controle de erros na camada de enlace de dados é geralmente implementado de forma simples: toda vez que for detectado um erro na transmissão, os quadros corrompidos são retransmitidos. Precisamos, entretanto, ressaltar que o controle de erros na camada de transporte é fim a fim, mas o controle de erros na camada de enlace de dados é nó a nó, ao longo do enlace. Em outras palavras, cada vez que um quadro passa por um enlace, precisamos ter certeza de que ele não está corrompido.

5.2.3 Detecção e correção de erros

Na camada de enlace de dados, se um quadro for corrompido entre os dois nós, ele precisa ser corrigido antes de continuar sua jornada até os outros nós. No entanto, a maioria dos protocolos da camada de enlace simplesmente descarta o quadro e deixa que os protocolos da camada superior lidem com a retransmissão dele. Alguns protocolos sem fio, entretanto, tentam corrigir o quadro corrompido.

Introdução

Primeiramente, discutiremos algumas questões relacionadas, direta ou indiretamente, à detecção e correção de erros.

Tipos de erros

Sempre que os *bits* trafegam de um ponto a outro, eles estão sujeitos a modificações imprevisíveis causadas por **interferências**, que podem alterar a forma do sinal. O termo **erro de um único *bit*** significa que apenas um *bit* de uma determinada unidade de dados (por exemplo, um *byte*, um caractere ou um pacote) é alterado de 1 para 0 ou de 0 para 1. O termo **erro em rajada** significa que 2 ou mais *bits* na unidade de dados são alterados de 1 para 0 ou de 0 para 1. A Figura 5.8 mostra os efeitos de um erro de um único *bit* e de um erro em rajada sobre uma unidade de dados.

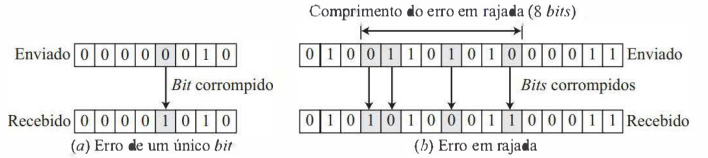


Figura 5.8 Erro de um único *bit* e erro em rajada.

Um erro em rajada é mais provável de ocorrer do que um erro de um único *bit*, porque a duração do sinal de ruído é normalmente maior que a duração de 1 *bit*, isto é, quando o ruído afeta os dados, ele afeta um conjunto de *bits*. O número de *bits* afetados depende da taxa de transmissão de dados e da duração do ruído. Por exemplo, se estivermos enviando dados a 1 kbps, um ruído com

duração de 1/100 segundos pode afetar 10 *bits*; se estivermos enviando dados a 1 Mbps, o mesmo ruído pode afetar 10.000 *bits*.

Redundância

O conceito central da detecção ou correção de erros é a *redundância*. Para sermos capazes de detectar ou corrigir erros, precisamos enviar alguns *bits* extras juntamente com os dados. Estes *bits* redundantes são adicionados pelo remetente e removidos pelo receptor. A sua presença permite que o receptor detecte ou corrija *bits* corrompidos.

Detecção *versus* correção

A correção de erros é mais difícil que a sua detecção. Na *detecção de erros*, estamos verificando apenas se algum erro ocorreu; nesse caso, a resposta é simplesmente sim ou não. Não estamos nem mesmo interessados no número de *bits* corrompidos. Não interessa saber se ocorreu um erro de um único *bit* ou um erro em rajada. Já na *correção de erros*, precisamos saber o número exato de *bits* que foram corrompidos e, ainda mais importante, a sua localização na mensagem. O número de erros e o tamanho da mensagem são fatores importantes. Se precisarmos corrigir um único erro em uma unidade de dados de oito *bits*, precisamos considerar oito possíveis locais de erro; se precisarmos corrigir dois erros em uma unidade de dados do mesmo tamanho, precisamos considerar 28 possibilidades (combinação de oito posições tomadas duas a duas). Não é difícil imaginar a dificuldade do receptor em encontrar 10 erros em uma unidade de dados de 1.000 *bits*. A seguir, nós nos concentraremos na detecção de erros; a correção de erros é mais difícil, mas será discutida brevemente no Capítulo 8.

Codificação

A redundância é obtida por meio de diversos esquemas de codificação. O remetente adiciona *bits* redundantes por meio de um processo que cria uma relação entre os *bits* redundantes e os *bits* de dados reais. O receptor verifica a relação entre os dois conjuntos de *bits* para detectar erros. A razão entre o número de *bits* redundantes e de *bits* de dados, bem como a robustez do processo, são fatores importantes em qualquer esquema de codificação.

Podemos dividir os esquemas de codificação em duas grandes categorias: *codificação de bloco* e *codificação convolucional*. Neste livro, nós nos concentramos na codificação de bloco; a codificação convolucional é mais complexa e está além do escopo desta obra.

Codificação de bloco

Na codificação de bloco, dividimos a mensagem em blocos, cada um tendo k *bits*, chamados **palavras de dados**. Adicionamos r *bits* redundantes a cada bloco para obter o comprimento $n = k + r$. Os blocos de n *bits* resultantes são denominados **palavras de código**. A forma como os r *bits* extras são escolhidos ou calculados é algo que discutiremos mais adiante. Por ora, é importante saber que temos um conjunto de palavras de dados, cada uma de tamanho k, e um conjunto de palavras de código, cada uma de tamanho n. Com k *bits*, podemos criar uma combinação de 2^k palavras de dados; com n *bits*, podemos criar uma combinação de 2^n palavras de código. Como $n > k$, o número de palavras de código possíveis é maior que o número de palavras de dados possíveis. O processo de codificação de bloco é um para um; uma determinada palavra de dados é sempre codificada como a mesma palavra de código. Isso significa que temos $2^n - 2^k$ palavras de código que não são utilizadas. Dizemos que essas palavras de código são inválidas ou ilegais. O truque na detecção de erros depende da existência desses códigos inválidos, conforme discutiremos a seguir. Se o receptor receber uma palavra de código inválida, isto indica que os dados foram corrompidos durante a transmissão.

Detecção de erros

Como os erros podem ser detectados usando codificação de bloco? Se as duas condições seguintes forem satisfeitas, o receptor pode detectar uma alteração na palavra de código original.

1. O receptor tem (ou pode determinar) uma lista de palavras de código válidas.
2. A palavra de código original foi transformada em uma palavra de código inválida.

A Figura 5.9 mostra o papel da codificação de bloco na detecção de erros.

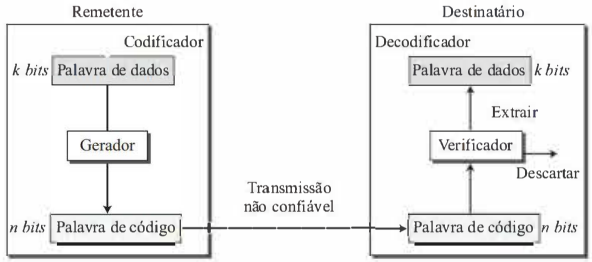


Figura 5.9 Processo de detecção de erros na codificação de bloco.

O remetente cria palavras de código a partir de palavras de dados usando um gerador que aplica as regras e procedimentos de codificação (discutidos mais adiante). Cada palavra de código enviada para o receptor pode vir a ser modificada durante a transmissão. Se a palavra de código recebida corresponde a uma palavra de código válida, a palavra é aceita; neste caso, a palavra de código correspondente é extraída e usada. Se a palavra de código recebida não for válida, ela é descartada. No entanto, se a palavra de código for corrompida durante a transmissão e a palavra recebida ainda corresponder a uma palavra de código válida, o erro passa despercebido.

Exemplo 5.1

Consideremos que $k = 2$ e $n = 3$. A Tabela 5.1 mostra a lista de palavras de dados e palavras de código. Mais adiante, veremos como obter uma palavra de código a partir de uma palavra de dados.

Tabela 5.1 Um código para a detecção de erros do Exemplo 5.1.

Palavras de dados	Palavras de código	Palavras de dados	Palavras de código
00	000	10	101
01	011	11	110

Considere que o remetente codifica a palavra de dados 01 como 011 e a envia para o receptor. Considere os seguintes casos:

1. O receptor recebe 011, uma palavra de código válida. O receptor extrai a palavra de dados 01 a partir dela.
2. A palavra de código é corrompida durante a transmissão, e o valor 111 é recebido (o *bit* mais à esquerda foi corrompido). Não é uma palavra de código válida, portanto, ela é descartada.
3. A palavra de código é corrompida durante a transmissão, de modo que o valor recebido é 000 (os dois *bits* da direita foram corrompidos). É uma palavra de código válida. O receptor extrai incorretamente a palavra de dados 00. Os dois *bits* corrompidos tornaram o erro indetectável.

Um código de detecção de erros pode detectar apenas os tipos de erros para os quais ele foi projetado; outros tipos de erros podem passar despercebidos.

Distância de Hamming

Um dos conceitos centrais da codificação voltada ao controle de erros é a ideia da **distância de Hamming**. A distância de Hamming entre duas palavras (do mesmo tamanho) corresponde ao número de diferenças entre os *bits* correspondentes. Representamos a distância de Hamming entre duas palavras x e y como $d(x, y)$. Podemos nos perguntar por que a distância de Hamming é importante para a detecção de erros. A razão é que a distância de Hamming entre a palavra de código recebida e a palavra de código enviada equivale ao número de *bits* que foram corrompidos durante a transmissão. Por exemplo, se a palavra de código 00000 é enviada e 01101 é recebida, 3 *bits* foram corrompidos e a distância de Hamming entre as duas palavras é $d(00000, 01101) = 3$. Ou seja, se a distância de Hamming entre as palavras de código enviadas e recebidas não for zero, a palavra de código foi corrompida durante a transmissão.

A distância de Hamming pode ser facilmente encontrada se aplicarmos a operação de XOR (também conhecida como OU-Exclusivo e representada como $\oplus$) sobre as duas palavras e contar o número de 1s no resultado. Perceba que a distância de Hamming é um valor maior ou igual a zero.

A distância de Hamming entre duas palavras é o número de diferenças entre *bits* correspondentes.

Exemplo 5.2

Determinaremos a distância de Hamming entre dois pares de palavras.

1. A distância de Hamming $d(000, 011)$ é 2 porque $(000 \oplus 011)$ é 011 (dois 1s).
2. A distância de Hamming $d(10101, 11110)$ é 3 porque $(10101 \oplus 11110)$ é 01011 (três 1s).

Distância de Hamming mínima para a detecção de erros

Em um conjunto de palavras de código, a distância de Hamming mínima corresponde à menor distância de Hamming entre todos os pares possíveis de palavras de código. Agora, determinaremos a distância de Hamming mínima de um código para que ele seja capaz de detectar até s erros. Se s erros ocorrerem durante a transmissão, a distância de Hamming entre as palavras de código enviadas e recebidas é s. Se o nosso sistema deve detectar até s erros, a distância mínima entre as palavras de código válidas deve ser $(s + 1)$, de modo que a palavra de código recebida não corresponda a uma palavra de código válida. Em outras palavras, se a distância mínima entre todas as palavras de código válidas for $(s + 1)$, a palavra de código recebida não pode ser erroneamente confundida com outra palavra de código. O erro será detectado. Precisamos esclarecer um ponto aqui: apesar de um código com $d_{min} = s + 1$ ser capaz de detectar mais do que s erros em alguns casos especiais, quando s ou menos erros ocorrem, tem-se a garantia de que eles serão detectados.

Para garantir a detecção de até s erros em todos os casos, a distância de Hamming mínima em um código de bloco deve ser $d_{min} = s + 1$.

Podemos observar esse critério geometricamente. Suponha que a palavra de código enviada x esteja no centro de um círculo de raio s. Todas as palavras de código recebidas criadas por 0 a s erros correspondem a pontos no interior do círculo ou no perímetro do círculo. Todas as outras palavras de código válidas devem estar fora do círculo, conforme mostra a Figura 5.10. Isto significa que d_{min} deve ser um número inteiro maior que s $d_{min} = s + 1$.

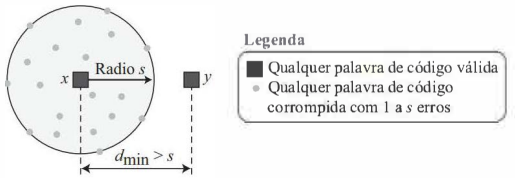


Figura 5.10 Conceito geométrico explicando d_{min} no contexto de detecção de erros.

Exemplo 5.3

A distância de Hamming mínima para o nosso primeiro esquema de codificação (Tabela 5.1) é 2. Este código garante apenas a detecção de um único erro. Por exemplo, se a terceira palavra de código (101) for enviada e um erro ocorrer, a palavra de código recebida não corresponde a uma palavra de código válida. Caso ocorram dois erros, no entanto, a palavra de código recebida pode corresponder a uma palavra de código válida e os erros não serão detectados.

Exemplo 5.4

Um esquema de codificação tem uma distância de Hamming $d_{min} = 4$. Esse código garante a detecção de até três erros ($d = s + 1$ ou $s = 3$).

Códigos de bloco lineares

Quase todos os códigos de bloco usados atualmente pertencem a um subconjunto de códigos de bloco denominados *códigos de bloco lineares*. O uso de códigos de bloco não lineares para detecção e correção de erros não é tão generalizado porque a estrutura deles dificulta a sua análise teórica e implementação. Portanto, vamos nos concentrar nos códigos de bloco lineares, cuja definição requer o conhecimento de álgebra abstrata (particularmente corpos de Galois), algo que está além do escopo deste livro. Portanto, apresentamos uma definição informal. Para nossos propósitos, um código de bloco linear é um código em que a operação de XOR (adição módulo 2) de duas palavras de código válidas resulta em uma outra palavra de código válida.

Exemplo 5.5

O código da Tabela 5.1 é um código de bloco linear porque o resultado de aplicar a operação de XOR sobre qualquer par de palavras de código resulta em uma palavra de código válida. Por exemplo, fazer o XOR da segunda com a terceira palavras de código cria a quarta palavra de código.

Distância mínima em códigos de bloco lineares

É simples calcular a distância de Hamming mínima de um código de bloco linear. A distância de Hamming mínima corresponde ao número de 1s na palavra de código válida com o menor número de 1s, porém diferente de zero.

Exemplo 5.6

No nosso primeiro código (Tabela 5.1), os números de 1s nas palavras de código diferentes de zero são 2, 2 e 2. Assim, a distância de Hamming mínima é $d_{min} = 2$.

Código de paridade

Talvez o código de detecção de erros mais conhecido seja o **código de paridade**, que é um código de bloco linear. Nele, uma palavra de dados de k *bits* é transformada em uma palavra de código de n *bits* onde $n = k + 1$. O *bit* extra, denominado *bit* de paridade, é selecionado de modo a fazer com que o número total de 1s na palavra de código seja par. Embora algumas implementações especifiquem um número ímpar de 1s, discutimos apenas o caso par. A distância de Hamming mínima para essa categoria é $d_{min} = 2$, o que significa que o código é detector de erros de um único *bit*. Nosso primeiro código (Tabela 5.1) é de paridade ($k = 2$ e $n = 3$). O código da Tabela 5.2 também é de paridade com $k = 4$ e $n = 5$.

Tabela 5.2 Código de paridade simples C(5, 4).

Palavras de dados	Palavras de código	Palavras de dados	Palavras de código
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

A Figura 5.11 mostra uma possível estrutura de um codificador (no remetente) e de um decodificador (no destinatário).

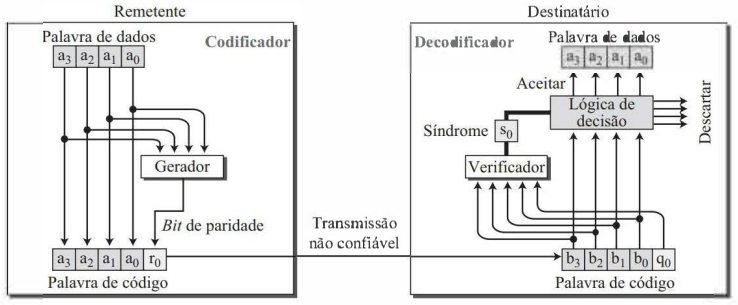


Figura 5.11 Codificador e decodificador para um código de paridade simples.

O codificador usa um gerador que toma uma cópia de uma palavra de dados de 4 *bits* (a_0, a_1, a_2 e a_3) e gera um *bit* de paridade r_0. Os *bits* da palavra de dados e o *bit* de paridade criam a palavra de código de 5 *bits*. O *bit* de paridade adicionado faz com que o número de 1s na palavra de código

seja par. Isso é normalmente obtido por meio da soma dos 4 *bits* da palavra de dados (módulo 2); o resultado é o *bit* de paridade. Em outras palavras,

$$r_0 = a_3 + a_2 + a_1 + a_0 \qquad \textbf{(módulo 2)}$$

Se o número de 1s for par, o resultado é 0; se for ímpar, o resultado é 1. Em ambos os casos, o número total de 1s na palavra de código é par.

O remetente envia a palavra de código, a qual pode ser corrompida durante a transmissão. O destinatário recebe uma palavra de 5 *bits*. O verificador no receptor faz a mesma coisa que o gerador no remetente, mas com uma exceção: a adição é feita considerando todos os 5 *bits*. O resultado, chamado **síndrome**, corresponde a apenas 1 *bit*. A síndrome é 0 quando o número de 1s na palavra de código recebida for par; caso contrário, a síndrome é 1.

$$s_0 = b_3 + b_2 + b_1 + b_0 + q_0 \qquad \textbf{(módulo 2)}$$

A síndrome é passada para o analisador de lógica de decisão. Se a síndrome for 0, não há erro detectável na palavra de código recebida; a parte de dados da palavra de código recebida é aceita como palavra de dados. Já se a síndrome for 1, a parte de dados recebida da palavra de código é descartada. A palavra de dados não é criada.

Exemplo 5.7

Vejamos alguns cenários de transmissão. Considere que o remetente envia a palavra de dados 1011. A palavra de código criada a partir dessa palavra de dados é 10111, enviada para o receptor. Examinemos cinco casos:

1. Nenhum erro ocorre; a palavra de código recebida é 10111. A síndrome é 0. A palavra de dados 1011 é criada.
2. Um erro de um único *bit* altera a_1. A palavra de código recebida é 10011. A síndrome é 1. Nenhuma palavra de dados é criada.
3. Um erro de um único *bit* altera r_0. A palavra de código recebida é 10110. A síndrome é 1. Nenhuma palavra de dados é criada. Observe que, embora nenhum dos *bits* da palavra de dados esteja corrompido, nenhuma palavra de dados é criada porque o código não é suficientemente sofisticado para mostrar a posição do *bit* corrompido.
4. Um erro altera r_0 e um segundo erro altera a_3. A palavra de código recebida é 00110. A síndrome é 0. A palavra de dados 0011 é criada no receptor. Perceba que, aqui, a palavra de dados é erroneamente criada devido ao valor da síndrome. O decodificador de paridade simples não é capaz de detectar um número par de erros. Os erros se anulam mutuamente e levam a um valor de síndrome igual a 0.
5. Três *bits* – a_3, a_2 e a_1 – são alterados devido a erros. A palavra de código recebida é 01011. A síndrome é 1. A palavra de dados não é criada. Isto mostra que a verificação de paridade simples, que garante a detecção de um único erro, pode também detectar qualquer número ímpar de erros.

Um código de paridade é capaz de detectar um número ímpar de erros.

Códigos cíclicos

Códigos cíclicos são um tipo especial de códigos de bloco lineares que possuem uma propriedade extra. Em um **código cíclico**, se uma palavra de código for deslocada ciclicamente (rotacionada), o resultado é outra palavra de código. Por exemplo, se 1011000 for um palavra de código, podemos deslocá-la ciclicamente para a esquerda e obter 0110001, que também é uma palavra de código. Nesse caso, se nomearmos os *bits* da primeira palavra de a_0 a a_6 e os *bits* na segunda palavra de b_0 a b_6, podemos deslocar os *bits* conforme segue:

$$b_1 = a_0 \quad b_2 = a_1 \quad b_3 = a_2 \quad b_4 = a_3 \quad b_5 = a_4 \quad b_6 = a_5 \quad b_0 = a_6$$

Na equação mais à direita, o último *bit* da primeira palavra é rotacionado ciclicamente e torna-se o primeiro *bit* da segunda palavra.

Verificação de Redundância Cíclica

Podemos criar códigos cíclicos para corrigir erros. No entanto, o embasamento teórico necessário está além do escopo deste livro. Nesta seção, discutimos simplesmente um subconjunto de códigos cíclicos denominados **Verificação de Redundância Cíclica** (CRC – Cyclic Redundancy Check), que são usados em redes como LANs e WANs.

A Tabela 5.3 mostra um exemplo de um código CRC. Podemos observar as propriedades lineares e cíclicas desse código.

Tabela 5.3 Um código CRC com C(7, 4).

Palavra de dados	Palavra de código	Palavra de dados	Palavra de código
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

A Figura 5.12 mostra uma possível estrutura para o codificador e para o decodificador.

No codificador, a palavra de dados tem k *bits* (aqui, $k = 4$); a palavra de código tem n *bits* (aqui, $n = 7$). O tamanho da palavra de dados é expandido pela adição de $n - k$ (aqui, 3) 0s à direita da palavra. O resultado de n *bits* é alimentado no gerador, que usa um divisor de tamanho $n - k + 1$ (aqui, 4), predefinido e pré-acordado. O gerador divide a palavra de dados expandida pelo divisor (divisão módulo 2). O quociente da divisão é descartado, o resto ($r_2 r_1 r_0$) é concatenado com a palavra de dados para criar a palavra de código.

O decodificador recebe a palavra de código (possivelmente corrompida durante a transmissão). Uma cópia de todos os n *bits* é alimentada no verificador, que é uma réplica do gerador. O resto produzido pelo verificador é uma síndrome de $n - k$ (aqui, 3) *bits*, fornecida ao analisador da lógica de decisão. Se os *bits* da síndrome forem todos 0s, os 4 *bits* mais à esquerda da palavra de código são considerados a palavra de dados (interpretada como livre de erros); caso contrário, os 4 *bits* são descartados (erro).

Codificador Analisemos o codificador mais de perto. Ele recebe uma palavra de dados e a expande com um número de 0s igual a $n - k$. Em seguida, ele divide a palavra de dados expandida pelo divisor, conforme mostra a Figura 5.13.

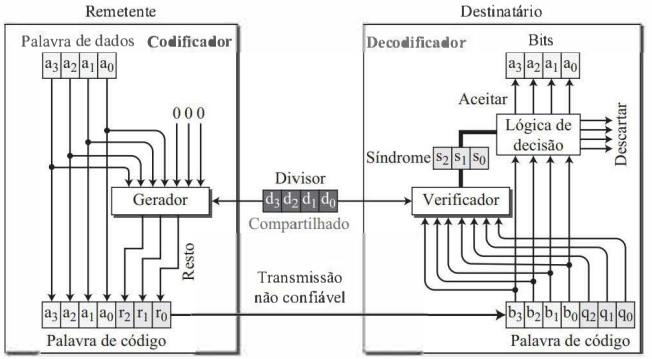


Figura 5.12 Codificador e decodificador CRC.

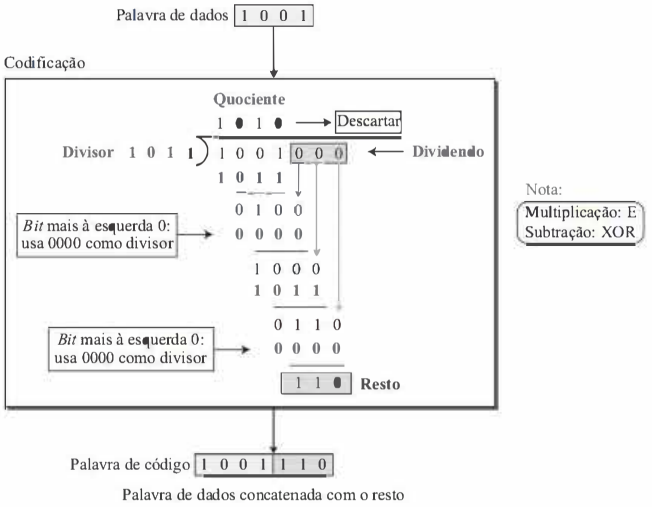


Figura 5.13 Divisão no codificador CRC.

O processo de divisão binária módulo 2 é equivalente ao processo de divisão geralmente utilizado para números decimais. Entretanto, nesse caso, o mesmo não vale para a adição e a subtração; usamos a operação XOR para efetuar ambas as operações.

Assim como na divisão decimal, o processo é feito passo a passo. Em cada passo, é feito o XOR de uma cópia do divisor com os 4 *bits* do dividendo. O resultado da operação de XOR (resto) tem 3 *bits* (nesse caso), que são usados no passo seguinte após um *bit* extra ser trazido para baixo, de

modo a deixá-lo com um comprimento de 4 *bits*. Há um ponto importante que precisamos lembrar nesse tipo de divisão: se o *bit* mais à esquerda do dividendo (ou a parte utilizada em cada passo) for 0, o passo não pode utilizar o divisor normal; precisamos usar um divisor composto apenas por 0s.

Quando não há *bits* para trazer para baixo, temos o resultado final. O resto, de 3 *bits*, corresponde aos *bits* de verificação (r_2, r_1 e r_0). Ele é concatenado à palavra de dados para criar a palavra de código.

Decodificador A palavra de código pode ser alterada durante a transmissão. O decodificador executa o mesmo processo de divisão realizado no codificador. O resto da divisão corresponde à síndrome. Se ela for composta apenas por 0s, existe uma elevada probabilidade de não ter havido erros; a palavra de dados é separada da palavra de código recebida e aceita. Caso contrário, todos os *bits* são descartados. A Figura 5.14 mostra dois casos: a figura do lado esquerdo mostra o valor da síndrome quando não há erros; a síndrome, nesse caso, é 000. O lado direito da figura mostra o caso em que há erro em um único *bit*. A síndrome não é composta apenas por 0s (ela vale 011).

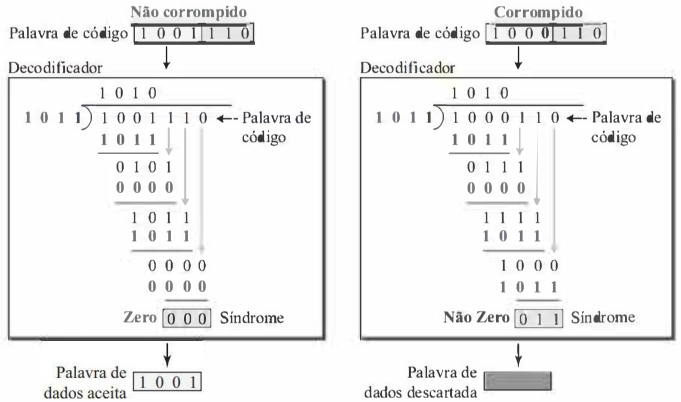


Figura 5.14 Divisão no decodificador CRC em duas situações.

Divisor Podemos estar nos perguntando como o divisor 1011 foi escolhido. Isto depende do que se espera do código. Discutimos os critérios no *site* www.grupoa.com.br. Alguns dos divisores-padrão utilizados na área de redes são mostrados na Tabela 5.4. O número no nome do divisor (por exemplo, CRC-32) refere-se ao grau do polinômio (a sua maior potência) que representa o divisor. O número de *bits* é sempre uma unidade a mais do que o grau do polinômio. Por exemplo, o CRC-8 tem 9 *bits* e o CRC-32 tem 33 *bits*.

Tabela 5.4 Polinômios-padrão.

Nome	Binário	Aplicação
CRC-8	100000111	Cabeçalho ATM
CRC-10	11000110101	ATM AAL
CRC-16	10001000000100001	HDLC
CRC-32	100000100110000010001110110110111	LANs

Polinômios

Uma maneira melhor de entender os códigos cíclicos e a forma como eles podem ser analisados é representá-los como polinômios. Discutimos polinômios no *site* www.grupoa.com.br para os leitores interessados.

Requisitos

Podemos comprovar matematicamente que uma sequência de *bits* precisa ter pelo menos duas propriedades para ser considerada um gerador (divisor):

1. A sequência deve ter pelo menos dois *bits*.
2. Os *bits* mais à direita e mais à esquerda devem ser ambos 1s.

Eficácia

Podemos comprovar matematicamente a eficácia do CRC conforme mostrado a seguir.

- **Erros individuais.** Todos os geradores que satisfazem os requisitos anteriores são capazes de detectar qualquer erro de um único *bit*.
- **Número ímpar de erros.** Todos os geradores que satisfazem os requisitos anteriores são capazes de detectar qualquer número ímpar de erros, contanto que o gerador seja divisível por $(11)_2$ usando divisão binária em aritmética módulo 2; caso contrário, apenas alguns erros em um número ímpar de *bits* serão detectados.
- **Erros em rajada.** Se considerarmos que o comprimento do erro em rajada é de L *bits* e que *r* é o comprimento do resto (*r* é o comprimento do gerador menos 1; ele é também o valor da potência mais elevada no polinômio que representa o gerador):
  a. Todos os erros em rajada de tamanho $L \leq r$ são detectados.
  b. Todos os erros em rajada de tamanho $L = r + 1$ são detectados com probabilidade $1 - (0,5)^{r-1}$.
  c. Todos os erros em rajada de tamanho $L > r + 1$ são detectados com probabilidade $1 - (0,5)^{r}$.

Vantagens de códigos cíclicos

Os códigos cíclicos podem ser facilmente implementados em *hardware* e *software*. Eles são especialmente rápidos quando implementados em *hardware*, o que tornou os códigos cíclicos bons candidatos para serem usados em muitas redes. No *site* www.grupoa.com.br, mostramos como a divisão pode ser feita usando um registrador de deslocamento que está incluído no *hardware* do nó.

Soma de verificação

A soma de verificação, ou *checksum*, é uma técnica de detecção de erros que pode ser aplicada a uma mensagem de qualquer tamanho. Na Internet, a técnica de soma de verificação é usada principalmente nas camadas de rede e de transporte e não na camada de enlace de dados. Entretanto, para tornar a nossa discussão sobre técnicas de detecção de erros mais completa, discutimos a soma de verificação (*checksum*) neste capítulo.

Na origem, a mensagem é primeiramente dividida em unidades de *m bits*. O gerador cria, então, uma unidade extra de *m bits* denominada *soma de verificação*, enviada com a mensagem. No destino, o verificador cria uma nova soma de verificação a partir da combinação da mensagem e da soma de verificação recebida. Se a nova soma de verificação for composta apenas por 0s, a mensagem é aceita; caso contrário, a mensagem é descartada (Figura 5.15). Perceba que, em uma aplicação real, a unidade da soma de verificação não é necessariamente adicionada ao final da mensagem; ela pode ser inserida no meio da mensagem.

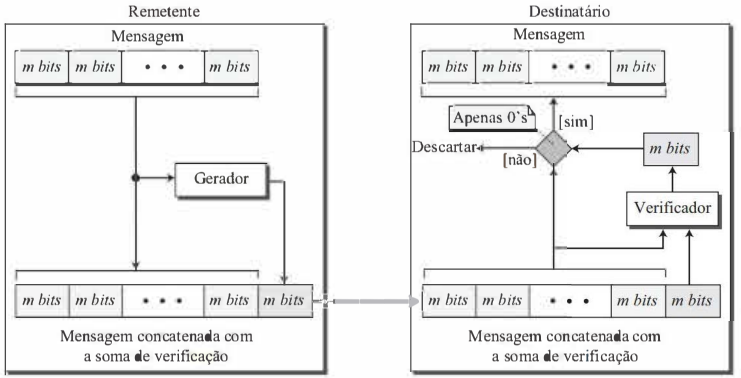


Figura 5.15 Soma de verificação (*checksum*).

Conceitos

A ideia por trás da soma de verificação tradicional é simples. Mostramos isto usando um exemplo simples.

Exemplo 5.8

Considere que a mensagem seja uma lista de cinco números de 4 *bits* que queremos enviar a um destino. Além de enviar esses números, enviamos também a soma deles. Por exemplo, se o conjunto de números for (7, 11, 12, 0, 6), enviamos (7, 11, 12, 0, 6, **36**), onde 36 é a soma dos números originais. O receptor soma os cinco números e compara o resultado com a soma recebida. Se os dois resultados forem iguais, o receptor considera que não houve qualquer erro, aceita os cinco números e então descarta a soma. Caso contrário, há um erro em algum lugar e a mensagem não é aceita.

Adição em complemento de um O exemplo anterior apresenta um grande problema. Cada número pode ser escrito como uma palavra de 4 *bits* (todos eles são menores do que 15), exceto pela soma. Uma solução é usar aritmética em **complemento de um**. Nesse tipo de aritmética, podemos representar números sem sinal entre 0 e $2^m - 1$ usando apenas *m bits*. Se o número tem mais do que *m bits*, os *bits* extras mais à esquerda precisam ser somados aos *m bits* mais à direita.

Exemplo 5.9

No exemplo anterior, o número decimal 36 em binário é escrito como $(100100)_2$. Para transformá-lo em um número de 4 *bits*, somamos o *bit* extra mais à esquerda aos quatro *bits* da direita, conforme mostrado a seguir.

$$(10)_2 + (0100)_2 = (0110)_2 \rightarrow (6)_{10}$$

Em vez de enviar 36 como o valor da soma, podemos enviar o valor 6, ou seja, (7, 11, 12, 0, 6, **6**). O destinatário pode somar os cinco primeiros números usando aritmética em complemento de um. Se o resultado for 6, os números são aceitos; caso contrário, eles são rejeitados.

Soma de verificação Podemos facilitar o trabalho do destinatário se enviarmos o complemento da soma, conhecido como soma de verificação, ou *checksum*. Na aritmética em complemento de um, o complemento de um número é encontrado invertendo-se todos os *bits* (transformando todos os 1s em 0s e todos os 0s em 1s). Isto equivale a subtrair o número de $2^m - 1$. Na aritmética em complemento de um, temos dois 0s: um positivo e um negativo, que são o complemento um do outro. O zero positivo tem todos os *m bits* valendo 0; o zero negativo tem todos os *bits* valendo 1 (ele corresponde a $2^m - 1$). Se somarmos um número ao seu complemento, obtemos um zero negativo (um número com todos os *bits* valendo 1). Quando o destinatário soma todos os cinco números (incluindo a soma de verificação), ele obtém um zero negativo. O destinatário pode complementar o resultado novamente para obter um zero positivo.

Exemplo 5.10

Apliquemos a ideia da soma de verificação no Exemplo 5.9. O remetente soma todos os cinco números usando aritmética em complemento de um para obter a soma = 6. O remetente, então, complementa o resultado para obter a soma de verificação = 9, que é o resultado de 15 – 6. Observe que $6 = (0110)_2$ e $9 = (1001)_2$, que são complementos um do outro. O remetente envia os cinco números (dados) juntamente com a soma de verificação (**7**, 11, 12, 0, 6, **9**). Se não houver qualquer corrupção durante a transmissão, o destinatário recebe (**7**, 11, 12, 0, 6, **9**) e soma esses números usando aritmética em complemento de um para obter o valor 15. O remetente complementa o valor 15 para obter 0. Isso mostra que os dados não foram corrompidos. A Figura 5.16 ilustra o processo.

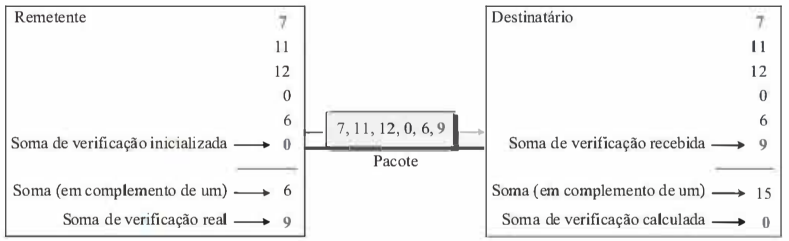


Figura 5.16 Esquema para o Exemplo 5.10.

Soma de verificação na Internet

Tradicionalmente, a Internet usa uma soma de verificação de 16 *bits*. O remetente e o destinatário seguem os passos descritos na Tabela 5.5. São cinco os passos usados tanto pelo remetente como pelo destinatário.

Algoritmo

Podemos usar o diagrama de fluxo da Figura 5.17 para mostrar o algoritmo do cálculo da soma de verificação. Um programa em qualquer linguagem pode ser facilmente escrito com base nesse algoritmo. Observe que o primeiro laço apenas calcula a soma das unidades de dados em complemento de dois; o segundo laço rotaciona os *bits* extras criados pelo cálculo em complemento de dois para simular os cálculos em complemento de um. Isto é necessário porque quase todos os computadores atuais fazem cálculos em complemento de dois.

Tabela 5.5 Procedimento para calcular a soma de verificação tradicional.

Remetente	Destinatário
1. A mensagem é dividida em palavras de 16 *bits*.	1. A mensagem e a soma de verificação são recebidas.
2. O valor da soma de verificação é inicialmente fixado em zero.	2. A mensagem é dividida em palavras de 16 *bits*.
3. Todas as palavras, incluindo a soma de verificação, são somadas usando soma em complemento de um.	3. Todas as palavras são somadas usando soma em complemento de um.
4. A soma é complementada e torna-se a soma de verificação.	4. A soma é complementada e torna-se a soma de verificação.
5. A soma de verificação é enviada juntamente com os dados.	5. Se o valor da soma de verificação for 0, a mensagem é aceita; caso contrário, ela é rejeitada.

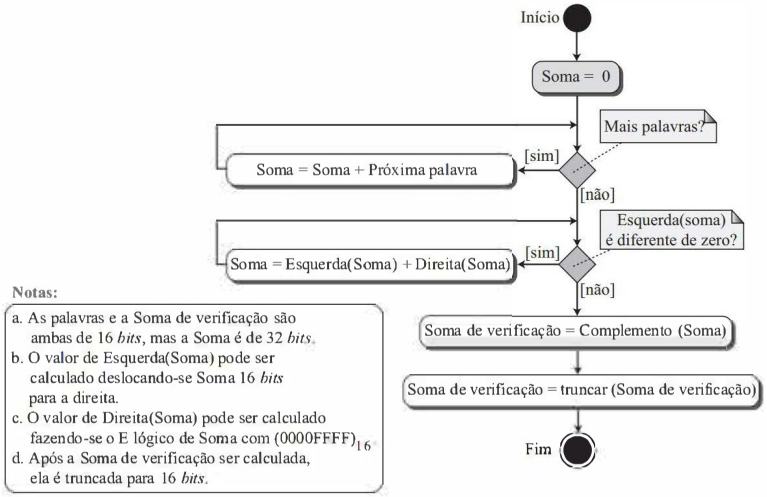


Figura 5.17 Algoritmo para calcular uma soma de verificação tradicional.

Eficácia A soma de verificação tradicional usa um número pequeno de *bits* (16) para detectar erros em uma mensagem de qualquer tamanho (algumas vezes milhares de *bits*). No entanto, essa técnica não é tão efetiva quanto o CRC com relação à sua capacidade de verificar erros. Por exemplo, se o valor de uma palavra é incrementado e o valor de outra palavra é decrementado da mesma quantidade, os dois erros não podem ser detectados porque a soma de verificação permanece a mesma. Além disso, se os valores de várias palavras forem incrementados, mas a sua soma e a soma de verificação não se alterarem, os erros não serão detectados. Fletcher e Adler propuseram algumas somas de verificação ponderadas que eliminam o primeiro desses problemas. No entanto, a tendência na Internet, particularmente no projeto de novos protocolos, é substituir a soma de verificação pelo CRC.

Outras abordagens para a soma de verificação

Conforme mencionado anteriormente, existe um grande problema com o cálculo da soma de verificação tradicional. Se dois itens de dados de 16 *bits* forem invertidos durante a transmissão, a soma de verificação não será capaz de detectar esse erro. A razão é que a soma de verificação tradicional não é ponderada: ela trata cada item de dados de forma idêntica. Em outras palavras, a ordem dos itens de dados é irrelevante para o cálculo. Várias abordagens podem ser utilizadas para evitar esse problema. Mencionamos duas delas aqui: Fletcher e Adler.

Soma de verificação de Fletcher A soma de verificação de Fletcher foi concebida para atribuir um peso a cada item de dados de acordo com sua posição. Fletcher propôs dois algoritmos: um de 8 e outro de 16 *bits*. O primeiro, Fletcher de 8 *bits*, faz os cálculos sobre itens de dados de 8 *bits* e gera uma soma de verificação de 16 *bits*. O segundo, Fletcher de 16 *bits*, faz os cálculos sobre itens de dados de 16 *bits* e gera uma soma de verificação de 32 *bits*.

O Fletcher de 8 *bits* é calculado sobre octetos (*bytes*) de dados e gera uma soma de verificação de 16 *bits*. O cálculo é feito em módulo 256 (2^8), o que significa que os resultados intermediários são divididos por 256 e o resto é mantido. O algoritmo utiliza dois acumuladores, L e R. O primeiro simplesmente soma itens de dados uns aos outros; o segundo introduz o peso no cálculo. Existem diversas variantes do algoritmo de Fletcher de 8 *bits*; mostramos uma variante simples na Figura 5.18.

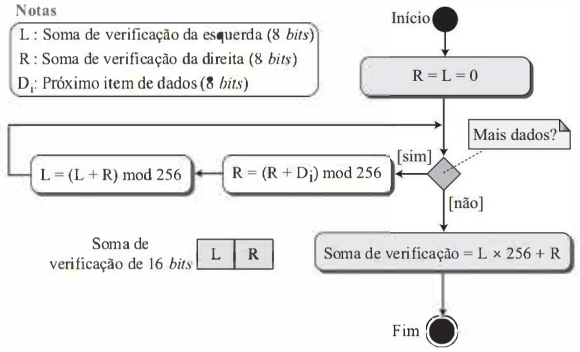


Figura 5.18 Algoritmo para calcular uma soma de verificação de Fletcher de 8 *bits*.

A soma de verificação Fletcher de 16 *bits* é semelhante à soma de verificação Flecher de 8 *bits*, mas é calculada sobre itens de dados de 16 *bits* e gera uma soma de verificação de 32 *bits*. O cálculo é feito em módulo 65.536.

Soma de verificação de Adler A soma de verificação de Adler é uma soma de verificação de 32 *bits*. A Figura 5.19 mostra um algoritmo simples na forma de um fluxograma. Essa abordagem é semelhante à abordagem de Fletcher de 16 *bits*, com três diferenças. Primeiro, o cálculo é realizado em *bytes* individuais em vez de 2 *bytes* de cada vez. Segundo, o módulo é um número primo (65.521) em vez de 65.536. Terceiro, L é inicializado com o valor 1 em vez de 0. Prova-se que um módulo primo tem uma melhor capacidade de detecção em algumas combinações de dados.

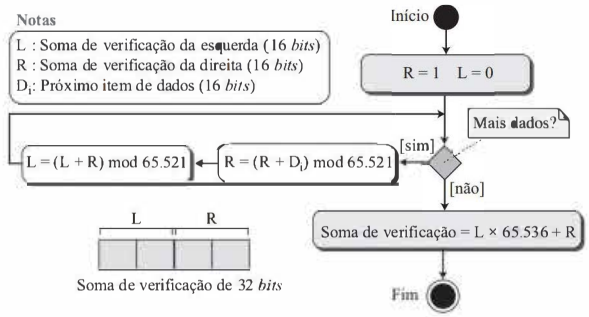


Figura 5.19 Algoritmo para calcular a soma de verificação de Adler.

5.2.4 Dois protocolos de DLC

Agora que terminamos a discussão de todas as questões relacionadas à subcamada DLC, discutimos dois protocolos DLC que de fato implementam tais conceitos. O primeiro, denominado HDLC, é a base de muitos protocolos que foram projetados para LANs. O segundo, conhecido como PPP, é um protocolo derivado do HDLC e é usado para enlaces ponto a ponto.

HDLC

O **Controle de Enlace de Dados de Alto Nível** (HDLC – High-level Data Link Control) é um protocolo orientado a *bits* para comunicação através de enlaces ponto a ponto ou multiponto. Ele implementa o protocolo *Stop-and-Wait* que discutimos no Capítulo 3.

Configurações e modos de transferência

O HDLC fornece dois modos comuns de transferência que podem ser usados em diferentes configurações: *Modo de Resposta Normal* (NRM – Normal Response Mode) e *Modo Balanceado Assíncrono* (ABM – Asynchronous Balanced Mode). No NRM, a configuração da estação é desbalanceada. Temos uma estação principal e diversas estações secundárias. Uma *estação primária* pode enviar comandos; uma *estação secundária* pode apenas responder a eles. O NRM é usado tanto para enlaces ponto a ponto como multiponto, conforme mostra a Figura 5.20.

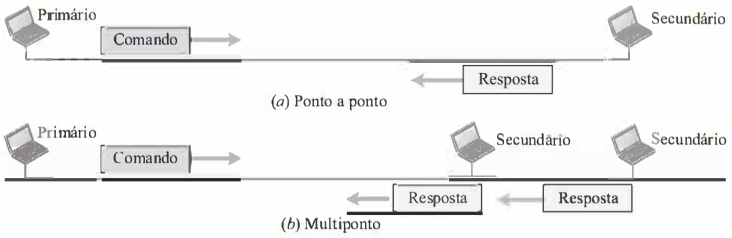


Figura 5.20 Modo de resposta normal.

No modo ABM, a configuração é balanceada. O enlace é ponto a ponto e cada estação pode operar como uma estação primária e secundária (atuando como *peers*), conforme mostra a Figura 5.21. Esse é o modo mais comum atualmente.

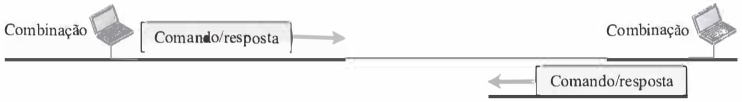


Figura 5.21 Modo balanceado assíncrono.

Quadros

Com o objetivo de fornecer a flexibilidade necessária para suportar todas as opções possíveis nos modos e configurações que acabamos de descrever, o HDLC define três tipos de quadros: *quadros de informação* (quadros-I), *quadros de supervisão* (quadros-S) e *quadros não numerados* (quadros-U, de *unnumbered*). Cada tipo de quadro serve como um envelope para a transmissão de um tipo diferente de mensagem. Os quadros-I são utilizados para transportar dados do usuário e informações de controle relativas aos dados do usuário (mecanismo de carona, ou *piggybacking*). Quadros-S são usados apenas para transportar informações de controle. Quadros-U são reservados para o gerenciamento do sistema. Informações transportadas por quadros-U destinam-se ao gerenciamento do próprio enlace. Cada quadro no HDLC pode conter até seis campos, conforme mostra a Figura 5.22: um campo marcador de início, um campo de endereço, um campo de controle, um campo de informação, um campo de Sequência de Verificação de Quadro (FCS – Frame Check Sequence) e um campo marcador do final do quadro. Nas transmissões de múltiplos quadros, o marcador de fim de um quadro pode servir como o marcador de início do próximo quadro.

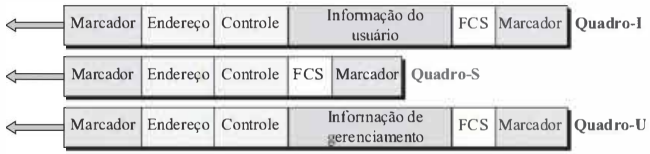


Figura 5.22 Quadros HDLC.

Analisemos os campos e seu uso em diferentes tipos de quadros.

- **Campo marcador.** Contém a sequência de *bits* de sincronização 01111110, que identifica tanto o início como o fim de um quadro.
- **Campo de endereço.** Contém o endereço da estação secundária. Se uma estação principal criou o quadro, esse campo contém o endereço do *destino*. Se uma estação secundária criou o quadro, ele contém o endereço da *origem*. O comprimento do campo de endereço pode ser um *byte* ou vários *bytes*, dependendo das necessidades da rede.
- **Campo de controle.** Consiste em um ou dois *bytes* usados para o controle de fluxo e de erros. A interpretação dos *bits* será discutida mais adiante.
- **Campo de informação.** Contém dados do usuário provenientes da camada de rede ou informação de gerenciamento. O seu comprimento pode variar de uma rede para outra.
- **Campo FCS.** É o campo de detecção de erros do HDLC. Ele contém um CRC de 2 ou de 4 *bytes*.

O campo de controle determina o tipo de quadro e define sua funcionalidade. Portanto, vamos discutir o formato desse campo em detalhes. O formato é específico para o tipo de quadro, conforme mostra a Figura 5.23.

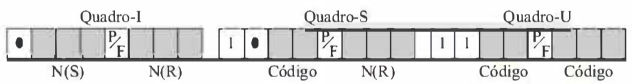


Figura 5.23 Formato do campo de controle para diferentes tipos de quadros.

Campo de controle dos quadros-I Os quadros-I foram projetados para transportar dados do usuário provenientes da camada de rede. Além disso, eles podem incluir informações de controle de fluxo e de erros (mecanismo de carona, ou *piggybacking*). Os subcampos no campo de controle são utilizados para definir essas funções. O primeiro *bit* define o tipo. Se o primeiro *bit* do campo de controle for 0, isto significa que o quadro é um quadro-I. Os próximos 3 *bits*, denotados $N(S)$, definem o número de sequência do quadro. Perceba que, com 3 *bits*, podemos definir um número de sequência entre 0 e 7. Os últimos 3 *bits*, denominados $N(R)$, correspondem ao número de confirmação quando o mecanismo de carona é usado. O único *bit* entre $N(S)$ e $N(R)$ é chamado *bit P/F*. O campo *P/F* consiste em um único *bit* com dupla finalidade. Ele só tem algum significado quando vale 1, podendo significar consulta ou final: significa *consulta* quando o quadro é enviado por uma estação primária para uma secundária (quando o campo de endereço contém o endereço do destinatário) e *final* quando o quadro é enviado de uma estação secundária para uma primária (quando o campo de endereço contém o endereço do remetente).

Campo de controle dos quadros-S Quadros de supervisão são usados para o controle de fluxo e de erros sempre que o mecanismo de carona for impossível de ser usado ou inadequado. Os quadros-S não apresentam campos de informação. Se os primeiros 2 *bits* do campo de controle forem 10, isto significa que o quadro é um quadro-S. Os últimos 3 *bits*, denotados $N(R)$, correspondem ao número de confirmação positiva (ACK) ou negativa (NAK), dependendo do tipo de quadro-S. Os 2 *bits* denominados *código* são usados para definir o tipo do quadro-S em si. Com 2 *bits*, podemos ter quatro tipos de quadros-S, descritos a seguir:

- **Receptor pronto** (RR – Receive Ready). Se o valor do subcampo de código for 00, o quadro-S é do tipo RR, que confirma a recepção correta de um quadro (ou grupo de quadros). Nesse caso, o valor do campo $N(R)$ define o número de confirmação.
- **Receptor não pronto** (RNR – Receive Not Ready). Se o valor do subcampo de código for 10, o quadro-S é do tipo RNR, que corresponde a um quadro RR com funções adicionais. Ele confirma o recebimento de um quadro ou grupo de quadros, mas anuncia que o receptor está ocupado e não pode receber mais quadros. Ele atua como uma espécie de mecanismo de controle de congestionamento, pedindo ao remetente para reduzir a taxa de transmissão. O valor de $N(R)$ define o número de confirmação.
- **Rejeição** (REJ – Reject). Se o valor do subcampo de código for 01, o quadro-S é do tipo REJ, um quadro de NAK, mas não como aquele usado no ARQ da Repetição Seletiva. Trata-se de um NAK que pode ser usado no ARQ *Go-Back-N* para melhorar a eficiência do processo, informando ao remetente, antes que o temporizador do remetente expire, que o último quadro foi perdido ou danificado. O valor do campo $N(R)$ corresponde ao número de confirmação negativa.
- **Rejeição seletiva** (SREJ – Selective Reject). Se o valor do subcampo de código for 11, o quadro-S é um SREJ, um quadro de NAK usado no ARQ de Repetição Seletiva. Perceba que o protocolo HDLC utiliza o termo *rejeição seletiva*, em vez de *repetição seletiva*. O valor do campo $N(R)$ corresponde ao número da confirmação negativa.

Campo de controle dos quadros-U Quadros não numerados são usados para trocar informações de gerenciamento de sessão e de controle entre dispositivos conectados. Ao contrário dos quadros-S, os quadros-U carregam um campo de informação, mas ele é utilizado para informações do sistema de gerenciamento, não para dados do usuário. Assim como acontece com os quadros-S, no entanto, boa parte das informações transportadas por quadros-U está contida em códigos incluídos no campo de controle. Os códigos dos quadros-U são divididos em duas seções: um prefixo de 2 *bits* antes do *bit* P/F e um sufixo de 3 *bits* após o *bit* P/F. Em conjunto, esses dois segmentos (5 *bits*) podem ser usados para criar até 32 tipos de quadros-U diferentes.

Protocolo Ponto a Ponto

Um dos protocolos mais usados para acesso ponto a ponto é o **Protocolo Ponto a Ponto** (PPP – Point-to-Point Protocol). Atualmente, milhões de usuários da Internet que precisam conectar seus computadores domésticos ao servidor de um ISP utilizam o PPP. A maioria desses usuários possui um modem tradicional; eles se conectam à Internet por meio de uma linha telefônica, a qual fornece os serviços da camada física. Entretanto, para controlar e gerenciar a transferência de dados, é necessário um protocolo ponto a ponto na camada de enlace de dados. O PPP é, de longe, o mais comum.

Serviços

Os projetistas do PPP incluíram diversos serviços para torná-lo adequado para uso como um protocolo ponto a ponto, mas ignoraram alguns serviços tradicionais com o objetivo de simplificá-lo.

Serviços fornecidos pelo PPP O PPP define o formato do quadro a ser trocado entre os dispositivos e também como dois dispositivos podem negociar o estabelecimento da conexão e a troca de dados. O PPP foi projetado para aceitar dados de várias camadas de rede (não apenas IP). A autenticação também é prevista no protocolo, mas é opcional. A nova versão do PPP, denominada *PPP Multienlace*, fornece conexões sobre múltiplos enlaces. Uma característica interessante do PPP é que ele fornece configuração de endereços de rede. Isto é particularmente útil quando um usuário doméstico precisa de um endereço de rede temporário para se conectar à Internet.

Serviços não fornecidos pelo PPP O PPP não fornece controle de fluxo. Um remetente pode enviar vários quadros um após o outro sem se importar com possíveis sobrecargas no receptor. O PPP fornece um mecanismo muito simples para controle de erros. Um campo de CRC é utilizado para detectar erros. Se o quadro estiver corrompido, ele é descartado silenciosamente; o protocolo da camada superior precisa tratar o problema. A ausência de controle de erros e de números de sequência pode fazer com que um pacote seja recebido fora de ordem. O PPP não oferece um mecanismo sofisticado de endereçamento para lidar com quadros em uma configuração multiponto.

Enquadramento

O PPP utiliza quadros orientados a caracteres (ou orientados a *bytes*). A Figura 5.24 mostra o formato de um quadro PPP. A descrição de cada campo é a seguinte:

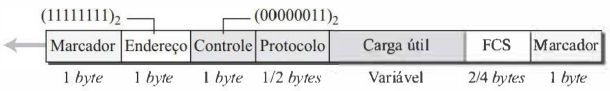


Figura 5.24 Formato do quadro PPP.

- **Marcador.** Um quadro PPP começa e termina com um marcador de 1 *byte* contendo a sequência de *bits* 01111110.
- **Endereço.** O campo de endereço neste protocolo é um valor constante igual a 11111111 (endereço de *broadcast*).
- **Controle.** Campo que tem o valor fixo e constante de 00000011 (imitando os quadros-U do HDLC). Conforme discutiremos mais adiante, o PPP não fornece qualquer controle de fluxo. O controle de erros também é limitado à detecção de erros.
- **Protocolo.** O campo de protocolo define o que está sendo transportado no campo de dados: dados de usuário ou outras informações. Por padrão, esse campo tem 2 *bytes* de comprimento, mas as duas partes podem concordar em usar apenas 1 *byte*.
- **Campo de carga útil (*Payload*).** Carrega os dados do usuário ou outras informações que discutiremos em breve. O campo de dados é uma sequência de *bytes* cujo tamanho máximo por padrão é 1.500 *bytes*; mas esse valor pode ser alterado durante a negociação. Aplica-se o mecanismo de preenchimento de *byte* sobre o campo de dados caso a sequência de *bytes* do marcador de início ou fim apareça nesse campo. Como não há qualquer campo definindo o tamanho do campo de dados, a adição de *bits* de enchimento (*padding*) é necessária se o tamanho for menor do que o valor máximo padrão ou aquele definido via negociação.
- **FCS.** A **Sequência de Verificação de Quadro** (FCS – Frame Check Sequence) é simplesmente um CRC padrão de 2 ou 4 *bytes*.

Preenchimento de *byte* Como o PPP é um protocolo orientado a *bytes*, o marcador no PPP é um *byte* ao qual deve ser adicionada uma sequência de escape sempre que ele aparecer na seção de dados do quadro. O *byte* de escape é 01111101, o que significa que toda vez que uma sequência equivalente a um marcador aparecer nos dados, este *byte* extra é adicionado para informar o receptor de que o próximo *byte* não é um marcador. Obviamente, o *byte* de escape em si deve ser preenchido com outro *byte* de escape.

Transição de fases

Uma conexão PPP passa por fases que podem ser mostradas em um diagrama de *transição de fases* (ver Figura 5.25). O diagrama de transição começa no estado *morto*. Nesse estado, não há qualquer portadora ativa (na camada física) e a linha está silenciosa. Quando um dos dois nós inicia a comunicação, a conexão vai para o estado *estabelecer*. Nele, as opções são negociadas entre as duas partes. Se elas concordam em fazer uma autenticação, o sistema vai para o estado *autenticar*; caso contrário, vai para o estado *rede*. Os pacotes do protocolo de controle de enlace, que será discutido em breve, são usados para essa finalidade. Diversos pacotes podem ser trocados nesse ponto. A transferência de dados acontece no estado *aberto*. Quando uma conexão atinge esse estado, a troca de pacotes de dados pode ser iniciada. A conexão permanece assim até que um dos terminais comunicantes decida encerrar a conexão. Nesse caso, o sistema vai para o estado *encerrar* e permanece nesse estado até que a portadora (sinal da camada física) desapareça, o que novamente leva o sistema para o estado *morto*.

Multiplexação

Embora o PPP seja um protocolo da camada de enlace, ele usa outro conjunto de protocolos para estabelecer a conexão, autenticar as partes envolvidas e transportar os dados da camada de rede. Três conjuntos de protocolos são definidos para tornar o PPP mais poderoso: o Protocolo de Controle de Enlace (LCP – Link Control Protocol), dois Protocolos de Autenticação (APs – Authentication Protocols) e vários Protocolos de Controle de Rede (NCPs – Network Control Protocols). A qualquer momento, um pacote PPP pode transportar dados provenientes de um desses protocolos em seu campo de dados, conforme mostra a Figura 5.26. Observe que há um LCP, dois APs, e vários NCPs. Os dados também podem vir de diversas camadas de rede diferentes.

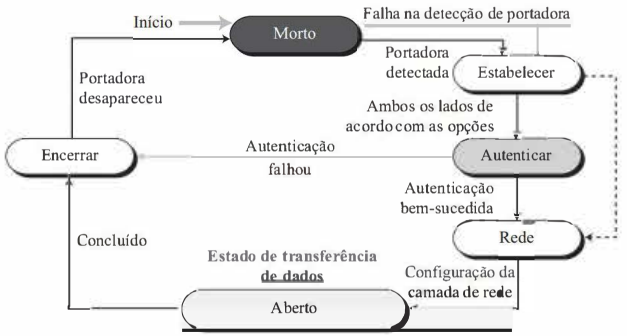


Figura 5.25 Transição de fases.

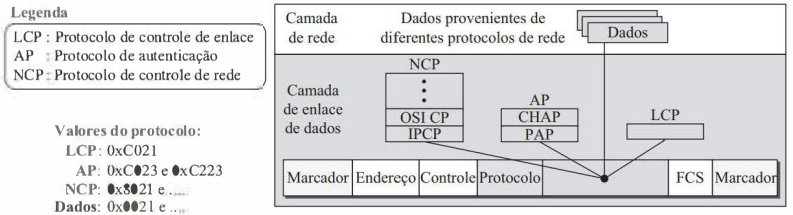


Figura 5.26 Multiplexação no PPP.

Protocolo de Controle de Enlace Este Protocolo (LCP – Link Control Protocol) é responsável por estabelecer, manter, configurar e encerrar conexões. Ele também fornece mecanismos de negociação para definir as opções entre os dois pontos finais. Os pontos finais do enlace devem ambos chegar a um acordo sobre as opções antes que a conexão seja estabelecida.

Protocolos de autenticação A autenticação desempenha um papel muito importante no PPP, pois o protocolo foi projetado para ser utilizado em conexões discadas nas quais a verificação da identidade do usuário é necessária. *Autenticação* significa validar a identidade de um usuário que precisa acessar um conjunto de recursos. O PPP criou dois protocolos de autenticação: o PAP e o CHAP. Note que esses protocolos são utilizados durante a fase de autenticação.

- **PAP.** O *Protocolo de Autenticação por Senha* (PAP – Password Authentication Protocol) é um mecanismo de autenticação simples cujo processo consiste em duas etapas:
  - **a.** O usuário que quer acessar um sistema envia uma identificação de autenticação (normalmente o nome de usuário) e uma senha.
  - **b.** O sistema verifica a validade do par identificação/senha e aceita ou recusa a conexão.
- **CHAP.** *O Protocolo de Autenticação por Desafio-Resposta* (CHAP – Challenge Handshake Authentication Protocol) é um protocolo de autenticação de três vias que proporciona maior segurança do que o PAP. Nesse método, a senha é mantida em sigilo; nunca é enviada pela rede.

a. O sistema envia ao usuário um pacote de desafio contendo o valor de um desafio, que geralmente consiste em alguns *bytes*.

b. O usuário aplica uma função predefinida, que pega o valor do desafio e a senha do próprio usuário e cria um resultado. O usuário envia o resultado para o sistema em um pacote de resposta.

c. O sistema faz o mesmo. Ele aplica a mesma função sobre a senha do usuário (conhecida pelo sistema) e sobre o valor do desafio para criar um resultado; se este for igual ao resultado enviado no pacote de resposta, o acesso é autorizado; caso contrário, o acesso é negado. O CHAP é mais seguro que o PAP, especialmente se o sistema alterar constantemente o valor de desafio. Mesmo que o intruso descubra o valor do desafio e o resultado, a senha continua em segredo.

Protocolos de controle de rede O PPP é um protocolo que aceita múltiplas camadas de rede. Ele pode transportar pacotes de dados da camada de rede provenientes de protocolos definidos pela Internet, OSI, Xerox, DECnet, AppleTalk, Novel, e assim por diante. Para que isto seja possível, o PPP definiu um Protocolo de Controle de Rede (NCP – Network Control Protocol) específico para cada protocolo de rede. Por exemplo, o **Protocolo de Controle de Rede da Internet** (IPCP – Internet Protocol Control Protocol) configura o enlace para que ele transporte pacotes de dados provenientes do IP. O Xerox CP faz o mesmo para os pacotes de dados do protocolo Xerox, e assim por diante. Perceba que nenhum dos pacotes NCP transporta dados da camada de rede; eles apenas configuram o enlace para suportar os dados recebidos dela. O IPCP, protocolo usado para configurar o enlace de modo a transportar pacotes IP na Internet, é de especial interesse para nós.

Dados provenientes da camada de rede Após a configuração da camada de rede ser completada por um dos protocolos NCP, os usuários podem trocar pacotes de dados vindos da camada de rede. Aqui, novamente, há campos de protocolo distintos para diferentes camadas de rede. Por exemplo, se o PPP estiver transportando dados provenientes da camada de rede IP, o valor do campo é $(0021)_{16}$. Se o PPP estiver transportando dados provenientes da camada de rede OSI, o valor do campo de protocolo é $(0023)_{16}$, e assim por diante.

PPP Multienlace

O PPP foi originalmente concebido para uso em um enlace físico ponto a ponto com um único canal. A disponibilidade de múltiplos canais em um mesmo enlace ponto a ponto motivou o desenvolvimento do PPP Multienlace. Nesse caso, um quadro PPP lógico é dividido em vários quadros PPP reais. Um segmento do quadro lógico é transportado como carga útil de um quadro PPP real, conforme mostra a Figura 5.27. Para mostrar que o quadro PPP real está carregando um fragmento de um quadro PPP lógico, o campo de protocolo é preenchido com o valor $(003d)_{16}$. Essa nova funcionalidade introduz maior complexidade. Por exemplo, é preciso adicionar um número de sequência ao quadro PPP real para mostrar a posição de um fragmento no quadro lógico.

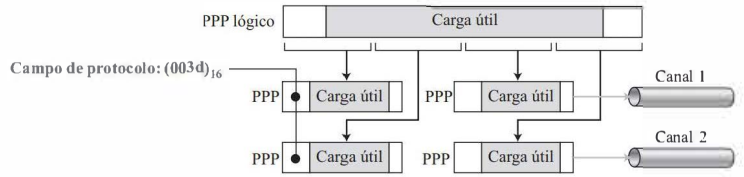


Figura 5.27 PPP Multienlace.

5.3 PROTOCOLOS DE ACESSO MÚLTIPLO

Dissemos que a camada de enlace de dados é dividida em duas subcamadas: Controle de Enlace de Dados (DLC – Data Link Control) e Controle de Acesso ao Meio (MAC – Media Access Control). Discutimos o DLC na seção anterior; nesta seção, falaremos sobre o MAC. Quando estamos usando uma conexão dedicada, como uma linha telefônica discada, precisamos apenas de um protocolo de controle de enlace de dados, tal como o PPP, que gerencia a transferência de dados entre as duas pontas. Por outro lado, se estivermos compartilhando o meio, seja ele fios ou o ar, com outros usuários, precisamos ter um protocolo para primeiramente gerenciar o processo de compartilhamento e então fazer a transferência de dados. Por exemplo, se usarmos um telefone celular para ligar para outro telefone celular, o canal (a banda alocada pela operadora de telefonia) não é dedicado. Uma pessoa a poucos metros de distância pode estar usando a mesma banda para conversar com um amigo.

Quando nós ou estações estão conectados e usando um enlace em comum, denominado enlace multiponto ou de *broadcast*, precisamos de um protocolo de acesso múltiplo para coordenar o acesso a tal enlace. O problema de controlar o acesso ao meio é semelhante às regras de conversação em uma reunião. Os procedimentos garantem que o direito de fala seja concedido a todos, que duas pessoas não falem ao mesmo tempo, que não interrompam umas às outras, que não monopolizem a discussão, e assim por diante. A situação é semelhante nas redes multiponto. Precisamos ter certeza de que cada nó terá acesso ao enlace. O primeiro objetivo é evitar qualquer colisão entre os nós. Se de alguma forma uma colisão ocorrer, o segundo objetivo é resolvê-la.

Muitos protocolos foram concebidos para controlar o acesso a um enlace compartilhado. Podemos categorizá-los em três grupos. Os protocolos pertencentes a cada grupo são mostrados na Figura 5.28.

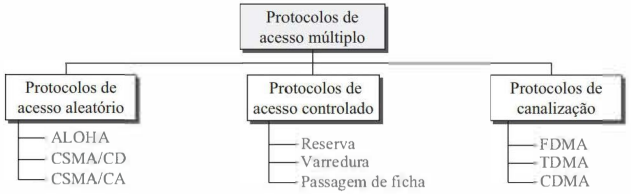


Figura 5.28 Taxonomia de protocolos de acesso múltiplo discutidos neste capítulo.

5.3.1 Acesso aleatório

Nos métodos de **acesso aleatório** ou **contenção**, nenhuma estação tem preferência ou controle sobre outra estação. A cada instante, uma estação que tem dados para enviar usa um procedimento definido pelo protocolo para tomar a decisão sobre se ela deve ou não enviar os dados. Essa decisão depende do estado do meio (ocioso ou ocupado). Em outras palavras, cada estação pode transmitir quando ela desejar, com a condição de que ela siga o procedimento pré-estabelecido, incluindo uma verificação do estado do meio.

Duas características dão a esse método o seu nome. Em primeiro lugar, não há instantes programados nos quais uma estação pode transmitir. A transmissão é aleatória entre as estações. É por isso que esses métodos são denominados *acesso aleatório*. Em segundo lugar, não há regras específicas sobre qual estação deve se a próxima a enviar dados. As estações competem umas com as outras pelo acesso ao meio. É por essa razão que esses métodos são também chamados métodos de *contenção*.

Em um método de acesso aleatório, cada estação tem direito de acessar o meio sem ser controlada por qualquer outra estação. Entretanto, se mais de uma estação tentar enviar, existe um conflito de acesso – *colisão* – e os quadros serão destruídos ou modificados. Para evitar conflitos de acesso ou para resolvê-los quando eles ocorrerem, cada estação segue um procedimento que responde às seguintes perguntas:

- Quando a estação pode acessar o meio?
- O que a estação pode fazer se o meio estiver ocupado?
- Como a estação pode determinar o sucesso ou a falha na transmissão?
- O que a estação pode fazer se houver um conflito de acesso?

Os métodos de acesso aleatório que estudamos neste capítulo evoluíram a partir de um protocolo muito interessante conhecido como ALOHA, o qual utilizava um procedimento muito simples denominado *Acesso Múltiplo* (MA – Multiple Access). O método foi aperfeiçoado com a adição de um procedimento que força a estação a verificar o estado do meio antes de transmitir. O resultado foi denominado *Acesso Múltiplo com Detecção de Portadora* (CSMA – Carrier Sense Multiple Access). Mais tarde, esse método evoluiu para dois métodos paralelos: *Acesso Múltiplo com Detecção de Portadora/Detecção de Colisão* (CSMA/CD – Carrier Sense Multiple Access/Collision Detection), que diz o que a estação deve fazer quando uma colisão é detectada, e o *Acesso Múltiplo com Detecção de Portadora/Prevenção de Colisão* (CSMA/CA – Carrier Sense Multiple Access/Collision *Avoidance*), que tenta evitar colisões.

ALOHA

O **ALOHA,** o método de acesso aleatório mais antigo, foi desenvolvido na Universidade do Havaí no início de 1970. Ele foi projetado para uso em LANs por rádio (sem fios), mas pode ser utilizado em qualquer meio compartilhado.

É óbvio que há colisões potenciais nesse cenário. O meio (ar) é compartilhado entre as estações. Quando uma estação envia dados, uma outra estação pode tentar fazer o mesmo simultaneamente. Os dados das duas estações colidem e acabam sendo corrompidos.

ALOHA puro

O protocolo ALOHA original é conhecido como **ALOHA puro**. É um protocolo simples, porém elegante. A ideia é que cada estação envie um quadro sempre que tiver um quadro para enviar (acesso múltiplo). No entanto, como existe apenas um canal compartilhado, é possível que ocorram colisões entre os quadros das diferentes estações. A Figura 5.29 mostra um exemplo de colisões de quadros no ALOHA puro.

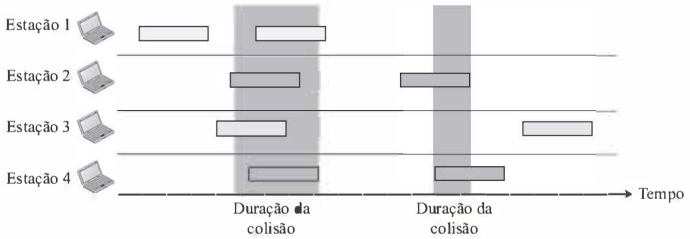


Figura 5.29 Quadros em uma rede usando ALOHA puro.

Existem quatro estações (hipótese pouco realista) que competem umas com as outras pelo acesso ao canal compartilhado. A figura mostra que cada estação envia dois quadros; há, portanto, um total de oito quadros no meio compartilhado. Alguns desses quadros colidem porque vários quadros estão disputando o canal compartilhado. A Figura 5.29 mostra que apenas dois sobrevivem: um quadro da estação 1 e um da estação 3. É importante mencionar que, mesmo que apenas um *bit* de um quadro coexista no canal ao mesmo tempo que um *bit* de outro, uma colisão ocorrerá e ambos os quadros serão destruídos. É óbvio que precisamos reenviar os quadros que foram destruídos durante a transmissão.

O protocolo ALOHA puro depende de confirmações (ACKs) provenientes do receptor. Quando uma estação envia um quadro, ela espera que o receptor envie uma confirmação. Se a confirmação não chegar depois de um tempo-limite, a estação considera que o quadro (ou a confirmação) foi destruído e reenvia o quadro.

Uma colisão pode envolver duas ou mais estações. Se todas essas estações tentarem reenviar os seus quadros logo após o tempo-limite, os quadros colidirão novamente. O ALOHA puro dita que, depois de esgotado o tempo-limite, cada estação espera um intervalo de tempo aleatório antes de reenviar seu quadro. A aleatoriedade ajuda a evitar novas colisões. Denominamos esse intervalo *tempo de retardo* ou *tempo de* back-off, que denotamos por T_B.

O ALOHA puro inclui um segundo método para evitar congestionamentos no canal devido a quadros retransmitidos. Após um número máximo de tentativas de retransmissão K_{max}, uma estação deve desistir e tentar mais tarde. A Figura 5.30 mostra o procedimento usado pelo ALOHA puro com base na estratégia anterior.

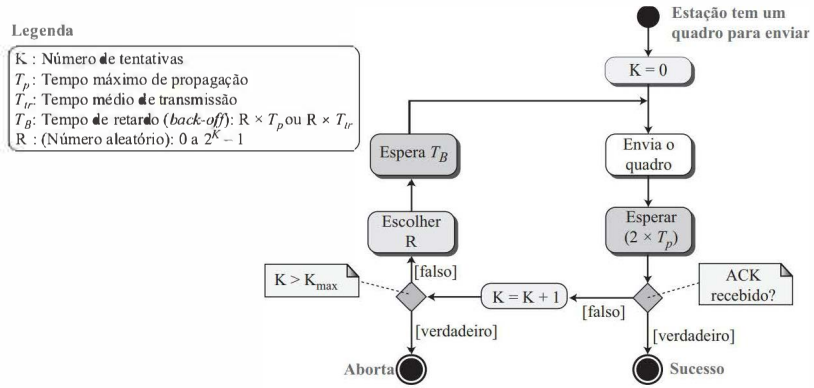


Figura 5.30 Procedimento do protocolo ALOHA puro.

O valor do tempo-limite é igual ao máximo valor possível do atraso de propagação de ida e volta, que é o dobro do tempo necessário para enviar um quadro entre as duas estações que estão separadas pela maior distância ($2 \times T_p$). O tempo de retardo T_B é um valor aleatório que normalmente depende de K (o número de tentativas de transmissão malsucedidas). A fórmula para calcular T_B depende da implementação. Uma fórmula comum é o *retardo exponencial binário*. Nesse método, para cada retransmissão, um multiplicador de $R = 0$ a $2^K - 1$ é escolhido aleatoriamente e multiplicado por T_p (tempo máximo de propagação) ou T_{tr} (o tempo médio necessário para enviar um quadro), resultando no valor de T_B. Perceba que, nesse procedimento, a faixa de números aleatórios aumenta após cada colisão. O valor de K_{max} é geralmente escolhido como 15.

Exemplo 5.11

As estações em uma rede sem fios ALOHA estão separadas por no máximo 600 km de distância. Se considerarmos que os sinais se propagam a uma velocidade de 3×10^8 m/s, verificamos que $T_p = (600 \times 10^3) / (3 \times 10^8) = 2$ ms. Para $K = 2$, os valores que R pode assumir são {0, 1, 2, 3}. Isto significa que T_b pode ser 0, 2, 4 ou 6 ms, dependendo do valor da variável aleatória *R*.

Período vulnerável Calculemos o intervalo de tempo, denominado *período vulnerável*, em que há possibilidade de colisão. Consideramos que as estações enviam quadros de comprimento fixo e que cada quadro leva T_{tr} segundos para ser enviado. A Figura 5.31 mostra o período vulnerável para a estação B.

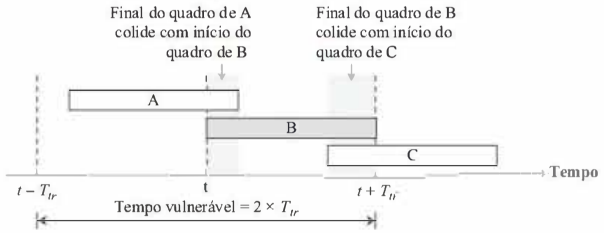


Figura 5.31 Período vulnerável para o protocolo ALOHA puro.

A estação B começa a enviar um quadro no instante *t*. Considere que a estação A começou a enviar seu quadro após o instante $t - T_{tr}$. Isto leva a uma colisão entre os quadros vindos das estações A e B. Além disso, considere que a estação C comece a enviar um quadro antes do instante $t + T_{tr}$. Nesse caso, também ocorre uma colisão entre os quadros das estações B e C.

Observando a Figura 5.31, vemos que o período vulnerável no ALOHA puro, durante o qual uma colisão pode ocorrer, corresponde a duas vezes o intervalo de transmissão de quadros.

Período vulnerável do ALOHA puro = $2 \times T_{tr}$

Exemplo 5.12

Uma rede usando ALOHA puro transmite quadros de 200 *bits* em um canal compartilhado de 200 kbps. Qual é o requisito para que esse quadro seja transmitido livre de colisões?

Solução

O tempo médio de transmissão de quadros T_b é de 200 *bits*/200 kbps ou 1 ms. O período vulnerável vale 2×1 ms = 2 ms. Isto significa que nenhuma estação deve começar a enviar quadros 1 ms antes de essa estação iniciar sua transmissão e que nenhuma estação deve começar a enviar quadros durante o período em que essa estação está transmitindo (1 ms).

Vazão Denotamos por *G* o número médio de quadros gerados pelo sistema durante um intervalo de transmissão de quadros. Pode-se, então, provar que o número médio de quadros transmitidos com sucesso pelo ALOHA puro é $V = G \times e^{-2G}$. A vazão máxima V_{max} é 0,184, para $G = 1/2$. (Podemos encontrar esse valor fazendo a derivada de *V* em função de *G* igual a 0, conforme mostrado na seção de Exercícios). Em outras palavras, se meio quadro for gerado em intervalos equivalentes a um intervalo de transmissão de quadros (um quadro para cada dois intervalos de transmissão de quadros), então 18,4% desses quadros chegarão ao seu destino com sucesso. Espera-se que $G = 1/2$ produza a vazão máxima porque o período vulnerável será duas vezes o intervalo de transmissão de quadros. Portanto, se uma estação gera apenas um quadro nesse período vulnerável (e nenhuma outra estação gera um quadro durante esse intervalo), o quadro vai chegar com sucesso ao seu destino.

A vazão do ALOHA puro é $V = G \times e^{-2G}$.
A vazão máxima é $V_{max} = 1/(2e) = 0{,}184$ quando $G = (1/2)$.

Exemplo 5.13

Uma rede usando ALOHA puro transmite quadros de 200 *bits* em um canal compartilhado de 200 kbps. Qual é a vazão total se o sistema (todas as estações juntas) gera

a. 1.000 quadros por segundo?
b. 500 quadros por segundo?
c. 250 quadros por segundo?

Solução

O intervalo de transmissão de quadros é 200/200 kbps ou 1 ms.

a. Se o sistema gera 1.000 quadros por segundo, ou 1 quadro por milissegundo, então $G = 1$. Nesse caso, $V = G \times e^{-2G} = 0{,}135$ (13,5%). Isto significa que a vazão é de $1.000 \times 0{,}135 = 135$ quadros. Apenas 135 quadros a cada 1.000 provavelmente sobreviverão.
b. Se o sistema gera 500 quadros por segundo, ou 1/2 quadro por milissegundo, então $G = 1/2$. Nesse caso, $V = G \times e^{-2G} = 0{,}184$ (18,4%). Isto significa que a vazão é de $500 \times 0{,}184 = 92$ e que apenas 92 quadros a cada 500 provavelmente sobreviverão. Note que esse é o caso de vazão *máxima*, em termos percentuais.
c. Se o sistema gera 250 quadros por segundo, ou 1/4 quadro por milissegundo, então $G = 1/4$. Nesse caso, $V = G \times e^{-2G} = 0{,}152$ (15,2%). Isto significa que a vazão é de $250 \times 0{,}152 = 38$. Apenas 38 quadros a cada 250 provavelmente sobreviverão.

Slotted ALOHA

O ALOHA puro tem um período vulnerável de $2 \times T_{tr}$. Isto acontece porque não existe uma regra que defina quando a estação pode transmitir. Uma estação pode enviar quadros logo após a outra estação já ter começado a fazê-lo ou logo antes de outra estação ter concluído sua transmissão. O *slotted* ALOHA foi inventado para melhorar a eficiência do ALOHA puro.

No ***slotted* ALOHA**, também conhecido como **ALOHA segmentado** ou **ALOHA discreto,** dividimos o tempo em intervalos de T_{tr} segundos, denominados *faixas* ou *slots*, e forçamos as estações a transmitir apenas no início de cada faixa. A Figura 5.32 mostra um exemplo de colisões de quadros no *slotted* ALOHA.

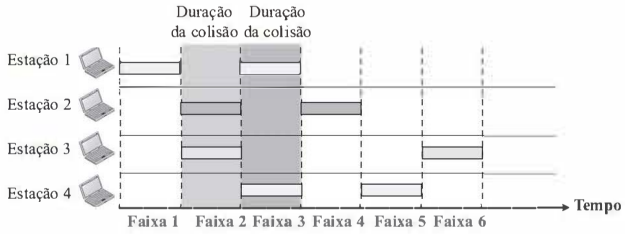


Figura 5.32 Quadros em uma rede baseada em *slotted* ALOHA.

Como as estações podem enviar apenas no início de cada faixa sincronizada de tempo, se uma estação deixa passar esse momento, ela deve esperar até o início da faixa seguinte. Isto significa que a estação que começou a transmissão no início dessa faixa já terá terminado o envio de seu quadro. É claro que ainda existe a possibilidade de colisão se duas estações tentarem transmitir seus quadros no início da mesma faixa. Entretanto, o período vulnerável é agora reduzido pela metade, passando a valer T_{tr}. A Figura 5.33 ilustra essa situação.

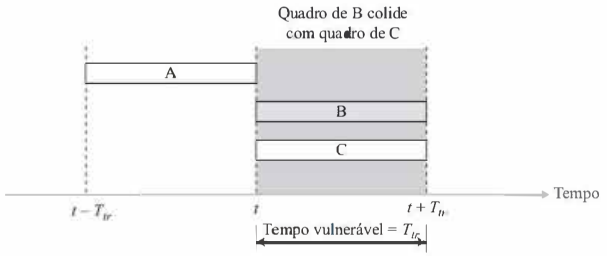


Figura 5.33 Período vulnerável para o protocolo *slotted* ALOHA.

Período vulnerável no *slotted* ALOHA = T_{tr}

Vazão Pode-se provar que o número médio de quadros transmitidos com sucesso pelo *slotted* ALOHA é dado por $V = G \times e^{-G}$. A vazão máxima V_{max} é 0,368, quando $G = 1$. Em outras palavras, se um quadro é gerado em intervalos equivalentes a um intervalo de transmissão de quadros, então 36,8% desses quadros chegarão ao seu destino com sucesso. Espera-se que $G = 1$ produza a vazão máxima porque o período vulnerável será igual ao intervalo de transmissão de quadros. Portanto, se uma estação gerar apenas um quadro nesse período vulnerável (e nenhuma outra estação gerar um quadro durante esse intervalo), o quadro chegará com sucesso ao seu destino.

A vazão para o *slotted* ALOHA é $S = G \times e^{-G}$.
A taxa de transferência máxima é $V_{max} = 0{,}368$ quando $G = 1$.

Exemplo 5.14

Uma rede baseada em *slotted* ALOHA transmite quadros de 200 *bits* usando um canal compartilhado com uma largura de banda de 200 kbps. Determine a vazão se o sistema (todas as estações juntas) gera

a. 1.000 quadros por segundo.
b. 500 quadros por segundo.
c. 250 quadros por segundo.

Solução

Essa situação é semelhante à do exercício anterior, exceto que a rede usa *slotted* ALOHA em vez do ALOHA puro. O intervalo de transmissão de quadros é 200/200 kbps ou 1 ms.

a. Nesse caso, $G = 1$. Portanto, $V = G \times e^{-G} = 0{,}368$ (36,8%). Isto significa que a vazão é de $1.000 \times 0{,}368 = 368$ quadros. Apenas 368 quadros a cada 1.000 provavelmente sobreviverão. Note que esse é o caso de vazão *máxima*, em termos percentuais.

b. Nesse caso, $G = 1/2$. Portanto, $V = G \times e^{-G} = 0{,}303$ (30,3%). Isto significa que a vazão é de $500 \times 0{,}303 = 151$. Apenas 151 quadros a cada 500 provavelmente sobreviverão.
c. Agora $G = 1/4$. Nesse caso, $V = G \times e^{-G} = 0{,}195$ (19,5%). Isto significa que a vazão é de $250 \times 0{,}195 = 49$. Apenas 49 quadros a cada 250 provavelmente sobreviverão.

Acesso Múltiplo com Detecção de Portadora

Para minimizar a possibilidade de colisões e, portanto, aumentar a vazão da rede, foi desenvolvido o método CSMA. A probabilidade de colisão pode ser reduzida se uma estação verificar o estado do meio antes de tentar usá-lo. O **Acesso Múltiplo com Detecção de Portadora** (CSMA – Carrier Sense Multiple Access) exige que cada estação ouça o meio (ou verifique o estado do meio) antes de enviar quadros. Em outras palavras, o CSMA é baseado no princípio de "verifique antes de transmitir" ou "ouça antes de falar".

O CSMA pode reduzir a possibilidade de colisão, mas não é capaz de eliminá-la. A razão para isso é mostrada na Figura 5.34, que mostra um modelo espaço-temporal de uma rede CSMA. As estações estão conectadas por meio de um canal compartilhado (geralmente um meio dedicado).

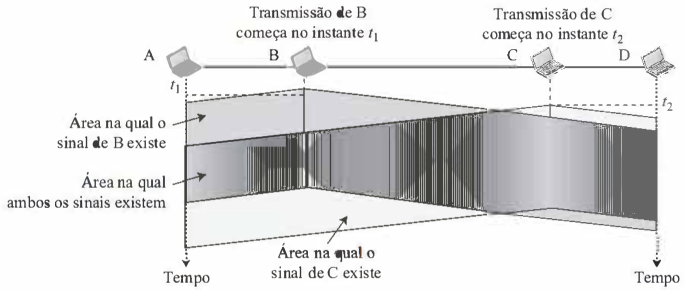


Figura 5.34 Modelo de espaço-temporal de uma colisão no CSMA.

A possibilidade de colisão ainda existe por causa do atraso de propagação; quando uma estação envia um quadro, ainda leva algum tempo (embora bastante curto) para que o primeiro *bit* chegue a cada estação e para que cada estação o perceba. Em outras palavras, uma estação pode verificar o meio e encontrá-lo livre apenas porque o primeiro *bit* enviado por outra estação ainda não foi recebido.

No instante t_1, a estação B verifica o meio e o encontra livre, e por isso envia um quadro. No instante t_2 ($t_2 > t_1$), a estação C verifica o meio e o encontra livre porque, nesse momento, os primeiros *bits* da estação B ainda não chegaram à estação C. A estação C também envia um quadro. Os dois sinais colidem e ambos os quadros são destruídos.

Período vulnerável

O período vulnerável do CSMA equivale ao tempo de propagação T_p. Esse é o intervalo necessário para que um sinal se propague de uma extremidade do meio até a outra. Quando uma estação envia um quadro e qualquer outra estação tenta enviar um quadro durante esse período, uma colisão ocorrerá. Porém, se o primeiro *bit* do quadro atingir a extremidade do meio, todas as estações já terão ouvido o *bit* e não enviarão dados. A Figura 5.35 mostra o pior caso. A estação mais à esquerda, A, envia um quadro no instante t_1, que alcança a estação D, mais à direita, no instante $t_1 + T_p$. A área cinzenta mostra a área vulnerável no tempo e espaço.

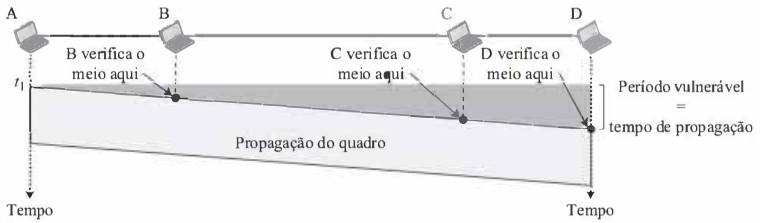


Figura 5.35 Período vulnerável no CSMA.

Métodos de persistência

O que uma estação deve fazer se o canal estiver ocupado? O que uma estação deve fazer se o canal estiver ocioso? Três métodos foram desenvolvidos para responder a essas perguntas: o **método 1-persistente**, o **método não persistente** e o **método *p*-persistente**. A Figura 5.36 mostra o comportamento desses três métodos de persistência quando uma estação encontra um canal ocupado.

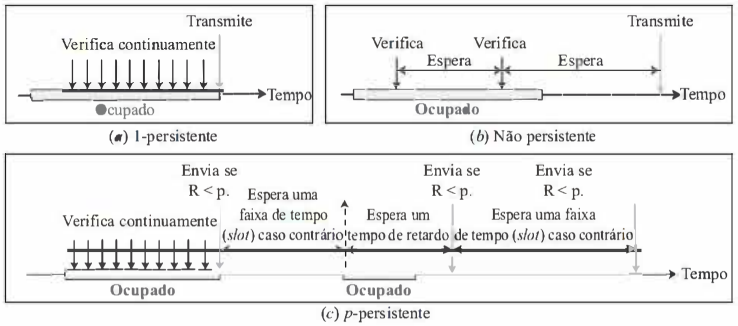


Figura 5.36 Comportamento de três métodos de persistência.

A Figura 5.37 mostra os diagramas de fluxo para esses métodos.

1-persistente O *método 1-persistente* é simples e direto. Nele, após a estação perceber que a linha está ociosa, ela envia o seu quadro imediatamente (com probabilidade 1). Esse método tem a maior chance de colisão porque duas ou mais estações podem perceber a linha ociosa e enviar seus quadros imediatamente. Veremos mais adiante que a Ethernet usa esse método.

Não persistente No *método não persistente*, uma estação que tem um quadro para enviar verifica a linha. Se ela estiver ociosa, a estação envia o quadro imediatamente. Caso não esteja, a estação espera um período aleatório de tempo e então verifica a linha novamente. A abordagem não persistente reduz a possibilidade de colisão, pois é improvável que duas ou mais estações aguardem a mesma quantidade de tempo e tentem o reenvio simultaneamente. No entanto, esse método reduz a eficiência da rede porque o meio permanece ocioso enquanto pode haver estações com quadros para enviar.

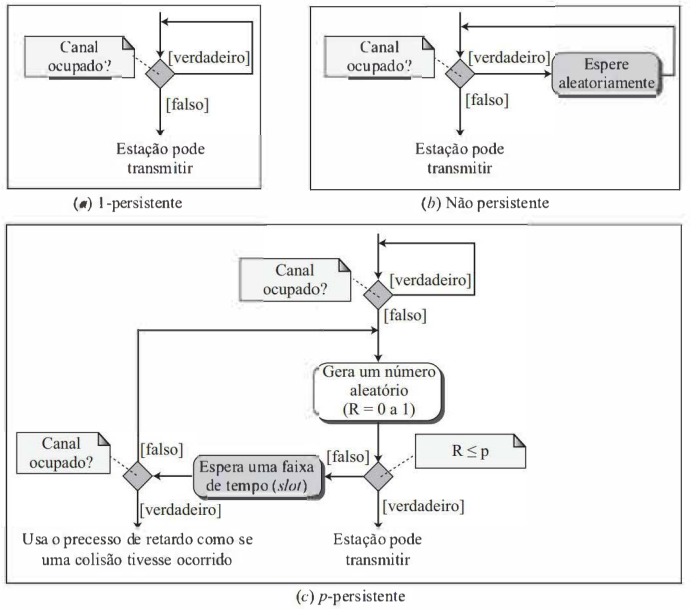


Figura 5.37 Diagrama de fluxo para três métodos de persistência.

***p*-persistente** O método p-*persistente* é utilizado se o canal empregar faixas de tempo (*slots*) com uma duração maior ou igual ao intervalo máximo de propagação. A abordagem *p*-persistente combina as vantagens das duas outras estratégias. Ela reduz a probabilidade de colisão e melhora a eficiência da rede. Nesse método, depois que a estação percebe a linha ociosa, ela segue os seguintes passos:

1. Com probabilidade *p*, a estação envia seu quadro.
2. Com probabilidade $q = 1 - p$, a estação aguarda o início da próxima faixa de tempo e verifica a linha novamente.
   a. Se a linha (meio) estiver ociosa, a estação volta ao passo 1.
   b. Se a linha (meio) estiver ocupada, a estação age como se uma colisão tivesse ocorrido e usa o procedimento de retardo.

CSMA/CD

O método CSMA não especifica o procedimento a ser seguido logo após uma colisão. O **Acesso Múltiplo com Detecção de Portadora/Detecção de Colisão** (CSMA/CD – Carrier Sense Multiple Access/Collision Detection) expande o algoritmo para tratar colisões.

Nesse método, uma estação monitora o meio após enviar um quadro para verificar se a transmissão foi bem-sucedida. Se assim for, o trabalho da estação está terminado. Se, no entanto, houver uma colisão, o quadro é enviado novamente.

Para entender melhor o CSMA/CD, observaremos os primeiros *bits* transmitidos pelas duas estações envolvidas na colisão. Embora cada estação continue a enviar os *bits* do quadro até detectar a colisão, mostraremos o que acontece quando os primeiros *bits* colidem. Na Figura 5.38, as estações A e C estão envolvidas na colisão.

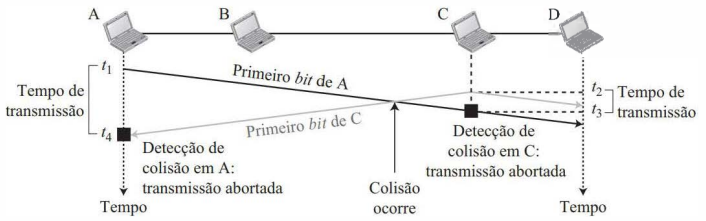


Figura 5.38 Colisão dos primeiros *bits* no CSMA/CD.

No instante t_1, a estação A executa o seu procedimento de persistência e começa a enviar os *bits* do seu quadro. No instante t_2, a estação C ainda não detectou o primeiro *bit* enviado pela estação A. A estação C executa o seu procedimento de persistência e começa a enviar os *bits* de seu quadro, que se propagam tanto para a esquerda como para a direita. A colisão ocorre algum tempo após o instante t_2. A estação C detecta a colisão no instante t_3, quando recebe o primeiro *bit* do quadro de A. A estação C imediatamente (ou após um curto período de tempo, mas aqui consideramos imediatamente) interrompe a transmissão. A estação A detecta colisão no instante t_4, quando recebe o primeiro *bit* do quadro de C; ela também interrompe a transmissão imediatamente. Observando a figura, vemos que A transmite por um intervalo de tempo igual a $t_4 - t_1$, enquanto a transmissão de C dura $t_3 - t_2$.

Agora que conhecemos a duração das duas transmissões, podemos mostrar um gráfico mais completo na Figura 5.39.

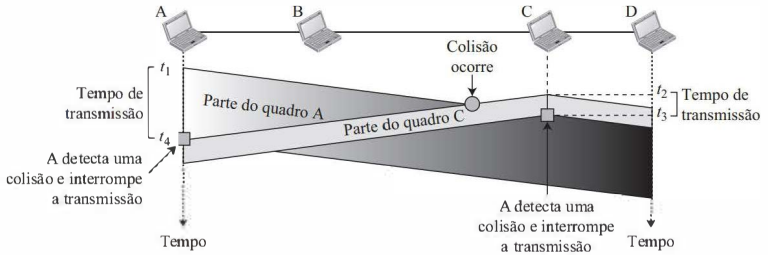


Figura 5.39 Colisão e interrupção da transmissão no CSMA/CD.

Tamanho mínimo dos quadros

Para que o CSMA/CD funcione, precisamos impor uma restrição ao tamanho dos quadros. Antes de enviar o último *bit* do quadro, a estação emissora deve detectar uma colisão, caso ela ocorra, e abortar a transmissão. Isto acontece porque a estação, após todo o quadro ter sido enviado, não mantém uma

cópia dele e não monitora a linha para detectar colisões. Portanto, o intervalo de transmissão de quadros T_{tr} deve ser pelo menos duas vezes o valor máximo do tempo de propagação T_p. Para entender essa razão, consideremos o cenário de pior caso. Se as duas estações envolvidas em uma colisão estiverem separadas pela distância máxima da rede, o sinal da primeira leva um tempo T_p para alcançar a segunda, e o efeito da colisão toma outro intervalo de tempo T_p para alcançar a primeira. Desse modo, exige-se que a primeira estação ainda esteja transmitindo após um intervalo de tempo de $2T_p$.

Exemplo 5.15

Uma rede usando CSMA/CD apresenta uma largura de banda de 10 Mbps. Se o tempo máximo de propagação (incluindo os atrasos observados nos dispositivos e ignorando o tempo necessário para enviar um sinal de interferência, conforme veremos mais adiante) é de 25,6 µs, qual é o tamanho mínimo do quadro?

Solução

O tempo mínimo de transmissão de quadros é $T_{tr} = 2 \times T_p = 51{,}2$ µs. Isto significa que, no pior caso, uma estação precisa transmitir por um período de 51,2 µs para detectar a colisão. O tamanho mínimo do quadro é de 10 Mbps × 51,2 µs = 512 *bits* ou 64 *bytes*. É realmente o tamanho mínimo do quadro para a Ethernet Padrão, conforme veremos mais adiante.

Procedimento

Agora, observemos o diagrama de fluxo do CSMA/CD na Figura 5.40. Ele é semelhante ao usado para o protocolo ALOHA, mas existem diferenças.

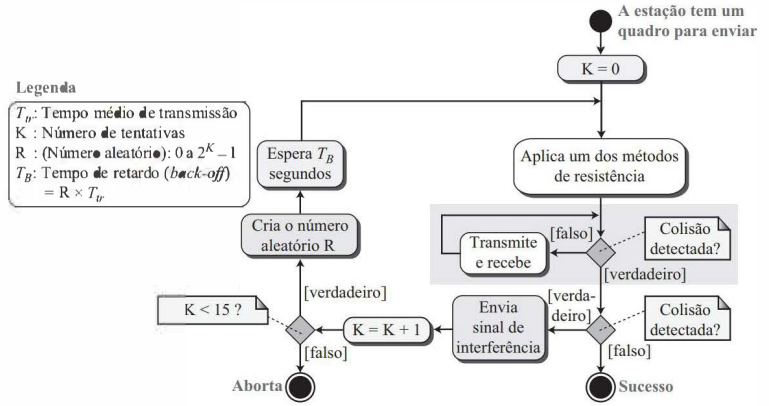


Figura 5.40 Diagrama de fluxo para o CSMA/CD.

A primeira diferença é a inserção do processo de persistência. Precisamos verificar o canal antes de começar a enviar o quadro usando um dos processos de persistência discutidos anteriormente (não persistente, 1-persistente ou *p*-persistente). A caixa correspondente pode ser substituída por um dos processos de persistência mostrados na Figura 5.37.

A segunda diferença refere-se à transmissão de quadros. No ALOHA, primeiro transmitimos o quadro inteiro e depois esperamos por uma confirmação. No CSMA/CD, a transmissão e a detecção de colisão fazem parte de um processo contínuo. Não enviamos o quadro inteiro e

depois verificamos se houve uma colisão. A estação transmite e recebe contínua e simultaneamente (usando duas portas diferentes ou uma porta bidirecional). Usamos um laço para mostrar que a transmissão é um processo contínuo. Monitoramos constantemente a fim de detectar uma dentre duas condições: ou a transmissão foi concluída ou uma colisão foi detectada. Qualquer um desses eventos interrompe a transmissão. Quando saímos do laço, se uma colisão não tiver sido detectada, significa que a transmissão foi concluída com sucesso, ou seja, o quadro todo foi transmitido. Caso contrário, ocorreu uma colisão.

A terceira diferença é o envio de um curto *sinal de interferência* (*jamming*) para se certificar de que todas as outras estações tomaram ciência da colisão.

Nível de energia

Podemos dizer que o nível de energia em um canal pode assumir três valores: zero, normal e anormal. No nível zero, o canal está ocioso. No nível normal, uma estação capturou com sucesso o canal e está enviando seu quadro. No nível anormal, há uma colisão e o nível da energia é duas vezes o nível normal. Uma estação que tem um quadro para enviar ou está enviando um quadro precisa monitorar o nível de energia para determinar se o canal está ocioso, ocupado ou no modo de colisão. A Figura 5.41 mostra essa situação.

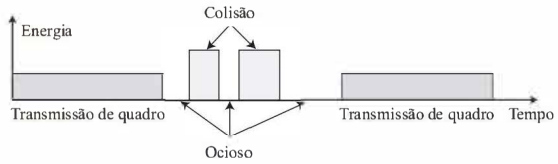


Figura 5.41 Nível de energia durante a transmissão, ociosidade ou colisão.

Vazão

A vazão do CSMA/CD é superior à do ALOHA puro e do *slotted* ALOHA. A vazão máxima é obtida com um valor diferente de G e depende do método de persistência e do valor de p na abordagem p-persistente. Para o método 1-persistente, a taxa de transferência máxima é de cerca de 50% quando $G = 1$. Para o método não persistente, a vazão máxima pode chegar a 90% quando G estiver entre 3 e 8.

Ethernet tradicional

Um dos protocolos de LAN que utilizava o CSMA/CD é a Ethernet tradicional com uma taxa de transferência de dados de 10 Mbps. Discutiremos as LANs Ethernet no final do capítulo, mas é bom saber que a Ethernet tradicional era uma LAN de *broadcast* que utilizava o método 1-persistente para o controle de acesso ao meio de comunicação compartilhado. As versões posteriores da Ethernet passaram a usar métodos de acesso diferentes do CSMA/CD pelas razões que discutiremos na próxima seção, na qual tratamos de LANs com fios.

CSMA/CA

Uma variante do método CSMA é o **Acesso Múltiplo com Detecção de Portadora/Prevenção de Colisão** (CSMA/CA – Carrier Sense Multiple Access/Collision Avoidance), que é usado em LANs sem fios. Adiaremos a discussão sobre esse método para o Capítulo 6.

5.3.2 Acesso controlado

No **acesso controlado**, as estações consultam umas às outras para determinar qual estação tem o direito de enviar quadros. A estação não pode enviar a menos que ela tenha sido autorizada por outras estações. Discutimos três métodos de acesso controlado.

Reserva

No método de **reserva**, uma estação precisa fazer uma reserva antes de enviar dados. O tempo é dividido em intervalos discretos; em cada um, um quadro de reserva precede os quadros de dados enviados naquele intervalo.

Se existem N estações no sistema, há exatamente N mini-intervalos de reserva em um quadro de reserva. Cada mini-intervalo pertence a uma estação. Quando uma estação precisa enviar um quadro de dados, ela faz uma reserva no seu próprio mini-intervalo. As estações que fizeram reservas podem enviar seus quadros de dados após o quadro de reserva.

A Figura 5.42 mostra uma situação com cinco estações e um quadro de reserva contendo cinco mini-intervalos. No primeiro intervalo, apenas as estações 1, 3 e 4 fizeram reservas. No segundo, apenas uma estação fez uma reserva.

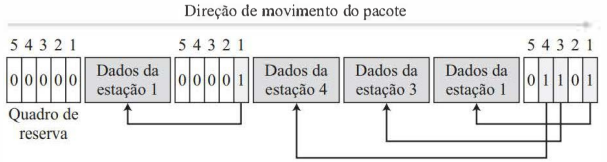


Figura 5.42 Método de acesso por reserva.

Varredura

O método de **varredura** (também conhecido como *polling*) funciona em topologias nas quais um dispositivo é designado como *estação primária* e os outros dispositivos são designados *estações secundárias*. Todas as trocas de dados devem ser feitas por meio do dispositivo primário, mesmo quando o destino final for um dispositivo secundário. O dispositivo primário controla o enlace; os secundários seguem as suas instruções. É tarefa do dispositivo primário determinar qual dispositivo está autorizado a utilizar o canal em um dado instante. O dispositivo primário, portanto, é sempre o iniciador de uma sessão (ver Figura 5.43). Esse método usa funções de varredura e seleção para evitar colisões. No entanto, a desvantagem é que, se a estação primária falhar, o sistema todo falha.

Seleção

A função de seleção é usada sempre que o dispositivo primário tem algo para enviar. Lembre-se de que a estação primária controla o enlace. Se ela não estiver enviando nem recebendo dados, ela sabe que o enlace está disponível. Se a estação primária tiver algo para enviar, ela envia. O que ela não sabe, no entanto, é se o dispositivo de destino está preparado para receber. Assim, a estação primária deve alertar a secundária sobre a transmissão que virá e deve aguardar uma confirmação (ACK) de que a estação secundária está pronta para receber dados. Antes de enviar os dados, a estação primária cria e transmite um quadro de seleção (SEL), que tem um campo no qual está o endereço da estação secundária de destino.

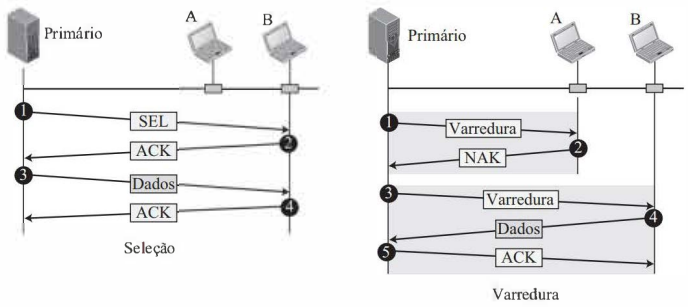


Figura 5.43 Funções de seleção e varredura no método de acesso por varredura.

Varredura

A função de varredura (*pool*) é utilizada pelo dispositivo primário para solicitar transmissões dos dispositivos secundários. Quando a estação primária está pronta para receber dados, ela deve perguntar a cada dispositivo, um de cada vez, se ele tem algo para enviar. Quando o primeiro dispositivo secundário é abordado, ele responde com um quadro de NAK caso não tenha algo para enviar, ou com dados (na forma de um quadro de dados) caso contrário. Se a resposta for negativa (um quadro NAK), a estação primária pergunta à próxima estação secundária como anteriormente, até encontrar uma estação com dados para enviar. Quando a resposta for positiva (um quadro de dados), a estação primária lê o quadro e retorna uma confirmação (quadro ACK), verificando sua recepção.

Passagem de ficha

No método de **passagem de ficha**, as estações da rede são organizadas em um anel lógico. Em outras palavras, para cada estação existe um *predecessor* e um *sucessor*. O predecessor é a estação logicamente anterior à estação em questão no anel, enquanto o sucessor é a estação seguinte à estação no anel. A estação atual é aquela que está acessando o canal no momento. O direito de realizar esse acesso foi passado pelo antecessor da estação atual. Esse direito será transferido ao sucessor da estação atual quando ela não tiver mais dados para enviar.

Mas como o direito de acessar o canal é passado de uma estação para outra? Nesse método, um pacote especial denominado *ficha* (*token*) circula através do anel. A posse da ficha dá à estação o direito de acessar o canal e enviar seus dados. Quando uma estação tem alguns dados para enviar, ela espera até receber a ficha de seu antecessor. Em seguida, detém a ficha e envia seus dados. Quando a estação não tiver mais dados para enviar, ela libera a ficha, passando-a para a próxima estação no anel lógico. A estação não pode enviar dados até receber a ficha novamente na próxima rodada. Nesse processo, quando uma estação recebe a ficha e não tem dados para enviar, ela simplesmente passa a ficha para a estação seguinte.

O gerenciamento de fichas é necessário nesse método de acesso. Deve haver um limite de tempo durante o qual as estações podem manter a posse da ficha, que deve ser monitorada para garantir que não seja perdida ou destruída. Por exemplo, se a estação que estiver de posse da ficha falhar, a ficha desaparecerá da rede. Outra função do gerenciamento de fichas é atribuir prioridades às estações e aos tipos de dados a serem transmitidos. Finalmente, o gerenciamento de fichas é necessário para garantir que estações com baixa prioridade liberarem a ficha para estações de prioridade mais alta.

Anel lógico

Em uma rede que usa o método de passagem de ficha, as estações não precisam estar fisicamente conectadas em um anel; ele pode ser lógico. A Figura 5.44 mostra quatro topologias físicas diferentes que podem criar um anel lógico.

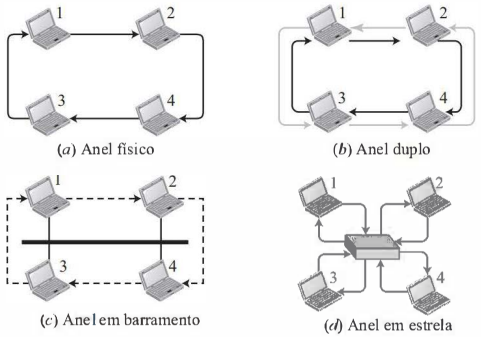


Figura 5.44 Anel lógico e topologia física no método de acesso por passagem de ficha.

Na topologia em anel físico, quando uma estação envia a ficha para o seu sucessor, ela não pode ser vista pelas outras estações; o sucessor é a próxima estação na linha. Isto significa que a ficha não precisa carregar o endereço do próximo sucessor. O problema com essa topologia é que, se um dos elos – o meio entre duas estações adjacentes – falhar, o sistema todo falha.

A topologia em anel duplo utiliza um segundo anel (auxiliar) que opera na direção inversa à do anel principal. O segundo anel é usado apenas para emergências (análogo a um pneu sobressalente em um carro). Se um dos elos do anel principal falhar, o sistema automaticamente combina os dois anéis para formar um anel temporário. Após o elo falho ser restabelecido, o anel auxiliar volta a ficar ocioso. Perceba que, para essa topologia funcionar, cada estação precisa ter duas portas de transmissão e duas portas de recepção. As redes Token Ring de alta velocidade conhecidas como Interface de Dados Distribuídos por Fibra (FDDI – Fiber Distributed Data Interface) e Interface de Dados Distribuídos por Cobre (CDDI – Copper Distributed Data Interface) adotam essa topologia.

Na topologia de anel em barramento (*bus ring*), também chamada de ficha em barramento (*bus token*), as estações estão conectadas a um único cabo denominado barramento. Elas formam, entretanto, um anel lógico, pois cada estação conhece o endereço de seu sucessor (e também do predecessor para fins de gerenciamento de fichas). Quando uma estação termina de enviar seus dados, ela libera a ficha e insere o endereço de seu sucessor na ficha. Apenas a estação cujo endereço corresponde ao endereço de destino da ficha fica com a ficha de acesso ao meio compartilhado. A LAN conhecida como Token Bus, padronizada pelo IEEE, adota esta topologia.

Na topologia de anel em estrela, a topologia física é uma estrela. Existe um *hub*, no entanto, que atua como conector da rede. O cabeamento dentro do *hub* é o que cria o anel; as estações são conectadas a ele por meio de duas conexões com fios. Essa topologia torna a rede menos propensa a falhas, porque se um elo falhar, ele pode ser ignorado pelo *hub* e as estações restantes podem continuar a operar normalmente. A adição e remoção de estações no anel também são facilitadas. Essa topologia ainda é usada na LAN Token Ring concebida pela IBM.

5.3.3 Canalização

A **canalização** (ou *divisão de canal*, como às vezes é denominada) é um método de acesso múltiplo em que a largura de banda disponível em um enlace é compartilhada no tempo, na frequência ou por meio de códigos, entre diferentes estações. Como tais métodos são normalmente usados em redes sem fios, deixamos a discussão sobre eles para o próximo capítulo.

5.4 ENDEREÇAMENTO NA CAMADA DE ENLACE

A próxima questão que precisamos discutir com relação à camada de enlace de dados são os endereços da camada de enlace. No Capítulo 4, discutimos endereços IP como os identificadores na camada de rede que definem os pontos exatos na Internet aos quais as estações de origem e de destino estão conectadas. No entanto, em uma internet não orientada à conexão como a Internet, não podemos fazer um datagrama chegar ao seu destino utilizando apenas endereços IP. A razão é que cada datagrama na Internet, mesmo que venha de uma mesma estação de origem e vá para uma mesma estação de destino, pode tomar um caminho diferente entre elas. Os endereços IP de origem e de destino definem as duas extremidades da comunicação, mas não podem definir os enlaces pelos quais o datagrama deve passar.

É preciso ter em mente que os endereços IP de um datagrama não devem ser modificados. Se o endereço IP de destino em um datagrama for alterado, o pacote nunca chegará ao seu destino; se o de origem for alterado, a estação de destino e os roteadores não serão capazes de se comunicar com a origem caso uma resposta precise ser retornada ou caso seja necessário lhes reportar um erro (ver a discussão sobre ICMP no Capítulo 4).

A discussão anterior mostra que precisamos de outro mecanismo de endereçamento em internets não orientadas à conexão: os endereços da camada de enlace dos dois nós. Um *endereço da camada de enlace* é também conhecido como *endereço do enlace*, *endereço físico* ou *endereço MAC*. Neste livro, usamos esses termos como sinônimos.

Como um enlace é controlado na camada de enlace de dados, os endereços precisam pertencer à camada de enlace de dados. Quando um datagrama passa da camada de rede para a camada de enlace de dados, o datagrama é encapsulado em um quadro e dois endereços da camada de enlace de dados são adicionados ao cabeçalho do quadro. Eles são alterados cada vez que o quadro passa de um enlace para o outro. A Figura 5.45 ilustra esse conceito em uma internet de pequeno porte.

Na internet da Figura 5.45, temos três enlaces e dois roteadores. Mostramos também apenas duas estações: Alice (origem) e Bob (destino). Para cada estação, mostramos dois endereços, os endereços IP (I) e os endereços da camada de enlace (E). Observe que um roteador tem tantos pares de endereços quanto o número de enlaces aos quais o roteador está conectado. Mostramos três quadros, um em cada enlace. Cada quadro carrega os mesmos endereços IP de origem e destino (I_1 e I_8), mas os endereços da camada de enlace no quadro mudam de enlace para enlace. No enlace 1, os endereços da camada de enlace são E_1 e E_2. No enlace 2, eles são E_4 e E_5. No 3, eles são E_7 e E_8. Perceba que os endereços IP e os endereços da camada de enlace não estão na mesma ordem. Para endereços IP, o endereço de origem vem antes do endereço de destino; para endereços da camada de enlace, o endereço de destino vem antes do de origem. Os datagramas e os quadros foram projetados assim e seguimos esse padrão. Podemos levantar várias questões:

- Se o endereço IP de um roteador não aparece em qualquer datagrama enviado de uma origem para um destino, por que precisamos atribuir endereços IP para os roteadores? A resposta é que, em alguns protocolos, um roteador pode atuar como o emissor ou receptor de um datagrama. Por exemplo, nos protocolos de roteamento que discutimos no Capítulo 4, um roteador é um emissor ou um receptor de uma mensagem. As comunicações nesses protocolos se dão entre os roteadores.

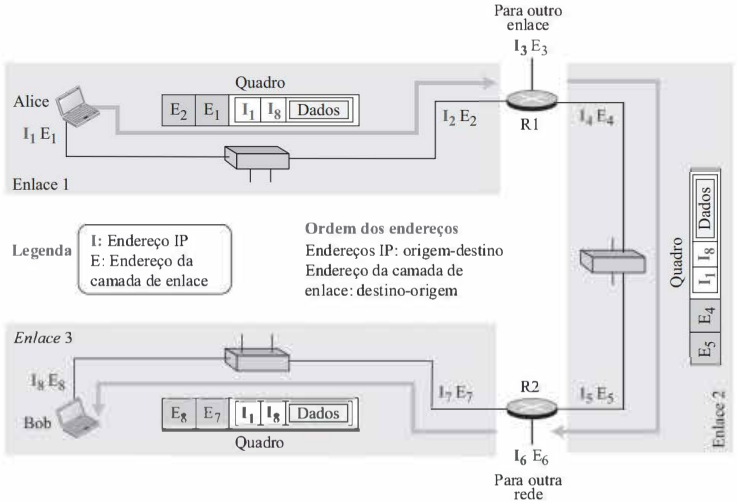


Figura 5.45 Endereços IP e da camada de enlace em uma pequena internet.

- Por que precisamos de mais de um endereço IP em um roteador, um para cada interface? A resposta é que uma interface é uma conexão de um roteador em um enlace. Dissemos que um endereço IP define um ponto na Internet ao qual um dispositivo está conectado. Um roteador com *n* interfaces está conectado à Internet em *n* pontos. É o caso de uma casa que fique na esquina de uma rua e que tenha duas portas; cada porta tem o endereço relativo à rua correspondente.
- Como os endereços IP de origem e destino de um pacote são determinados? A resposta é que a estação deve saber o seu próprio endereço IP, que é usado como endereço IP de origem no pacote. Conforme discutimos no Capítulo 2, a camada de aplicação utiliza os serviços do DNS para localizar o endereço de destino do pacote e o passa para a camada de rede para que o mesmo seja inserido no pacote.
- Como os endereços da camada de enlace de origem e de destino são determinados para cada enlace? Mais uma vez, cada salto (roteador ou estação) deve saber o seu próprio endereço da camada de enlace, conforme discutiremos mais adiante neste capítulo. O endereço de destino da camada de enlace é determinado por meio do Protocolo de Resolução de Endereços (ARP – Address Resolution Protocol), que abordaremos em breve.
- Qual é o tamanho dos endereços da camada de enlace? A resposta é que isto depende do protocolo utilizado pelo enlace. Embora tenhamos apenas o protocolo IP para a Internet toda, podemos usar diferentes protocolos da camada de enlace de dados em enlaces distintos. Isto significa que podemos definir o tamanho do endereço quando discutimos os diferentes protocolos da camada de enlace, o que será feito para redes com fios neste capítulo e para redes sem fio no próximo.

Exemplo 5.16

Como discutiremos mais adiante neste capítulo, os endereços da camada de enlace no tipo mais comum de LAN, a Ethernet, apresentam 48 *bits* (seis *bytes*) que são representados como 12 dígitos hexadecimais separados por dois pontos; por exemplo, o seguinte é um endereço da camada de enlace de um computador:

A2:34:45:11:92:F1

Protocolo de Resolução de Endereços

Sempre que um nó tem um datagrama IP para enviar para outro nó em um enlace, ele tem o endereço IP do nó de destino. A estação de origem sabe o endereço IP do roteador-padrão. Cada roteador, exceto pelo último no caminho, obtém o endereço IP do próximo roteador usando sua tabela de roteamento (na coluna próximo salto). O último roteador conhece o endereço IP da estação de destino. No entanto, o endereço IP do próximo nó não é útil para movimentar um quadro por meio de um enlace; precisamos do endereço da camada de enlace do nó seguinte. É nesse momento que o **Protocolo de Resolução de Endereços** (ARP – Address Resolution Protocol) mostra-se útil. O protocolo ARP é um dos protocolos auxiliares definidos na camada de rede, conforme mostrado na Figura 5.46. Ele pertence à camada de rede, mas adiamos a sua discussão até agora porque ele mapeia um endereço IP para um endereço de enlace lógico. O ARP recebe um endereço IP vindo do protocolo IP, mapeia o endereço para o endereço correspondente da camada de enlace, e passa o resultado para a camada de enlace de dados.

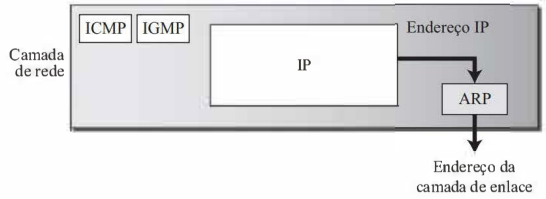


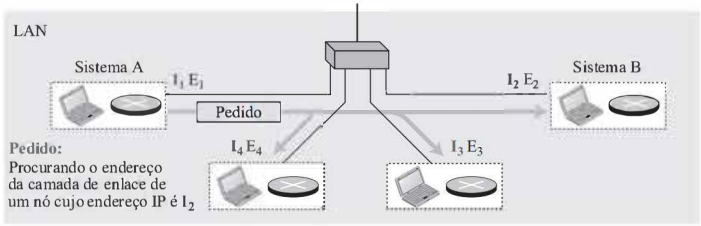
Figura 5.46 Posição do ARP na pilha de protocolos TCP/IP

Sempre que uma estação ou um roteador precisa encontrar o endereço da camada de enlace de outra estação ou outro roteador em sua rede, ele envia um pacote de pedido ARP. O pacote inclui os endereços da camada de enlace e IP do remetente e o endereço IP do destinatário. Como o remetente não sabe o endereço da camada de enlace do destinatário, a consulta é transmitida através do enlace com o endereço de *broadcast* da camada de enlace, que discutiremos mais adiante para cada protocolo (ver Figura 5.47).

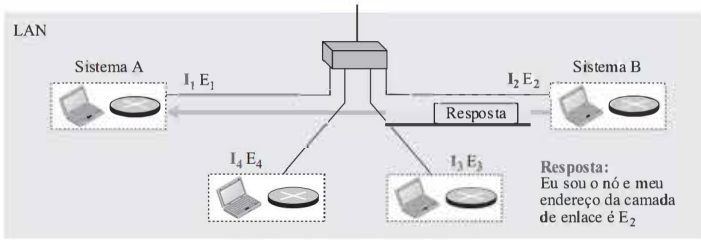
Todas as estações e roteadores na rede recebem e processam o pacote de pedido ARP, mas somente o destinatário correto reconhece seu endereço IP e envia de volta um pacote de resposta ARP. O pacote de resposta contém os endereços IP e da camada de enlace do destinatário. O pacote é enviado via *unicast*, diretamente para o nó que enviou o pacote de pedido.

Na Figura 5.47*a*, o sistema do lado esquerdo (A) tem um pacote que precisa ser entregue a outro sistema (B) cujo endereço IP é I_2. O sistema A precisa entregar o pacote para sua camada de enlace de dados para a entrega de fato, mas ele não sabe o endereço físico do destinatário. Ele usa os serviços do ARP, pedindo ao protocolo que envie um pacote de pedido ARP via *broadcast*, no qual seja solicitado o endereço físico de um sistema cujo endereço IP é I_2.

Esse pacote é recebido por todos os sistemas na rede física, mas apenas o sistema B responderá, conforme mostra a Figura 5.47*b*. O sistema B envia um pacote de resposta ARP que inclui seu endereço físico. Agora, o sistema A pode enviar todos os pacotes que ele tem para esse destino usando o endereço físico recebido.



(*a*) Pedido ARP é enviado via *broadcast*



(*b*) Resposta ARP é enviada via *unicast*

Figura 5.47 Operação do ARP.

Formato dos pacotes

A Figura 5.48 mostra o formato de um pacote ARP. Os nomes dos campos são autoexplicativos. O campo *tipo de hardware* define o tipo do protocolo da camada de enlace; para a Ethernet,

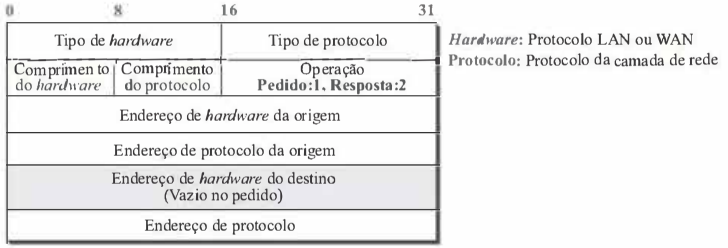


Figura 5.48 Pacote ARP.

o valor desse campo é 1. O campo *tipo de protocolo* define o protocolo da camada de rede: o valor para o protocolo IPv4 é $(0800)_{16}$. Os campos de *endereço de hardware da origem* e *endereço de protocolo da origem* têm tamanho variável e definem os endereços da camada de enlace e da camada de rede do remetente. Os campos de *endereço de hardware do destino* e *endereço de protocolo do destino* definem os endereços da camada de enlace e da camada de rede do destinatário. Um pacote ARP é encapsulado diretamente em um quadro da camada de enlace de dados. O quadro precisa ter um campo para mostrar que a carga útil corresponde ao ARP e não a um datagrama da camada de rede.

Exemplo 5.17

Uma estação cujo endereço IP é I_1 e cujo endereço MAC é E_1 deseja enviar um pacote para outra estação, cujo endereço IP é I_2 e cujo endereço físico é E_2 (o qual não é conhecido pela primeira estação). As duas estações estão na mesma rede. Descreva o pedido ARP e os pacotes de resposta encapsulados em quadros Ethernet (ver Figura 5.55).

Solução

A Figura 5.49 mostra o pedido ARP e os pacotes de resposta. Observe que o campo de dados do ARP nesse caso é de 28 *bytes*, e que os endereços individuais não se encaixam nos limites de 4 *bytes*. Por isso não mostramos os limites usuais de 4 *bytes* para esses endereços. Perceba também que os endereços IP são mostrados em hexadecimal.

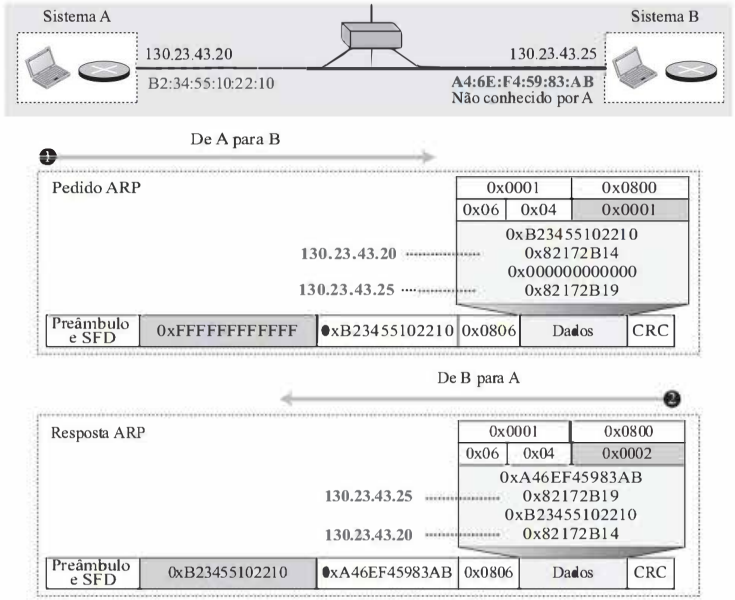


Figura 5.49 Esquema para o Exemplo 5.17.

Um exemplo

Discutimos endereçamento nas quatro camadas mais altas da pilha de protocolos TCP/IP. Não existe endereçamento na camada física, pois a unidade de comunicação nela é um *bit*, o qual não pode carregar um endereço; na camada física, os *bits* são enviados e recebidos por qualquer receptor conectado ao remetente (*broadcast*). Consideramos que agora é a hora de mostrar todos esses endereços em uma única transação, enfatizando o relacionamento entre eles. Também mostramos como todos eles podem de fato ser criados pelo sistema. O que é necessário para a transação é apenas o nome do local que um usuário deseja contatar. A Figura 5.50 mostra uma internet à qual Alice e Bob estão conectados. Alice é uma usuária que quer acessar a lista dos produtos oferecidos por Bob, que tem uma pequena empresa chamada ***Bob.biz***, na qual a lista dos produtos contendo suas especificações é armazenada em um documento denominado ***produtos***.

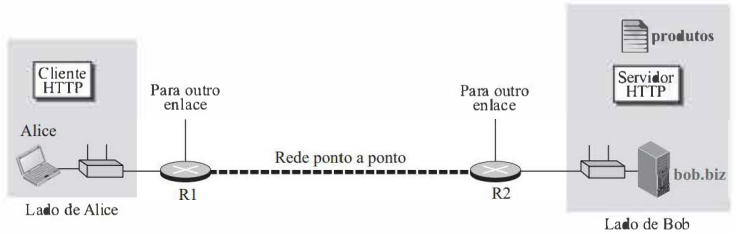


Figura 5.50 A internet para o nosso exemplo.

O único identificador que Alice precisa saber é o URL do *site* de Bob: ***http://bob.biz/product***. O restante dos endereços, de origem e de destino, é gerado pelo sistema, conforme mostra o exemplo.

Operações no lado de Alice

Consideramos que Alice está conectada a uma LAN e que seu computador conhece seu próprio endereço IP e endereço da camada de enlace. O servidor de Bob também está conectado a uma LAN e seu computador conhece seu próprio endereço IP e endereço da camada de enlace. Os roteadores na internet também conhecem seus próprios endereços IP e da camada de enlace. Utilizaremos endereços simbólicos para tornar as figuras mais legíveis. No entanto, Alice não precisa saber qualquer um desses endereços: eles são criados automaticamente à medida que a transação é executada.

A Figura 5.51 mostra o que acontece no lado de Alice. Alice, usando um navegador, digita o URL do *site* do Bob (http://bob.biz/product). O computador de Alice usa o cliente HTTP para responder à solicitação de Alice. O cliente HTTP, no entanto, não sabe o endereço IP de Bob. Esse cliente invoca o cliente DNS para obter ajuda. O cliente DNS envia uma solicitação a um servidor DNS e descobre o endereço IP do computador de Bob ($\mathbf{I_B}$), que é passado para a camada de transporte.

A camada de transporte cria, então, um segmento utilizando o número de porta bem conhecido do servidor HTTP ($\mathbf{P_B} = 80$) e um número de porta temporário recebido do sistema operacional para o cliente HTTP ($\mathbf{P_A}$). A camada de transporte pode agora criar um segmento e passá-la para a camada de rede com o endereço IP do computador de Bob ($\mathbf{I_B}$), obtido no passo anterior. Obviamente, não mostramos todos os processos que ocorrem no estabelecimento da conexão; consideramos que a conexão é feita aqui e que apenas um segmento é trocado.

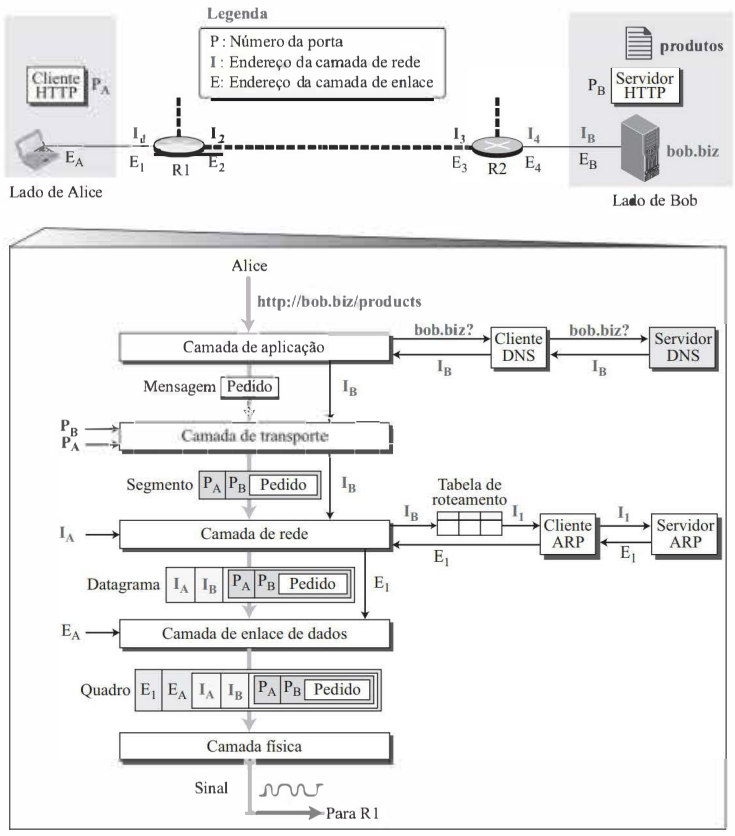


Figura 5.51 Fluxo de pacotes no computador de Alice.

A camada de rede recebe o segmento e I_B, mas precisa descobrir o endereço da camada de enlace. A camada de rede consulta sua tabela de roteamento e tenta descobrir qual é o próximo roteador (nesse caso, o roteador-padrão) para o destino I_B. A tabela de roteamento fornece I_1, mas a camada de rede precisa determinar o endereço da camada de enlace do roteador R1. Ela usa seu cliente ARP para determinar o endereço da camada de enlace E1. A camada de rede pode agora passar o datagrama com o endereço da camada de enlace para a camada de enlace de dados.

A camada de enlace de dados conhece seu próprio endereço de camada de enlace, E_A. Ela cria o quadro e o passa para a camada física, onde os dados são convertidos para sinais (conforme veremos no Capítulo 7) e enviados pelo meio de comunicação.

Operações no roteador R1

Agora, veremos o que acontece no roteador R1. O roteador R1, como sabemos, tem apenas três camadas inferiores. O pacote recebido precisa subir essas três camadas e depois descer. A Figura 5.52 mostra essas operações. Na chegada dos dados, a camada física da conexão da esquerda cria o quadro e o passa para a camada de enlace de dados, que desencapsula o datagrama e o passa para a camada de rede. Esta examina o endereço de camada de rede do datagrama e descobre que ele deve ser entregue para o dispositivo cujo endereço IP é I_B. A camada de rede consulta sua tabela de roteamento para descobrir qual é o próximo nó (roteador) no caminho até I_B. A tabela de roteamento retorna I_3 como resultado. Este é o endereço IP do roteador R2, que está no mesmo enlace que R1. A camada de rede usa, então, o cliente e o servidor ARP para determinar o endereço da camada de enlace desse roteador, que vem a ser E_3. A camada de rede passa o datagrama e E_3 para a camada de enlace de dados pertencente à conexão do lado direito. A camada de enlace encapsula o datagrama, adiciona E_3 e E_2 (o seu próprio endereço da camada de enlace) e passa o quadro para a camada física. A camada física codifica os *bits* em sinais e os envia pelo meio de comunicação até R2.

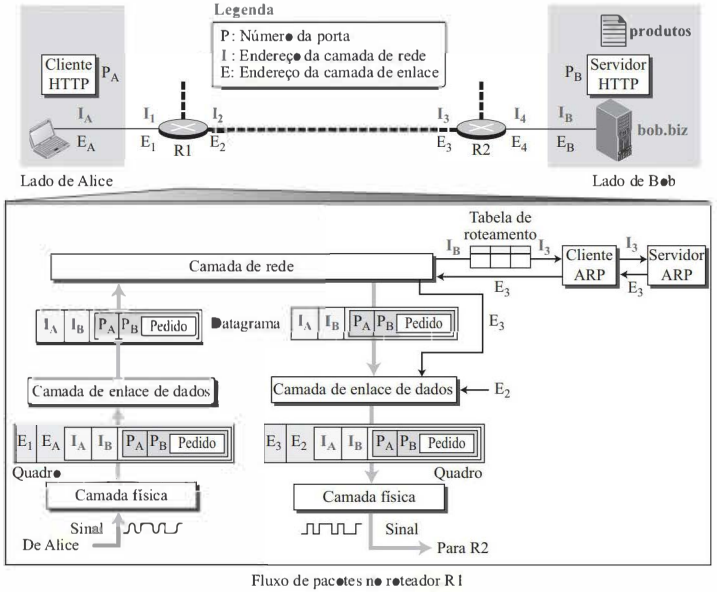


Figura 5.52 Fluxo de operações no roteador R1.

Operações no roteador R2

As operações neste roteador são as mesmas que ocorrem no roteador R1, exceto pelo fato de que alguns endereços são alterados. Por isso, não repetimos tais operações aqui.

Operações no lado de Bob

Agora, veremos o que acontece no lado de Bob. A Figura 5.53 mostra como os sinais no lado de Bob são transformados em uma mensagem para o programa-servidor HTTP. No lado de Bob, não

são mais necessários endereços ou mapeamentos. O sinal recebido do enlace é transformado em um quadro. Este é passado para a camada de enlace de dados, que desencapsula o datagrama e o passa para a camada de rede. Ela desencapsula a mensagem e a passa para a camada de transporte. A camada de transporte desencapsula o segmento e o passa para a porta P_B (80). O servidor HTTP recebe a mensagem, prepara uma cópia do arquivo de produtos, formata-o como um documento HTML, e envia o resultado para Alice usando o mesmo fluxo de comunicação, mas na ordem inversa. A mensagem, provavelmente, passará pelos mesmos roteadores, R1 e R2, até chegar a Alice.

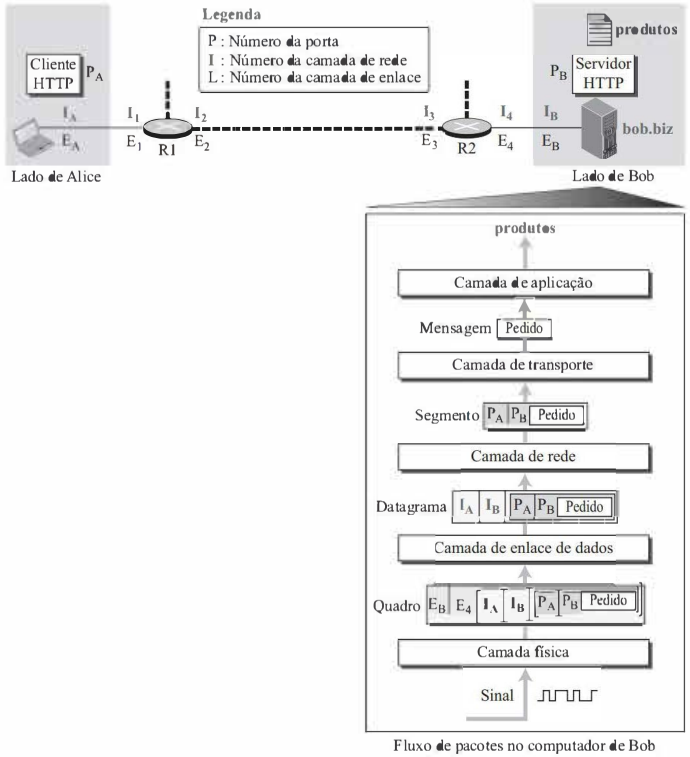


Figura 5.53 Operações no lado de Bob.

5.5 REDES COM FIOS: PROTOCOLO ETHERNET

No Capítulo 1, mencionamos que a pilha de protocolos TCP/IP não define qualquer protocolo da camada de enlace de dados nem da camada física. Em outras palavras, a pilha de protocolos TCP/IP aceita qualquer protocolo para essas duas camadas que sejam capazes de fornecer serviços para a camada de rede. A camada de enlace de dados e a camada física são realmente o território das redes

locais e de longa distância. Isto significa que, quando discutimos essas duas camadas, estamos falando sobre as redes que as estão usando. Como podemos ver neste capítulo e no próximo, podemos ter redes com ou sem fios. Discutimos redes com fios neste capítulo e deixamos a discussão sobre redes sem fios para o próximo capítulo.

No Capítulo 1, aprendemos que uma rede local (LAN) é uma rede de computadores projetada para cobrir uma área geográfica limitada, como um prédio ou um *campus*. Embora uma LAN possa ser usada como uma rede isolada para conectar computadores em uma organização com o único propósito de compartilhar recursos, a maioria das LANs hoje também está ligada a uma rede de longa distância (WAN) ou à Internet.

Nas décadas de 1980 e 1990, vários tipos diferentes de LANs eram utilizados. Todas essas LANs usavam um método de acesso ao meio para resolver o problema do compartilhamento do meio de comunicação. A Ethernet adotou a abordagem CSMA/CD. As tecnologias Token Ring, Token Bus e Interface de Dados Distribuídos por Fibra (FDDI – Fiber Distributed Data Interface) adotaram a abordagem de passagem de ficha. Durante esse período, outra tecnologia de LAN, denominada ATM e que implantou a tecnologia de WAN de alta velocidade (ATM), apareceu no mercado.

Exceto pela Ethernet, quase todas as tecnologias de LAN desapareceram do mercado porque a Ethernet foi capaz de se atualizar para atender as necessidades de cada época. Várias razões para esse sucesso são mencionadas na literatura, mas acreditamos que o protocolo Ethernet foi projetado para que ele fosse capaz de evoluir com a demanda por maiores taxas de transmissão. É natural que uma organização que usava uma LAN Ethernet no passado e agora precisava de uma taxa de dados mais elevada preferisse fazer a atualização para a nova geração em vez de migrar para outra tecnologia, algo que pode envolver maiores custos. Isto significa que limitaremos nossa discussão sobre LANs com fios a uma discussão sobre a Ethernet.

A tecnologia de LAN Ethernet foi desenvolvida em 1970 por Robert Metcalfe e David Boggs. Desde então, ela passou por quatro gerações: **Ethernet Padrão** (10 Mbps), **Fast Ethernet** (100 Mbps), **Ethernet Gigabit** (1 Gbps) e **10 Gibabit Ethernet** (10 Gbps).

5.5.1 Projeto IEEE 802

Antes de discutirmos o protocolo Ethernet e todas as suas gerações, precisamos discutir o padrão IEEE que comumente encontramos na literatura e na vida real. Em 1985, a Sociedade de Computação do IEEE iniciou um projeto, denominado **Projeto 802**, com o objetivo de estabelecer normas para permitir a intercomunicação entre equipamentos de vários fabricantes. O Projeto 802 não tem intenção de substituir qualquer parte do modelo OSI ou da pilha de protocolos TCP/IP. Na verdade, ele é uma maneira de especificar as funções da camada física e da camada de enlace de dados dos principais protocolos de LAN.

A relação do Padrão 802 com a pilha de protocolos TCP/IP é mostrada na Figura 5.54. O IEEE subdividiu a camada de enlace de dados em duas subcamadas: **Controle de Enlace**

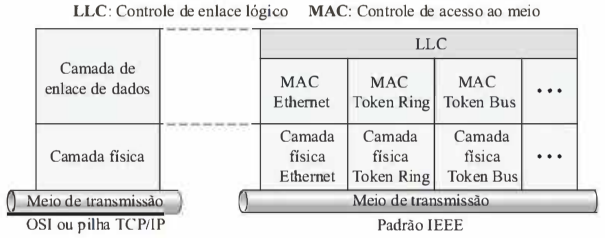


Figura 5.54 Padrões IEEE para LANs.

Lógico (LLC – Logical Link Control) e **Controle de Acesso ao Meio** (MAC – Media Access Control). O IEEE também criou vários padrões de camada física para diferentes protocolos de LAN.

Controle de Enlace Lógico

Anteriormente, discutimos sobre o *controle de enlace de dados*. Dissemos que o controle de enlace de dados lida com o enquadramento, controle de fluxo e controle de erros. No Projeto IEEE 802, as tarefas de controle de fluxo, controle de erros e parte das funções de enquadramento são agrupadas em uma subcamada denominada *Controle de Enlace Lógico* (LLC – Logical Link Control). O enquadramento é tratado tanto na subcamada LLC como na subcamada MAC.

O LLC fornece um único protocolo de controle da camada de enlace para todas as LANs IEEE. Isto significa que o protocolo LLC pode fornecer interconexão entre LANs diferentes porque ele torna a subcamada MAC transparente.

Controle de Acesso ao Meio

Anteriormente, discutimos diversos métodos de acesso, incluindo acesso aleatório, acesso controlado e canalização. O Projeto IEEE 802 criou uma subcamada denominada Controle de Acesso ao Meio (MAC) que define o método de acesso específico para cada LAN. Por exemplo, ela define o CSMA/CD como método de acesso ao meio para LANs Ethernet e define o método de passagem de ficha para LANs Token Ring e Token Bus. Conforme mencionamos na seção anterior, parte da função de enquadramento também é tratada pela camada MAC.

5.5.2 Ethernet Padrão

Referimo-nos à tecnologia Ethernet original com uma taxa de transferência de dados de 10 Mbps como Ethernet Padrão. Embora a maioria das implementações já tenha migrado para outras tecnologias ao longo da evolução da Ethernet, existem algumas características da Ethernet Padrão que não foram alteradas durante essa evolução. Discutimos essa versão básica com o objetivo de preparar o terreno para a compreensão das outras três tecnologias.

Formato dos quadros

O quadro Ethernet contém sete campos, conforme mostra a Figura 5.55.

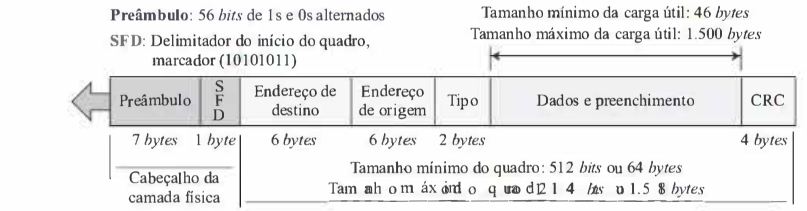


Figura 5.55 Quadro Ethernet.

- **Preâmbulo.** Campo que contém 7 *bytes* (56 *bits*) de 0s e 1s alternados, os quais alertam o sistema receptor da chegada de um quadro e permitem que ele sincronize seu relógio caso ele esteja fora de sincronismo. A sequência de *bits* fornece apenas um alerta e um pulso de sincronismo. A sequência de 56 *bits* permite que as estações percam alguns *bits* no início do quadro. O *preâmbulo* é, na verdade, adicionado na camada física e não é (formalmente) parte do quadro.
- **Delimitador de Início do Quadro.** Campo de 1 *byte*, com o valor $(10101011)_2$, que indica o início do quadro. O campo SFD (do inglês Start Frame Delimiter) alerta a estação ou estações de que esta é a última chance para realizar a sincronização. Os últimos 2 *bits* são $(11)_2$ e alertam o receptor de que o campo seguinte é o endereço de destino. Esse campo é, na verdade, um marcador que sinaliza o início do quadro. Precisamos ter em mente que um quadro Ethernet é um quadro de tamanho variável. É preciso adicionar um marcador para sinalizar o início do quadro. O campo SFD também é adicionado na camada física.
- **Endereço de destino.** Campo que tem seis *bytes* (48 *bits*), o endereço de destino (ED) contém o endereço da camada de enlace da estação (ou estações) de destino que receberá o pacote. Discutiremos o endereçamento em breve. Quando o receptor vê o seu próprio endereço da camada de enlace, um endereço *multicast* para um grupo do qual o receptor é membro, ou um endereço de *broadcast*, ele desencapsula os dados do quadro e os passa para o protocolo da camada superior definido pelo valor do campo de *tipo*.
- **Endereço de origem.** Campo que também tem seis *bytes*, o endereço de origem (EO) contém o endereço da camada de enlace do remetente do pacote. Vamos discutir o endereçamento em breve.
- **Tipo.** Campo que define o protocolo da camada superior cujo pacote está encapsulado no quadro. Esse protocolo pode ser IP, ARP, OSPF e assim por diante. Em outras palavras, ele serve ao mesmo propósito que o campo de protocolo em um datagrama e que o número da porta em um segmento ou datagrama de usuário. É utilizado para permitir a multiplexação e demultiplexação.
- **Dados.** Campo que transporta os dados encapsulados provenientes dos protocolos da camada superior. Seu comprimento mínimo é de 46 *bytes* e o máximo é de 1.500 *bytes*. Discutiremos a razão para esses valores mínimos e máximos em breve. Se os dados provenientes da camada superior têm um comprimento maior que 1.500 *bytes*, eles devem ser fragmentados e encapsulados em mais de um quadro. Se o tamanho for inferior a 46 *bytes*, ele precisa ser complementado com 0s extras (*padding*). Um quadro de dados preenchido é entregue ao protocolo da camada superior sem alterações (sem remover os *bits* de enchimento), o que significa que a camada superior é responsável por remover ou adicionar o preenchimento. O protocolo da camada superior precisa saber o comprimento dos seus dados. Por exemplo, um datagrama tem um campo que define o tamanho dos dados.
- **CRC.** O último campo contém informações para a detecção de erros, neste caso, um CRC-32. O CRC é calculado sobre os campos de endereços, tipo e dados. Se o receptor calcular o CRC e identificar que ele não vale zero (corrompido na transmissão), ele descarta o quadro.

Serviço não orientado à conexão e não confiável

A Ethernet fornece um serviço não orientado à conexão, o que significa que cada quadro enviado é independente do quadro anterior e do seguinte. A Ethernet não apresenta qualquer fase de estabelecimento de conexão ou de encerramento de conexão. O remetente envia um quadro sempre que ele o tiver, de modo que o receptor pode ou não estar pronto para recebê-lo. O remetente pode sobrecarregar o receptor com quadros, o que pode resultar no descarte de quadros. Se um quadro for descartado, o remetente não será informado sobre o ocorrido. Como o IP, que usa os serviços da Ethernet, também não é orientado à conexão, ele também não será informado sobre o ocorrido. Se a camada de transporte também for um protocolo não orientado à conexão, como é o caso do UDP,

o quadro é perdido e sua recuperação só poderá ser possível se realizada pela camada de aplicação. Entretanto, se a camada de transporte estiver usando TCP, o TCP no remetente não receberá a confirmação para o seu segmento e o enviará novamente.

A Ethernet também não é confiável, assim como ocorre com o IP e com o UDP. Se um quadro for corrompido durante a transmissão e o receptor descobrir a corrupção, o que acontece com alta probabilidade devido ao CRC-32, o receptor descarta o quadro silenciosamente. Perceber o ocorrido faz parte dos deveres dos protocolos das camadas mais altas.

Tamanho dos quadros

A Ethernet impôs restrições sobre os tamanhos mínimo e máximo de um quadro. A restrição do tamanho mínimo é necessária para o correto funcionamento do CSMA/CD, como veremos mais adiante. Um quadro Ethernet deve ter um comprimento mínimo de 512 *bits* ou 64 *bytes*. Parte desse comprimento refere-se ao cabeçalho e ao rodapé. Se contarmos os 18 *bytes* de cabeçalho e rodapé (6 *bytes* de endereço de origem, 6 *bytes* de endereço de destino, 2 *bytes* de tipo e 4 *bytes* de CRC), então o comprimento mínimo dos dados provenientes da camada superior é de 64 – 18 = 46 *bytes*. Se o pacote da camada superior tiver um tamanho inferior a 46 *bytes*, ele é preenchido com *bytes* adicionais para compensar a diferença.

A norma define que o comprimento máximo de um quadro (sem o preâmbulo ou o campo SFD) é de 1.518 *bytes*. Se subtrairmos os 18 *bytes* do cabeçalho e do rodapé, o comprimento máximo da carga útil é de 1.500 *bytes*. A restrição do comprimento máximo tem duas razões históricas: em primeiro lugar, memória era um recurso muito caro quando a Ethernet foi concebida; uma restrição de comprimento máximo ajudava a reduzir o tamanho dos *buffers* (unidades de armazenamento temporário). Em segundo lugar, a restrição de comprimento máximo impede que uma estação monopolize o meio compartilhado, bloqueando outras estações que tenham dados para enviar.

Tamanho mínimo do quadro: 64 *bytes*	Tamanho mínimo dos dados: 46 *bytes*
Tamanho máximo do quadro: 1.518 *bytes*	Tamanho máximo dos dados: 1.500 *bytes*

Endereçamento

Cada estação em uma rede Ethernet (por exemplo, um PC, uma estação de trabalho ou uma impressora) tem a sua própria **placa de rede**, também conhecida como **Placa de Interface de Rede** (NIC – Network Interface Card). A NIC está inserida na estação e fornece a ela um endereço da camada de enlace. O endereço Ethernet apresenta 6 *bytes* (48 *bits*), normalmente escritos em *notação hexadecimal*, com dois pontos entre cada *byte*. Por exemplo, a seguir mostramos um endereço MAC do protocolo Ethernet:

4A:30:10:21:10:1A

Transmissão de *bits* de endereço

A forma como os endereços são colocados na linha é diferente da forma como eles são escritos em notação hexadecimal. A transmissão é da esquerda para a direita, *byte* a *byte*; no entanto, para cada *byte*, o *bit* menos significativo é enviado primeiro e o mais significativo é enviado por último. Isto significa que o *bit* que define se um endereço é *unicast* ou *multicast* chega primeiro ao receptor. Isso ajuda o receptor a determinar imediatamente se o pacote é *unicast* ou *multicast*.

Exemplo 5.18

Mostre como o endereço 47:20:1B:2E:08:EE é enviado.

Solução

O endereço é enviado da esquerda para a direita, *byte* a *byte*; porém, cada *byte* é enviado da direita para a esquerda, *bit* a *bit*, conforme mostrado a seguir:

Hexadecimal	47	20	1B	2E	08	EE
Binário	01000111	00100000	00011011	00101110	00001000	11101110
Transmitido ←	11100010	00000100	11011000	01110100	00010000	01110111

Endereços *unicast*, *multicast* e *broadcast*

Um endereço de origem é sempre um endereço *unicast* – o quadro vem de uma única estação. O endereço de destino, entretanto, pode ser *unicast*, *multicast* ou *broadcast*. A Figura 5.56 mostra como distinguir um endereço *unicast* de um endereço *multicast*. Se o *bit* menos significativo do primeiro *byte* de um endereço de destino for 0, o endereço é *unicast*; caso contrário, ele é *multicast*.

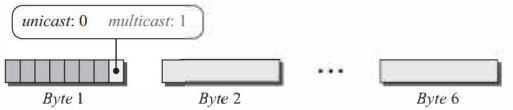


Figura 5.56 Endereços *unicast* e *multicast*.

Perceba que, da forma como os *bits* são transmitidos, o *bit* que diferencia o *unicast* do *multicast* é o primeiro *bit* transmitido e recebido. O endereço *broadcast* é um caso especial do endereço *multicast*: os destinatários são todas as estações na LAN. Um endereço de destino *broadcast* é formado por 48 *bits* 1.

Exemplo 5.19

Determine o tipo dos seguintes endereços de destino:

a. 4A:30:10:21:10:1A
b. 47:20:1B:2E:08:EE
c. FF:FF:FF:FF:FF:FF

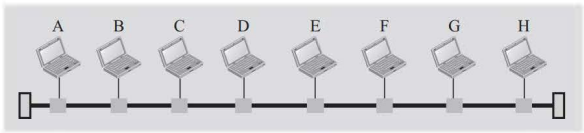
Solução

Para determinar o tipo de endereço, é preciso inspecionar o segundo dígito hexadecimal a partir da esquerda. Se ele for par, o endereço é de *unicast*. Se for ímpar, o endereço é de *multicast*. Se todos os dígitos forem Fs, o endereço é de *broadcast*. Portanto, temos o seguinte:

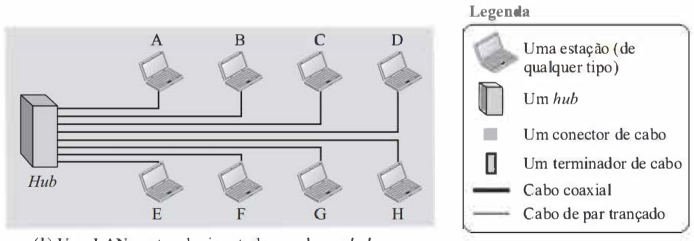
a. É um endereço *unicast* porque A em binário é 1010 (par).
b. É um endereço *multicast* porque 7 em binário é 0111 (ímpar).
c. É um endereço *broadcast*, pois todos os dígitos são Fs em hexadecimal.

Distinção entre as transmissões *unicast*, *multicast* e *broadcast*

A Ethernet Padrão usa um cabo coaxial (topologia em barramento) ou um conjunto de cabos de par trançado com um *hub* (topologia em estrela), conforme mostra a Figura 5.57.



(*a*) Uma LAN com uma topologia em barramento usando um cabo coaxial



(*b*) Uma LAN em topologia estrela usando um *hub*

Figura 5.57 Implementação da Ethernet Padrão.

É preciso dizer que a transmissão na Ethernet Padrão é sempre via *broadcast*, não importando se a intenção é fazer *unicast*, *multicast* ou *broadcast*. Em uma topologia em barramento, quando a estação A envia um quadro para a estação B, todas as estações recebem esse quadro. Na topologia em estrela, quando uma estação envia um quadro para a estação B, o *hub* o receberá. Como o *hub* é um elemento passivo, ele não verifica o endereço de destino do quadro; ele regenera os *bits* (caso o sinal tenha se enfraquecido) e os envia para todas as estações com exceção da estação A. Na realidade, ele inunda a rede com o quadro.

A questão, então, é como as transmissões *unicast*, *multicast* e *broadcast* são de fato diferenciadas umas das outras. A resposta está na forma como os quadros são guardados ou descartados.

- Em uma transmissão *unicast*, todas as estações recebem o quadro, mas enquanto o destinatário guarda e manipula esse quadro, as outras estações o descartam.
- Em uma transmissão *multicast*, todas as estações recebem o quadro, mas enquanto as estações que são membros do grupo guardam e manipulam esse quadro, as outras estações o descartam.
- Em uma transmissão *broadcast*, todas as estações (exceto pelo emissor) recebem o quadro e todas elas guardam e manipulam esse quadro.

Método de acesso

Como a rede que usa o protocolo Ethernet Padrão é uma rede de *broadcast*, precisamos usar um método de acesso para controlar o acesso ao meio compartilhado. A Ethernet Padrão adotou o CSMA/CD com o método 1-persistente discutido anteriormente nas Figuras 5.37 a 5.40. Analisaremos um cenário para entender como esse método funciona para o protocolo Ethernet.

- Considere que a estação A na Figura 5.57 tenha um quadro para enviar à estação D. A estação A deve primeiramente verificar se outra estação está transmitindo (detecção de portadora). Esta estação mede o nível de energia no meio (por um curto intervalo de tempo, normalmente menos de 100 µs). Se não houver qualquer energia de sinal no meio, significa que nenhuma estação está transmitindo (ou que o sinal ainda não atingiu a estação A). A estação A interpreta essa situação como "meio ocioso". Ela, então, começa a enviar seu quadro. Por outro lado, se o nível de energia de sinal não for zero, significa que o meio está sendo usado por outra estação. A estação A monitora continuamente o meio até que ele fique ocioso por 100 µs. Logo em seguida, ela inicia o envio do quadro. No entanto, a estação A precisa manter uma cópia do quadro em seu *buffer* até ter certeza de que não houve uma colisão. O que acontece após a estação A ter certeza disso é o assunto que discutiremos a seguir.

- A escuta do meio não se encerra após a estação A começar a enviar o quadro. A estação A precisa enviar dados e ouvir o meio continuamente. Dois casos podem ocorrer:

  a. A estação A envia 512 *bits* e nenhuma colisão é detectada nesse período (o nível de energia não ficou acima do nível de energia normal). A estação tem, então, certeza de que o quadro foi enviado e interrompe a escuta do meio. De onde vem o número de 512 *bits*? Se considerarmos que a taxa de transmissão da Ethernet é 10 Mbps, significa que a estação leva 512/(10 Mbps) = 51,2 µs para enviar 512 *bits*. Com a velocidade de propagação em um cabo (2×10^8 metros), o primeiro *bit* poderia ter viajado 10.240 metros (ida) ou apenas 5.120 metros (ida e volta), ter colidido com um *bit* da última estação do cabo, e voltado. Em outras palavras, se uma colisão vier a ocorrer, ela deve acontecer até o momento em que o remetente enviou 512 *bits* (pior caso) e que o primeiro *bit* fez uma viagem de 5.120 metros. Devemos notar que, se a colisão acontecer no meio do cabo, não no final, a estação A percebe a colisão mais cedo e aborta a transmissão. Precisamos também mencionar outra questão. O pressuposto é que o comprimento do cabo é de 5.120 metros, mas na verdade, por projeto, a Ethernet Padrão, impõe uma restrição de 2.500 metros porque é preciso considerar os atrasos encontrados ao longo do caminho. Isso significa que o pior caso é considerado. A ideia toda é que, se a estação A não detectar a colisão antes de enviar 512 *bits*, não deve ter havido qualquer colisão, porque durante esse tempo, o primeiro *bit* já deve ter atingido a extremidade da linha e todas as outras estações sabem que uma estação está enviando dados e abstêm-se de transmitir. Em outras palavras, o problema ocorre quando alguma outra estação (por exemplo, a última estação) começa a transmitir antes que o primeiro *bit* enviado pela estação A atinja aquela estação. A outra estação erroneamente pensa que a linha está livre porque o primeiro *bit* ainda não chegou até ela. O leitor deve observar que a restrição de 512 *bits*, na verdade, ajuda a estação que está transmitindo: ela tem certeza de que nenhuma colisão ocorrerá se ela não acontecer durante a transmissão dos primeiros 512 *bits*, por isso ela pode descartar a cópia do quadro de seu *buffer*.

  b. A estação A detecta uma colisão antes de terminar de enviar 512 *bits*. Isto significa que um dos *bits* anteriores colidiu com um *bit* enviado por outra estação. Nesse caso, ambas as estações devem abster-se de enviar e manter os quadros em seus *buffers* para reenviá-los quando a linha voltar a ficar disponível. No entanto, para informar outras estações de que houve uma colisão na rede, as estações envia um sinal de interferência (*jamming*) de 48 *bits*. O objetivo do sinal de interferência é criar sinal suficiente (mesmo se a colisão acontecer após alguns poucos *bits*) para alertar as outras estações sobre a colisão. Depois de enviar o sinal de interferência, as estações precisam incrementar o valor de K (número de tentativas). Se, após ser incrementado, K atingir o valor de 15, a experiência mostrou que a rede está muito ocupada; a estação deve interromper seus esforços e tentar novamente. Se $K < 15$, a estação pode esperar um tempo de retardo (T_B na Figura 5.40) e reiniciar o processo. Como mostra a Figura 5.40, a estação cria um número aleatório entre 0 e $2^K - 1$, o que significa que cada vez que ocorrer uma colisão, a faixa de valores possíveis para o número aleatório aumenta exponencialmente. Após a primeira colisão ($K = 1$) o número aleatório

encontra-se na faixa (0,1). Após a segunda colisão (K = 2), ele está na faixa (0, 1, 2, 3). Após a terceira (K = 3), a faixa passa a ser (0, 1, 2, 3, 4, 5, 6, 7). Assim, depois de cada colisão, aumenta a probabilidade de o tempo de retardo ficar maior. Isto acontece pelo fato de que, se a colisão acontecer mesmo após a terceira ou quarta tentativa, significa que a rede está realmente ocupada; um maior tempo de retardo é necessário.

Eficiência da Ethernet Padrão

A eficiência da Ethernet é definida como a razão entre o tempo utilizado por uma estação para enviar dados e o tempo em que o meio é ocupado por ela. A eficiência prática da Ethernet Padrão foi medida como

$$\text{Eficiência} = 1 / (1 + 6{,}4 \times a)$$

onde o parâmetro a é o número de quadros que podem ser colocados simultaneamente no meio. Ele pode ser calculado como a = (atraso de propagação)/(atraso de transmissão), pois o atraso de transmissão é o tempo necessário para um quadro de tamanho médio ser colocado no meio e o atraso de propagação é o tempo que ele leva para atingir a extremidade do meio. Perceba que, à medida que o valor do parâmetro **a** diminui, a eficiência aumenta. Isto significa que, à medida que o comprimento do meio de comunicação for menor ou o tamanho do quadro for maior, tem-se um aumento da eficiência. No caso ideal, $a = 0$ e a eficiência é 1. Pedimos para o leitor calcular essa eficiência nos problemas do fim do capítulo.

Exemplo 5.20

Na Ethernet Padrão, com a taxa de transmissão de 10 Mbps, consideramos que o comprimento do meio é de 2.500 m e que o tamanho do quadro é de 512 *bits*. A velocidade de propagação de um sinal em um cabo é normalmente de 2×10^8 m/s.

Atraso de propagação = $2500/(2 \times 10^8) = 12{,}5\ \mu s$	Atraso de transmissão = $512/(10^7) = 51{,}2\ \mu s$
$a = 12{,}5/51{,}2 = 0{,}24$	Eficiência = 39%

O exemplo mostra que $a = 0{,}24$, o que significa que apenas 0,24 de um quadro ocupa o meio todo nesse caso. A eficiência é de 39%, considerado um valor moderado; isto significa que em apenas 61% do tempo o meio está ocupado, mas não está sendo utilizado por uma estação.

Implementação

● protocolo Ethernet Padrão definia várias implementações possíveis, mas apenas quatro delas se tornaram populares durante a década de 1980. A Tabela 5.6 mostra um resumo das implementações da Ethernet Padrão.

Tabela 5.6 Resumo das implementações da Ethernet Padrão.

Implementação	Meio	Comprimento do meio	Codificação
10Base5	Cabo coaxial grosso	500 m	Manchester
10Base2	Cabo coaxial fino	185 m	Manchester
10Base-T	2 UTPs	100 m	Manchester
10Base-F	2 Fibras	2000 m	Manchester

Na nomenclatura 10BaseX, o prefixo numérico define a taxa de transferência de dados (10 Mbps), o termo *Base* significa sinal (digital) de banda base e X define aproximadamente o comprimento máximo do cabo em unidades de 100 metros (por exemplo, 5 para 500 ou 2 para 185 metros) ou o tipo do cabo, T para cabo de par trançado sem blindagem (UTP – Unshielded Twisted Pair) e F para fibra óptica. A Ethernet Padrão utiliza um sinal de banda base, o que significa que os *bits* são transformados em um sinal digital e enviados diretamente para a linha. Todas as implementações usam codificação Manchester, que será discutida em detalhes no Capítulo 7.

5.5.3 Fast Ethernet (100 Mbps)

Na década de 1990, algumas tecnologias de LAN com taxas de transmissão superiores a 10 Mbps, como FDDI e Fiber Channel, apareceram no mercado. Se o padrão Ethernet quisesse sobreviver, ele seria obrigado a competir com essas tecnologias. A Ethernet deu um grande salto, aumentando a taxa de transmissão para 100 Mbps, e a nova geração foi chamada de Fast Ethernet ou Ethernet Rápida. Os projetistas da Fast Ethernet precisavam torná-la compatível com a Ethernet Padrão. A subcamada MAC manteve-se inalterada, o que significava que o formato do quadro e os tamanhos máximo e mínimo também podiam permanecer inalterados. Devido ao aumento da taxa de transmissão, características da Ethernet Padrão que dependiam da taxa de transmissão, método de acesso e implementação, tiveram que ser reconsideradas.

Método de acesso

Lembre-se que o funcionamento correto do CSMA/CD depende da taxa de transmissão, do tamanho mínimo do quadro e do comprimento máximo da rede. Se quisermos conservar o tamanho mínimo do quadro, o comprimento máximo da rede deve ser alterado. Em outras palavras, se o tamanho mínimo do quadro continua sendo de 512 *bits*, e ele for transmitido 10 vezes mais rápido, a colisão precisa ser detectada 10 vezes mais cedo, ou seja, o comprimento máximo da rede deve ser 10 vezes menor (a velocidade de propagação não se altera). Assim, a Fast Ethernet surgiu com duas soluções (ela funciona com qualquer uma destas opções):

1. A primeira solução era remover completamente a topologia de barramento; nesse caso, utiliza-se um *hub* passivo e uma topologia em estrela, mas reduz-se o tamanho máximo da rede para 250 metros em vez dos 2.500 metros da Ethernet Padrão. Essa abordagem mantém a compatibilidade com a Ethernet Padrão.
2. A segunda solução é usar um *switch* de camada de enlace (discutido mais adiante neste capítulo) com *buffers* para armazenar quadros e com conexão *full-duplex* para cada estação, de modo a criar um meio de transmissão privado para cada estação. Nesse caso, não é necessário utilizar CSMA/CD, pois as estações não competem umas com as outras. O *switch* da camada de enlace recebe um quadro de uma estação de origem e o armazena em seu *buffer* (fila), de modo que ele fica à espera de processamento. Em seguida, o *switch* verifica o endereço de destino do quadro e o envia pela interface correspondente. Como a conexão com o *switch* é *full-duplex*, a estação de destino pode até mesmo enviar um quadro para outra estação ao mesmo tempo em que ela recebe um quadro. Em outras palavras, o meio compartilhado é transformado em diversos meios de comunicação ponto a ponto, não havendo necessidade de mecanismos de contenção.

Autonegociação

Um novo recurso adicionado à Fast Ethernet é a *autonegociação*. Ela fornece a uma estação ou *hub* uma variedade de recursos. A autonegociação permite que dois dispositivos negociem o modo e a taxa de transferência de dados da operação. Esse mecanismo foi concebido especialmente para

permitir que dispositivos incompatíveis possam se conectar uns aos outros. Por exemplo, um dispositivo com uma taxa de transferência de dados máxima de 10 Mbps pode se comunicar com um dispositivo com uma taxa de transferência de dados de 100 Mbps (desde que este último seja capaz de operar a uma taxa mais baixa).

Implementação

A implementação da Fast Ethernet na camada física pode ser categorizada dependendo se ela usa dois ou quatro fios. A implementação de dois fios pode consistir em um Par Trançado Blindado (STP – Shielded Twisted Pair), denominado 100BASE-TX, ou em um cabo de fibra óptica, denominado 100Base-FX. A implementação de quatro fios foi concebida apenas para uso com Par Trançado Sem Blindagem (UTP – Unshielded Twisted Pair), conhecido como 100Base-T4. A Tabela 5.7 é um resumo das implementações da Fast Ethernet. Discutiremos as codificações no Capítulo 7.

Tabela 5.7 Resumo de implementações da Fast Ethernet.

Implementação	Meio	Comprimento do meio	Número de fios	Codificação
100Base-TX	STP	100 m	2	4B5B + MLT-3
100Base-FX	Fibra	185 m	2	4B5B + NRZ-I
100Base-T4	UTP	100 m	4	Dois 8B/6T

5.5.4 Ethernet Gigabit

A necessidade de uma taxa de transferência de dados ainda mais elevada levou ao projeto do protocolo Ethernet Gigabit (1000 Mbps). O comitê do IEEE batizou o padrão de 802.3z. O objetivo da Ethernet Gigabit era aumentar a taxa de transferência de dados para 1 Gbps, mas mantendo o mesmo comprimento de endereço, o formato do quadro e seus comprimentos máximo e mínimo.

Subcamada MAC

Uma das principais considerações na evolução da Ethernet era manter inalterada a subcamada MAC. No entanto, para atingir uma taxa de transferência de dados de 1 Gbps, isto já não era mais possível. A Ethernet Gigabit adota duas abordagens distintas para acesso ao meio: *half-duplex* e *full-duplex*. Quase todas as implementações da Ethernet Gigabit seguem a abordagem *full-duplex* e, por isso, não discutimos o modo *half-duplex*. No modo *full-duplex*, existe um *switch* central conectado a todos os computadores ou outros *switches*. Nesse modo, para cada porta de entrada, cada *switch* tem *buffers* nos quais os dados são armazenados até serem transmitidos. Como o *switch* usa o endereço de destino do quadro e o envia pela porta conectada a este destino em particular, não há colisões. Isto significa que o CSMA/CD não é utilizado. A ausência de colisões implica que o comprimento máximo de um cabo é determinado pela atenuação do sinal neste cabo, não pelo processo de detecção de colisões.

Implementação

A Tabela 5.8 é um resumo das implementações da Ethernet Gigabit. O-C e O-L significam onda curta e onda longa, respectivamente. Discutimos questões de codificação no Capítulo 7.

Tabela 5.8 Resumo das implementações da Ethernet Gigabit.

Implementação	Meio	Comprimento do meio	Número de fios	Codificação
1000Base-SX	Fibra O-C	550 m	2	8B/10B + NRZ
1000Base-LX	Fibra O-L	5000 m	2	8B/10B + NRZ
1000Base-CX	STP	25 m	2	8B/10B + NRZ
1000Base-T4	UTP	100 m	4	4D-PAM5

5.5.5 Ethernet 10-Gigabit

Nos últimos anos, a Ethernet passou a ser revista para o uso em áreas metropolitanas. A ideia é estender a tecnologia, a taxa de transferência de dados e a distância de cobertura, de modo que a Ethernet possa ser usada como LAN e MAN (*Metropolitan Area Network*, ou Rede de Área Metropolitana). O comitê do IEEE criou a Ethernet 10-Gigabit e a batizou de Padrão 802.3ae. Os objetivos de projeto da Ethernet 10-Gigabit podem ser resumidos como a elevação da taxa de transferência de dados para 10 Gbps, a manutenção do mesmo tamanho e formato dos quadros, e a possibilidade de interconexão de LANs, MANs e WANs. Atualmente, essa taxa de transferência de dados só é possível com a tecnologia de fibra óptica. O padrão define dois tipos de camadas físicas: LAN PHY e WAN PHY. A primeira é projetada para suportar LANs existentes; já a segunda, na verdade, define uma WAN com enlaces conectados usando SONET OC-192 (discutido mais adiante).

Implementação

A Ethernet 10-Gigabit funciona apenas em modo *full-duplex*, o que significa que não há necessidade de mecanismos de contenção; o CSMA/CD não é usado na Ethernet 10-Gigabit. Quatro implementações são as mais comuns: 10GBase-SR, 10GBase-LR, 10GBase-EW e 10GBase-X4. A Tabela 5.9 mostra um resumo das implementações da Ethernet 10-Gigabit. Discutiremos questões de codificação no Capítulo 7.

Tabela 5.9 Resumo das implementações da Ethernet 10-Gigabit.

Implementação	Meio	Comprimento do meio	Número de fios	Codificação
10GBase-SR	Fibra de 850 nm	300 m	2	64B66B
10GBase-LR	Fibra de 1310 nm	10 Km	2	64B66B
10GBase-EW	Fibra de 1350 nm	40 Km	2	SONET
10GBase-X4	Fibra de 1310 nm	300 m a 10 Km	2	8B10B

5.5.6 LANs virtuais

Uma estação é considerada parte de uma LAN se ela pertencer fisicamente àquela LAN. O critério de participação na LAN é geográfico. O que acontece se precisarmos de uma conexão virtual entre duas estações pertencentes a duas LANs físicas distintas? Basicamente, podemos definir uma **Rede Local Virtual** (VLAN – Virtual Local Area Network) como uma rede local configurada por *software*, e não pelo cabeamento físico.

Usaremos um exemplo para elaborar melhor essa definição. A Figura 5.58 mostra uma LAN comutada em uma empresa de engenharia na qual dez estações são agrupadas em três LANs conectadas por um *switch*.

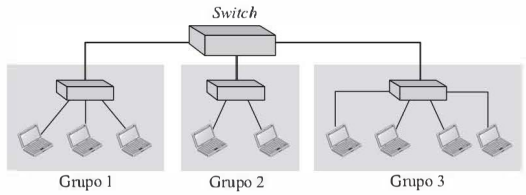


Figura 5.58 Um *switch* conectando três LANs.

Os primeiros três engenheiros trabalham juntos como o primeiro grupo, os outros dois engenheiros trabalham juntos como o segundo grupo e os últimos quatro engenheiros trabalham juntos como o terceiro grupo. A LAN é configurada para permitir esse arranjo.

Mas o que aconteceria se os administradores precisassem mover dois engenheiros do primeiro grupo para o terceiro grupo com o objetivo de acelerar o projeto que está sendo realizado pelo terceiro grupo? A configuração da LAN precisaria ser alterada. O técnico da rede deverá reconfigurar os fios/cabos. O problema repete-se se, uma semana depois, os dois engenheiros voltarem para seu grupo original. Em uma LAN comutada, mudanças nos grupos de trabalho se traduzem em mudanças físicas na configuração da rede.

A Figura 5.59 mostra a mesma LAN comutada dividida em VLANs. A ideia da tecnologia de VLAN é dividir uma LAN em segmentos lógicos em vez de físicos. A LAN pode ser dividida em várias LANs lógicas denominadas VLANs. Cada VLAN constitui um grupo de trabalho na organização. Se uma pessoa se mover de um grupo para outro, não é necessário alterar a configuração física. A participação em grupos nas VLANs é definida via *software*, não via *hardware*. Qualquer estação pode ser logicamente movida para outra VLAN. Todos os membros pertencentes a uma VLAN podem receber mensagens de *broadcast* enviadas àquela VLAN em particular. Isto significa que, se uma estação é movida da VLAN 1 para a VLAN 2, ela recebe mensagens de *broadcast* enviadas para a VLAN 2, porém não recebe mensagens de *broadcast* enviadas para a VLAN 1.

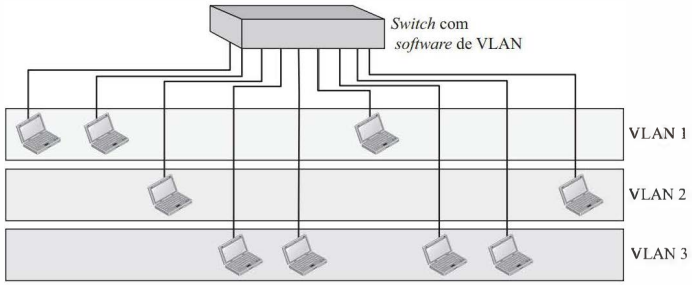


Figura 5.59 Um *switch* usando *software* de VLAN.

É óbvio que o problema ilustrado no exemplo anterior pode ser facilmente resolvido usando VLANs. Mover engenheiros de um grupo para outro via *software* é mais fácil do que alterar a configuração da rede física.

A tecnologia de VLAN permite ainda que estações conectadas a *switches* diferentes sejam agrupadas em uma VLAN. A Figura 5.60 mostra o *backbone* de uma rede local com dois *switches* e três VLANs. Cada VLAN tem estações conectadas tanto ao *switch* A como ao *switch* B.

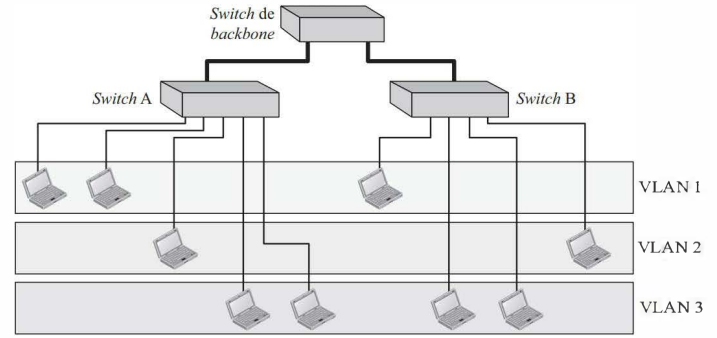


Figura 5.60 Dois *switches* em um *backbone* usando *software* de VLAN.

Essa é uma configuração interessante para uma empresa que possui dois edifícios separados. Cada edifício pode ter sua própria LAN comutada, as quais são conectadas por um *backbone*. As pessoas no primeiro edifício e as pessoas no segundo edifício podem pertencer ao mesmo grupo de trabalho apesar de estarem conectadas a diferentes LANs físicas.

Considerando esses três exemplos, podemos ver que uma VLAN define domínios de *broadcast*. As VLANs agrupam estações pertencentes a uma ou mais LANs físicas em domínios de *broadcast*. As estações em uma VLAN se comunicam umas com as outras como se estivessem conectadas a um mesmo segmento físico.

Associação a grupos

Qual característica pode ser usada para agrupar estações em uma VLAN? Diferentes fornecedores usam diferentes características, tais como número da interface, números de porta, endereços MAC, endereços IP, endereços IP *multicast* ou uma combinação de duas ou mais delas.

Números de interface

Alguns fornecedores de soluções de VLAN usam os números de interface do *switch* como a característica que define a associação das estações aos grupos. Por exemplo, o administrador pode definir que as estações conectadas às portas 1, 2, 3 e 7 pertencem à VLAN 1, que as estações conectadas às portas 4, 10 e 12 pertencem à VLAN 2, e assim por diante.

Endereços MAC

Alguns fornecedores de soluções de VLAN usam o endereço MAC de 48 *bits* como uma característica que define a associação das estações aos grupos. Por exemplo, o administrador pode estipular que as estações com os endereços MAC E2:13:42:A1:23:34 e F2:A1:23:BC:D3:41 pertencem à VLAN 1.

Endereços IP

Alguns fornecedores de soluções de VLAN usam o endereço IP de 32 *bits* (ver Capítulo 4) como uma característica que define a associação das estações aos grupos. Por exemplo, o administrador pode estipular que as estações com os endereços IP 181.34.23.67, 181.34.23.72, 181.34.23.98 e 181.34.23.112 pertencem à VLAN 1.

Endereços IP *multicast*

Alguns fornecedores de soluções de VLAN usam o endereço IP *multicast* (ver Capítulo 4) como uma característica que define a associação das estações aos grupos. O *multicast* na camada IP se traduz então em *multicast* na camada de enlace de dados.

Combinação

Mais recentemente, o *software* disponibilizado por alguns fornecedores passou a permitir combinações de todas essas características. O administrador pode escolher uma ou mais características ao instalar o *software*. Além disso, o *software* pode ser reconfigurado para modificar essas escolhas.

Configuração

Como as estações são agrupadas em VLANs diferentes? As estações são configuradas usando uma de três formas: manual, semiautomática ou automática.

Configuração manual

Na configuração manual, o administrador da rede usa o *software* de VLAN para atribuir manualmente as estações a diferentes VLANs durante a instalação. Mais tarde, a migração de uma VLAN para outra também é feita manualmente. Perceba que não estamos falando de uma configuração física, mas, sim, de uma configuração lógica. O termo *manualmente* aqui significa que o administrador digita os números de portas, os endereços IP ou outras características utilizando o *software* de VLAN.

Configuração automática

Em uma configuração automática, as estações são automaticamente conectadas ou desconectadas de uma VLAN com base em critérios definidos pelo administrador. Por exemplo, o administrador pode especificar o número do projeto como o critério para que uma estação seja membro de um grupo. Quando um usuário muda de projeto, ele automaticamente migra para uma nova VLAN.

Configuração semiautomática

Uma configuração semiautomática é uma solução intermediária entre uma configuração manual e outra automática. Normalmente, a inicialização é feita manualmente, mas as migrações são feitas automaticamente.

Comunicação entre *switches*

Em um *backbone* com múltiplos *switches*, cada *switch* deve saber não apenas qual estação pertence a qual VLAN, mas também a associação de estações conectadas a outros *switches*. Por exemplo, na Figura 5.60, o *switch* A deve conhecer a associação atual das estações conectadas ao *switch* B, e o mesmo vale para o *switch* B com relação ao *switch* A. Três métodos foram concebidos com esse propósito: manutenção de tabela, marcação de quadros e multiplexação por divisão de tempo.

Manutenção de tabela

Neste método, quando uma estação envia um quadro via *broadcast* para os membros do seu grupo, o *switch* cria uma entrada em uma tabela e registra a associação da estação. Os *switches* enviam suas tabelas uns aos outros periodicamente para atualização.

Marcação de quadros

Neste método, enquanto um quadro trafega entre os *switches*, um cabeçalho extra é adicionado a este quadro para especificar a sua VLAN de destino. A marcação do quadro é usada pelos *switches* receptores para determinar as VLANs que devem receber a mensagem de *broadcast*.

Multiplexação por Divisão de Tempo (TDM)

Neste método, a conexão (conhecida como *trunk*) entre os *switches* é dividida em canais compartilhados no tempo (ver a discussão sobre TDM no Capítulo 7). Por exemplo, se o número total de VLANs em um *backbone* for cinco, cada conexão será dividida em cinco canais. O tráfego destinado à VLAN 1 viaja no canal 1, o tráfego destinado à VLAN 2 viaja no canal 2, e assim por diante. O *switch* que recebe o quadro determina a VLAN de destino verificando o canal pelo qual o quadro chegou.

Padrão IEEE

Em 1996, a subcomissão do IEEE 802.1 aprovou um padrão chamado 802.1Q que especifica o formato para a marcação de quadros. O padrão também define o formato a ser utilizado em *backbones* com múltiplos *switches* e permite a utilização de equipamento de fornecedores diferentes na mesma VLAN. O IEEE 802.1Q abriu caminho para uma maior padronização relativa a outras questões envolvendo VLANs. A maioria dos fornecedores já aderiu ao padrão.

Vantagens

Existem diversas vantagens em se usar VLANs.

Redução de tempo e custo

VLANs podem reduzir o custo envolvido na migração de estações de um grupo para outro. A reconfiguração física leva tempo e é custosa. Em vez de mover fisicamente uma estação para outro segmento da rede ou mesmo para outro *switch*, é muito mais fácil e rápido movê-la via *software*.

Criação de grupos de trabalho virtuais

VLANs podem ser usadas para criar grupos de trabalho virtuais. Por exemplo, em um *campus*, professores que trabalham no mesmo projeto podem enviar mensagens de *broadcast* uns aos outros sem a necessidade de pertencerem ao mesmo departamento. Isso pode reduzir tráfego como se a capacidade de *multicast* do IP estivesse sendo usada.

Segurança

VLANs podem ser usadas como uma medida adicional de segurança. Pessoas que pertencem ao mesmo grupo podem enviar mensagens de *broadcast* com a garantia de que os usuários de outros grupos não receberão tais mensagens.

5.6 OUTRAS REDES COM FIOS

Conforme discutimos no Capítulo 1, as redes que encontramos na Internet são classificadas como LANs ou WANs. No entanto, algumas vezes, não há consenso na terminologia. Por exemplo,

algumas redes de acesso, como conexões discadas ou conexões a cabo, são chamadas WANs por algumas pessoas e de Redes Metropolitanas (MANs – Metropolitan Area Networks) por outras. Nesta seção, discutimos algumas dessas redes que são, direta ou indiretamente, usadas na Internet, sem, no entanto, classificá-las como MANs ou WANs; aqui, elas serão denominadas simplesmente redes.

5.6.1 Redes ponto a ponto

Algumas redes ponto a ponto, tais como conexões discadas, DSL e a cabo, são usadas para fornecer acesso à Internet a partir das instalações dos usuários. Como essas redes utilizam uma conexão dedicada entre os dois dispositivos, elas não usam protocolos de Controle de Acesso ao Meio (MAC – Media Access Control). O único protocolo necessário é o PPP, conforme discutimos anteriormente.

Conexão discada

Redes ou conexões discadas usam os serviços fornecidos pelas redes de telefonia para transmitir dados. A rede de telefonia teve o seu início no fim do século 19*. A rede toda, também conhecida como *Rede de Telefonia Convencional* (POTS – Plain Old Telephone System), foi concebida originalmente como um sistema analógico para transmissão de voz. Com o advento da era do computador, a rede, na década de 1980, começou a transportar dados juntamente com voz. Nas últimas décadas, a rede de telefonia passou por diversas alterações técnicas. De fato, a maior parte da rede de telefonia é agora digital. A única parte que continua sendo analógica é a linha que conecta o assinante à rede de telefonia. A necessidade de transmitir dados digitais levou à invenção do modem discado.

O termo **modem** é uma palavra composta que se refere às duas entidades funcionais que compõem o dispositivo: um *modulador* de sinal e um *demodulador* de sinal. Um **modulador** cria um sinal analógico a partir de dados digitais. Um **demodulador** recupera os dados digitais a partir do sinal modulado. Tais conexões podem ser usadas apenas se uma das partes estiver usando sinalização digital (por exemplo, via um provedor de Internet). Elas são assimétricas no sentido de que a taxa de recepção de dados (fluxo de dados vindo do provedor de serviços de Internet para o PC) é no máximo de 56 kbps, enquanto a taxa de envio de dados (fluxo de dados indo do PC para o provedor de Internet) pode atingir um máximo de 33,6 kbps, conforme mostra a Figura 5.61.

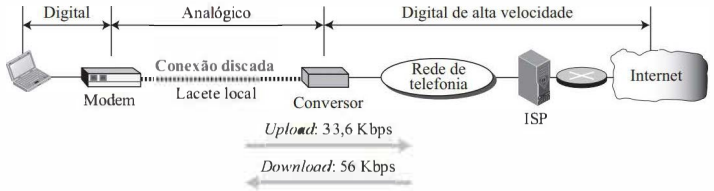


Figura 5.61 Rede discada para prover acesso à Internet.

No *upload* (envio), o sinal analógico ainda precisa ser amostrado antes de entrar na rede de telefonia digital de alta velocidade. Nessa direção, o ruído de quantização (conforme veremos no Capítulo 7) é introduzido no sinal, o que reduz a taxa de transmissão para 33,6 kbps. No entanto, não há qualquer processo de amostragem no *download* (recepção). O sinal não é afetado pelo ruído de quantização. O usuário individual que acessa a Internet normalmente precisa de uma velocidade maior para *download* que para *upload*. Esses usuários desejam, por exemplo, obter um arquivo grande com maior

* N. de T.: O leitor interessado pode encontrar informações sobre a história da rede de telefonia no Brasil no *site* do Ministério das Telecomunicações: http://www.mc.gov.br/o-ministerio/historico/historia-da-telefonia.

frequência do que desejam enviar um arquivo dessa natureza, de modo que a velocidade assimétrica normalmente passa despercebida. É preciso mencionar, entretanto, que quando uma linha de telefone é usada para acesso à Internet, ela não pode ser usada para comunicação de voz ao mesmo tempo.

Podemos nos perguntar como chegamos a uma velocidade de 56 kbps. As empresas de telefonia praticam uma frequência de 8.000 amostras por segundo, com 8 *bits* por amostra. Um dos *bits* em cada amostra é utilizado para fins de controle, o que significa que cada amostra apresenta 7 *bits*. A taxa de transferência é, portanto, 8.000 × 7, ou 56.000 bps ou 56 kbps.

Linha de Assinante Digital

Depois que os modems tradicionais atingiram o seu pico de taxa de transferência de dados, as empresas de telefonia desenvolveram uma outra tecnologia, denominada DSL, para proporcionar maior velocidade de acesso à Internet. A tecnologia **Linha de Assinante Digital** (DSL – Digital Subscriber Line) é uma das mais promissoras no sentido de suportar comunicações digitais de alta velocidade sobre a infraestrutura de telefonia existente. A tecnologia DSL consiste em um conjunto de tecnologias, cada uma com uma primeira letra diferente (ADSL, VDSL, HDSL e SDSL). O conjunto é muitas vezes denominado *x*DSL, onde o *x* pode ser substituído por A, V, H ou S. Aqui, discutimos apenas o primeiro, o ADSL. A primeira tecnologia no conjunto é a *DSL Assimétrica* (*ADSL*). A ADSL, assim como um modem de 56K, fornece maior velocidade (taxa de transferência de *bits*) na direção de recepção (da Internet para a residência) do que na direção de envio (da residência para a Internet). Por isso ela é denominada assimétrica. Contrariamente à assimetria em modems de 56K, os projetistas da ADSL dividiram propositalmente a largura de banda disponível de forma irregular no lacete local para o cliente residencial. O serviço não é adequado para clientes empresariais que necessitam de uma grande largura de banda em ambas as direções.

Usando lacetes locais existentes

Um ponto interessante é que a tecnologia ADSL utiliza as linhas telefônicas existentes (lacete local). Mas como a ADSL alcança uma taxa de dados que nunca foi conseguida com modems tradicionais? A resposta é que o cabo de par trançado usado nas linhas telefônicas é, na verdade, capaz de lidar com uma largura de banda de até 1,1 MHz, mas os filtros instalados na central local da companhia telefônica, à qual todo lacete local se conecta, limitam a largura de banda a 4 kHz (suficiente para comunicações de voz). Se o filtro for removido, no entanto, toda a faixa de 1,1 MHz fica disponível para comunicações de dados e de voz. Tipicamente, uma largura de banda disponível de 1.104 MHz é dividida em um canal de voz, um canal de recepção e um canal de envio, conforme mostra a Figura 5.62.

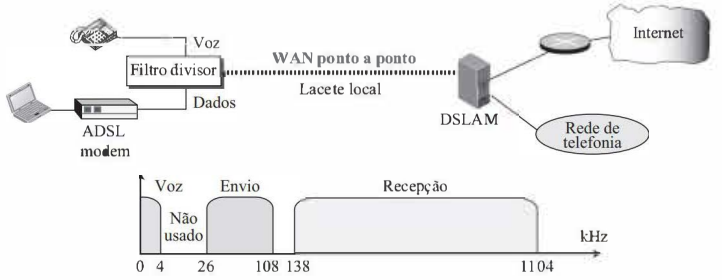


Figura 5.62 Rede (ADSL) ponto a ponto.

A ADSL permite que o assinante use o canal de voz e o canal de dados simultaneamente. A taxa de transferência na direção de envio pode chegar a 1,44 Mbps. Entretanto, a taxa de transferência de dados é normalmente inferior a 500 kbps por causa do elevado nível de ruído nesse canal. A taxa de transferência de dados na direção de recepção pode chegar a 13,4 Mbps. No entanto, esta taxa é normalmente inferior a 8 Mbps por causa do ruído nesse canal. Um ponto muito interessante é que a empresa de telefonia, nesse caso, atua como o ISP, de modo que serviços como e-mail ou acesso à Internet podem ser fornecidos pela companhia de telefonia em si.

Cabo

As redes a cabo foram originalmente criadas para fornecer acesso a programas de TV para assinantes que não conseguiam receber o sinal devido a obstáculos naturais, como montanhas. Mais tarde, as redes a cabo tornaram-se populares entre pessoas que queriam simplesmente um sinal de melhor qualidade. Além disso, as redes a cabo permitiam o acesso a estações remotas de TV por meio de conexões de micro-ondas. As empresas de TV a cabo também descobriram um mercado promissor no fornecimento de acesso à Internet, usando alguns dos canais originalmente projetados para vídeo. Depois de descrevermos a estrutura básica das redes a cabo, discutiremos como os modems a cabo podem fornecer uma conexão de alta velocidade à Internet.

Redes a cabo tradicionais

A TV a cabo começou a distribuir sinais de radiodifusão de vídeo em locais com recepção ruim ou inexistente no final da década de 1940. A tecnologia era chamada **Antena Comunitária de TV** (CATV – Community Antenna TV), porque uma antena no topo de uma colina alta ou edifício recebia os sinais das emissoras de TV e os redistribuía, via cabos coaxiais, para a comunidade. A Figura 5.63 mostra um diagrama esquemático de uma rede de TV a cabo tradicional.

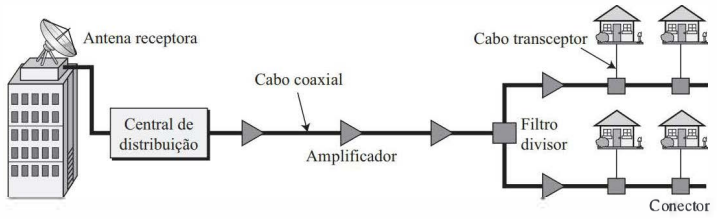


Figura 5.63 Rede de TV a cabo tradicional.

O escritório da TV a cabo, conhecido como a *central de distribuição*, recebe os sinais de vídeo das estações de radiodifusão e alimenta os sinais em cabos coaxiais. Os sinais ficam cada vez mais fracos com a distância, de modo que foram instalados amplificadores ao longo de toda a rede para renová-los. Poderia haver até 35 amplificadores entre a central de distribuição e as instalações do assinante. Na outra extremidade, filtros divisores repartem o cabo, enquanto conectores e cabos transceptores criam as conexões com as instalações do assinante.

O sistema de TV a cabo tradicional utilizava cabos coaxiais fim a fim. Devido à atenuação dos sinais e à utilização de um grande número de amplificadores, a comunicação na rede tradicional era unidirecional (apenas ida). Os sinais de vídeo eram transmitidos da central de distribuição para as instalações do assinante.

Rede Híbrida Fibra-Coaxial

A segunda geração de redes a cabo é chamada **rede Híbrida Fibra-Coaxial** (HFC – Hybrid Fiber-Coaxial). A rede usa uma combinação de fibra óptica e cabo coaxial. O meio de transmissão do escritório de TV a cabo até uma caixa, conhecida como *nó óptico*, é uma fibra óptica; do nó óptico até o interior das residências, passando pela vizinhança dos assinantes, o meio ainda é um cabo coaxial. A Figura 5.64 mostra um diagrama esquemático de uma rede HFC.

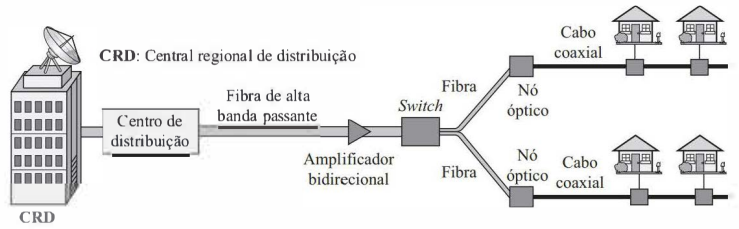


Figura 5.64 Rede Híbrida Fibra-Coaxial (HFC).

A *Central Regional de Distribuição* (CRD) normalmente serve até 400.000 assinantes. Os CRDs alimentam os *centros de distribuição*, cada qual servindo até 40.000 assinantes. O centro de distribuição desempenha um papel fundamental nessa nova infraestrutura. A modulação e a distribuição dos sinais são feitas aqui; os sinais são, então, enviados aos nós ópticos por meio de cabos de fibra óptica. O nó óptico divide os sinais analógicos de modo que o mesmo sinal é enviado para todos os cabos coaxiais. Cada cabo coaxial serve até 1.000 assinantes. A utilização de cabos de fibra óptica reduz a necessidade de amplificadores para oito ou menos.

Uma razão para migrar da infraestrutura tradicional para a híbrida é tornar a rede a cabo bidirecional (os dados trafegam nos dois sentidos).

TV a cabo para transferência de dados

As companhias de TV a cabo estão agora competindo com empresas de telefonia pelos clientes residenciais que desejam alta velocidade de transferência de dados. A tecnologia DSL oferece conexões com altas taxas de transferência de dados para assinantes residenciais por meio de um lacete local. No entanto, a tecnologia DSL utiliza os já existentes cabos de par trançado sem revestimento, os quais são muito suscetíveis a interferências. Isto impõe um limite superior à taxa de transferência de dados. Uma solução é a utilização da rede de TV a cabo. Nesta seção, discutimos brevemente essa tecnologia.

Mesmo em um sistema de HFC, a última parte da rede, do nó óptico até as instalações do assinante, ainda é formada por um cabo coaxial. Esse cabo coaxial tem uma largura de banda que varia de 5 a 750 MHz (aproximadamente). Para fornecer acesso à Internet, as empresas de TV a cabo dividiram essa largura de banda em três faixas: vídeo, recepção de dados e envio de dados, conforme mostra a Figura 5.65.

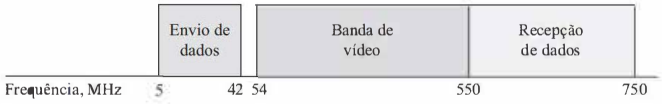


Figura 5.65 Divisão de banda pela CATV no cabo coaxial.

A *banda de vídeo* ocupa as frequências de 54 a 550 MHz. Como cada canal de TV ocupa 6 MHz, esta banda pode acomodar mais de 80 canais. A recepção de dados (vindos da Internet para as instalações do assinante) ocupa a banda superior, de 550 a 750 MHz. Essa banda também é dividida em canais de 6 MHz.

O envio de dados (das instalações do assinante para a Internet) ocupa a banda inferior, de 5 a 42 MHz. Essa banda também é dividida em canais de 6 MHz. Existem 2 *bits*/*baud* na modulação QPSK (*Quadrature Phase Shift Keying* ou Deslocamento de Fase em Quadratura). A norma especifica 1 Hz por cada *baud*[*], o que significa que, teoricamente, os dados podem ser enviados a 12 Mbps (2 *bits*/Hz × 6 MHz). No entanto, a taxa de dados é normalmente inferior a 12 Mbps.

Compartilhamento

Tanto a banda de envio como a de recepção são compartilhadas pelos assinantes. A largura de banda de dados de envio é de 37 MHz. Isto significa que existem apenas seis canais de 6 MHz disponíveis na direção de envio. Um assinante deve utilizar um canal para enviar seus dados para a Internet. A pergunta é: "Como seis canais podem ser compartilhados em uma área com 1.000, 2.000 ou até 100.000 assinantes?" A solução é o compartilhamento de tempo. A banda é dividida em canais; estes devem ser compartilhados entre os assinantes na mesma região. O provedor de serviços a cabo aloca um canal, estática ou dinamicamente, para um grupo de assinantes. Se um assinante desejar enviar dados, ele disputa o canal com os outros usuários que querem acessar a rede; o assinante deve esperar até que o canal esteja disponível.

A situação é semelhante na direção de recepção. A banda de recepção tem 33 canais de 6 MHz. Um provedor de serviços a cabo provavelmente tem mais de 33 assinantes; portanto, cada canal deve ser compartilhado entre um grupo de assinantes. No entanto, a situação é diferente na direção de recepção; nesse caso, temos um cenário de *multicast*. Se houver dados para qualquer um dos assinantes no grupo, eles são enviados para aquele canal. Os dados são enviados para todos os assinantes. Porém, como cada assinante também tem um endereço registrado junto ao provedor, o modem a cabo usado pelo grupo compara o endereço referente aos dados com o endereço atribuído pelo provedor. Se os endereços forem iguais, os dados são conservados; caso contrário, são descartados.

CM e CMTS

Para usar uma rede a cabo na transmissão de dados, precisamos de dois dispositivos essenciais: um **Modem a Cabo** (CM – Cable Modem) e um **Sistema de Transmissão por Modem a Cabo** (CMTS – Cable Modem Transmission System). O modem a cabo é colocado nas instalações do assinante. O **sistema de transmissão por modem a cabo** é instalado dentro da empresa de TV a cabo. O CMTS recebe dados da Internet e os envia para o assinante. Ele também recebe os dados do assinante e os passa para a Internet. Ele atua de forma semelhante a um modem ADSL. A Figura 5.66 mostra a localização desses dois dispositivos. Assim como na tecnologia DSL, a empresa de TV a cabo precisa se tornar um ISP e fornecer serviços de Internet para o assinante. Nas instalações do assinante, o CM separa o vídeo dos dados e os envia para o aparelho de televisão ou para o computador.

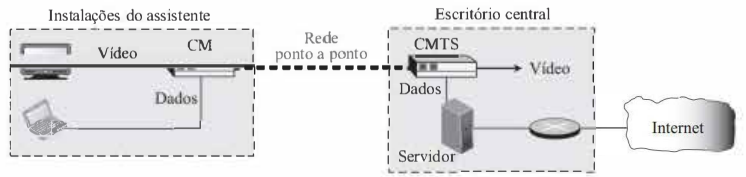


Figura 5.66 Sistema de Transmissão por Modem a Cabo (CMTS).

[*] N. de T.: O termo *baud* refere-se ao número de elementos de sinal transmitidos por segundo. Ver Capítulo 7 para obter mais detalhes.

5.6.2 SONET

Nesta seção, apresentamos uma rede de alta velocidade, a SONET, usada como uma rede de transporte para levar dados provenientes de outras redes. Primeiramente, discutimos a SONET como um protocolo, e então mostramos como as redes SONET podem ser construídas a partir dos padrões definidos no protocolo.

As elevadas larguras de banda do cabo de fibra óptica são adequadas para as tecnologias atuais que exigem alta taxa de transferência de dados (como videoconferência) e para o transporte simultâneo de dados pertencentes a um grande número tecnologias que exigem uma baixa taxa de transferência. Por isso, a importância das fibras ópticas cresce juntamente com o desenvolvimento de tecnologias que requerem alta taxa de transferência de dados ou ampla largura de banda para a transmissão. Com sua importância, veio a necessidade de padronização. Os Estados Unidos (ANSI) e a Europa (ITU-T) responderam a essa demanda com a especificação de normas que, embora independentes, são fundamentalmente semelhantes e compatíveis. O padrão ANSI é conhecido como **Rede Óptica Síncrona** (SONET – Synchronous Optical Network), e é discutido nesta seção.

Arquitetura

O SONET é uma rede síncrona que usa multiplexação síncrona por divisão de tempo TDM (Time Division Multiplexing). Todos os relógios do sistema são atrelados a um relógio-mestre. Primeiramente, introduzimos a arquitetura de um sistema SONET: sinais, dispositivos e conexões.

Sinais

O SONET define uma hierarquia de níveis de sinalização elétrica denominados **Sinais de Transporte Síncrono** (STSs – Synchronous Transport Signals). Cada nível STS (STS-1 a STS-192) suporta uma determinada taxa de transferência de dados, especificada em *megabits* por segundo (ver Tabela 5.10). Os sinais ópticos correspondentes são conhecidos como **Portadoras Ópticas** (OCs – Optical Carriers).

Tabela 5.10 Taxas de transferência no SONET.

STS	OC	Taxa (Mbps)	STS	OC	Taxa (Mbps)
STS-1	OC-1	51.840	STS-24	OC-24	1244.160
STS-3	OC-3	155.520	STS-36	OC-36	1866.230
STS-9	OC-9	466.560	STS-48	OC-48	2488.320
STS-12	OC-12	622.080	STS-96	OC-96	4976.640
STS-18	OC-18	933.120	STS-192	OC-192	9953.280

Observando a Tabela 5.10, percebemos alguns pontos interessantes. Em primeiro lugar, o nível mais baixo nessa hierarquia apresenta uma taxa de transferência de dados de 51.840 Mbps, que é maior do que aquela do serviço DS-3 (44.736 Mbps, explicado no Capítulo 7). Na realidade, o STS-1 foi projetado para acomodar taxas de dados equivalentes às do DS-3. A diferença de capacidade é fornecida para que seja possível tratar a carga adicional introduzida pelo sistema óptico. Em segundo lugar, a taxa de transferência do STS-3 é exatamente três vezes a taxa do STS-1, enquanto a taxa do STS-9 é exatamente metade da taxa do STS-18. Essas relações indicam que 18 canais STS-1 podem ser multiplexados em um canal STS-18, seis canais STS-3 podem ser multiplexados em um canal STS-18 e assim por diante.

Dispositivos SONET

A Figura 5.67 mostra uma conexão simples usando dispositivos SONET. A transmissão via SONET depende de três dispositivos básicos: multiplexadores/demultiplexadores STS, regeneradores, multiplexadores de inserção/remoção (ADM – Add/Drop Multiplexers) e terminais.

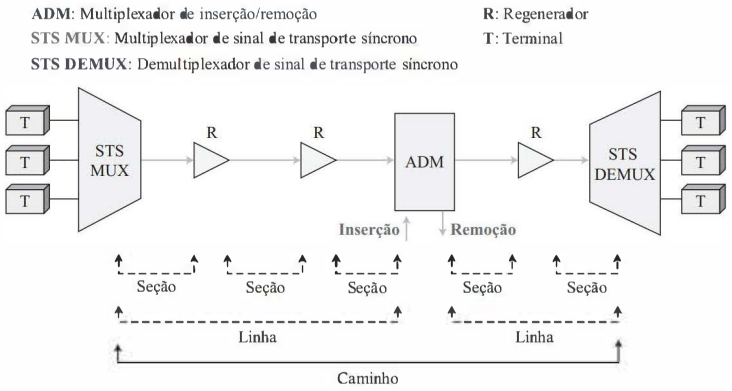


Figura 5.67 Uma rede simples usando dispositivos SONET.

Multiplexador/demultiplexador STS Multiplexadores/demultiplexadores STS marcam os pontos de início e fim de um enlace SONET. Eles fornecem a interface entre um ramo de uma rede elétrica e a rede óptica. Um *multiplexador STS* multiplexa sinais provenientes de várias fontes elétricas e cria o sinal OC correspondente. Um *demultiplexador STS* demultiplexa um sinal óptico OC nos sinais elétricos correspondentes.

Regenerador Regeneradores estendem o comprimento das conexões. Um *regenerador* é um repetidor (discutido mais adiante) que leva um sinal óptico recebido (OC-*n*), faz a sua demodulação para transformá-lo no sinal elétrico correspondente (STS-*n*), regenera o sinal elétrico e finalmente modula o sinal elétrico em seu sinal OC-*n* correspondente. Um regenerador SONET substitui algumas das informações adicionais existentes (informações de cabeçalho) por novas.

Multiplexador de inserção/remoção Multiplexadores de inserção/remoção permitem a inserção e extração de sinais. Um *Multiplexador de inserção/remoção* (ADM – Add/Drop Multiplexer) pode inserir STSs provenientes de diferentes origens em um determinado caminho, ou pode remover um sinal específico de um caminho e redirecioná-lo sem demultiplexar o sinal todo. Em vez de depender de temporização e posições de *bits*, os multiplexadores de inserção/remoção usam informações de cabeçalho, como endereços e ponteiros (descritos mais adiante nesta seção) para identificar os fluxos individuais.

Terminais Um *terminal* é um dispositivo que usa os serviços de uma rede SONET. Por exemplo, na Internet, um terminal pode ser um roteador que precisa enviar pacotes para outro roteador na outra extremidade de uma rede SONET.

Conexões

Os dispositivos definidos na seção anterior são conectados usando *seções*, *linhas* e *caminhos*.

Seções

Uma *seção* é um enlace óptico conectando dois dispositivos vizinhos: multiplexador a multiplexador, multiplexador a regenerador, ou regenerador a regenerador.

Linhas

Uma *linha* refere-se à região da rede entre dois multiplexadores.

Caminhos

Um *caminho* é uma região fim a fim da rede entre dois multiplexadores STS. Em uma SONET simples contendo dois multiplexadores STS conectados diretamente um ao outro, a seção, a linha e o caminho são equivalentes.

Camadas SONET

O padrão SONET inclui quatro camadas funcionais: a camada fotônica, a de seção, a de linha e a de caminho. Elas correspondem às camadas física e de enlace de dados. Os cabeçalhos adicionados ao quadro nas várias camadas são discutidos mais adiante neste capítulo. A Figura 5.68 mostra as camadas e a relação entre as camadas e os dispositivos descritos anteriormente.

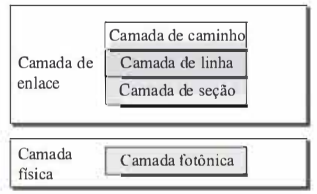


Figura 5.68 Camadas SONET comparadas com as camadas OSI e Internet.

Camada de caminho

A *camada de caminho* é responsável pela movimentação de um sinal de sua origem óptica para o seu destino óptico. Na origem óptica, o sinal é transformado de sua forma eletrônica em sua forma óptica, multiplexado com outros sinais e encapsulado em um quadro. No destino óptico, o quadro é recebido e demultiplexado, sendo os sinais ópticos individuais transformados de volta em suas formas eletrônicas. As informações relativas à camada de caminho são adicionadas nessa camada. Multiplexadores STS fornecem funções da camada de caminho.

Camada de linha

A *camada de linha* é responsável pela movimentação de um sinal ao longo de uma linha física. As informações relativas à camada de linha são adicionadas ao quadro nessa camada. Os multiplexadores STS e os multiplexadores de inserção/remoção fornecem funções da camada de linha.

Camada de seção

A *camada de seção* é responsável pela movimentação de um sinal ao longo de uma seção física. Ela lida com enquadramento, embaralhamento e controle de erros. As informações relativas à camada de seção são adicionadas ao quadro nessa camada.

Camada fotônica

A *camada fotônica* corresponde à camada física. Ela inclui especificações físicas para o canal de fibra óptica, sensibilidade do receptor, funções de multiplexação, e assim por diante. O SONET usa a codificação NRZ (ver o Capítulo 7) na qual a presença de luz representa 1 e a ausência de luz representa 0.

Quadros SONET

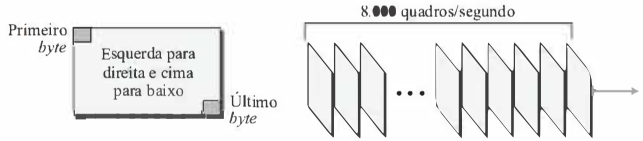
Cada sinal de transporte síncrono STS-*n* é composto por 8.000 quadros. Cada quadro consiste em uma matriz bidimensional de *bytes* com 9 linhas por 90 × *n* colunas. Por exemplo, um quadro STS-1 apresenta 9 linhas por 90 colunas (810 *bytes*), enquanto um quadro STS-3 consiste em 9 linhas por 270 colunas (2.430 *bytes*). A Figura 5.69 mostra o formato geral de um STS-1 e de um STS-*n*.

(*a*) Quadro STS-1 (*b*) Quadro STS-*n*

Figura 5.69 Um quadro STS-1 e um quadro STS-*n*.

Transmissão de quadros, *bytes* e *bits*

Um dos pontos interessantes do SONET é que cada sinal STS-*n* é transmitido a uma taxa fixa de 8.000 quadros por segundo. Essa é a taxa na qual a voz é digitalizada (ver o Capítulo 7). Para cada quadro, os *bytes* são transmitidos da esquerda para a direita, de cima para baixo. Para cada *byte*, os *bits* são transmitidos do mais significativo para o menos significativo (esquerda para a direita). A Figura 5.70 mostra a ordem de transmissão de quadros e *bytes*.



(*a*) Transmissão de *byte* (*b*) Transmissão de quadro

Figura 5.70 Quadros STS-1 em trânsito.

Um sinal SONET STS-*n* é transmitido a uma taxa de 8.000 quadros por segundo.

Se amostrarmos um sinal de voz e utilizarmos 8 *bits* (1 *byte*) para representar cada amostra, podemos dizer que cada *byte* em um quadro SONET pode transportar informações vindas de um canal de voz digitalizado. Em outras palavras, um sinal STS-1 pode transportar 774 canais de voz simultaneamente (810 menos 36 *bytes* necessários para informações de controle adicionadas pelas camadas).

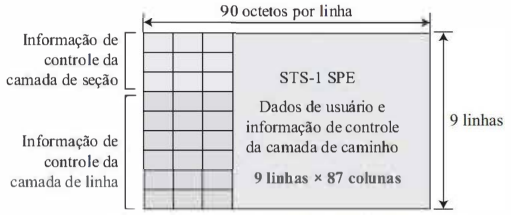
Cada *byte* em um quadro SONET pode transportar um canal de voz digitalizada.

Perceba que, no SONET, existe uma relação exata entre as taxas de transferência de dados de diferentes sinais STS. Podemos calcular a taxa de transferência de dados do STS-3 usando o taxa de transferência de dados do STS-1 (multiplicando o valor deste último por 3).

No SONET, a taxa de transferência de dados de um sinal STS-*n* é *n* vezes o valor da taxa de transferência de dados de um sinal STS-1.

Formato dos quadros STS-1

O formato básico de um quadro STS-1 é mostrado na Figura 5.71. Como dissemos anteriormente, um quadro SONET é uma matriz de 9 linhas de 90 *bytes* (octetos) cada, totalizando 810 *bytes*.



Dados do usuário (SPE – SONET Payload Envelope) na verdade ocupam 86 × 9 *bytes*

Figura 5.71 Informação de controle no quadro STS-1.

Multiplexação STS

No SONET, quadros transmitidos a uma taxa menor podem ser sincronamente multiplexados usando divisão de tempo para gerar um quadro com maior taxa de transferência. Por exemplo, três sinais (canais) STS-1 podem ser combinados em um sinal (canal) STS-3, quatro STS-3 podem ser multiplexados em um STS-12, e assim por diante.

A multiplexação é do tipo TDM síncrono, e todos os relógios na rede são atrelados a um relógio-mestre para conseguir a sincronização.

Em SONET, todos os relógios na rede são atrelados a um relógio-mestre.

Precisamos ressaltar que a multiplexação também pode acontecer com taxas de transferência de dados maiores. Por exemplo, quatro sinais STS-3 podem ser multiplexados em um sinal STS-12, conforme mencionado anteriormente. No entanto, os sinais STS-3 precisam primeiro ser demultiplexados em 12 sinais STS-1, e em seguida esses doze sinais precisam ser multiplexados em um sinal STS-12. A razão para esse trabalho extra ficará mais clara depois que discutirmos o mecanismo de intercalação de *bytes*.

Multiplexador de inserção/remoção

A multiplexação de vários sinais STS-1 em um sinal STS-*n* é feita no multiplexador STS (na camada de caminho). A demultiplexação de um sinal STS-*n* em componentes STS-1 é feita no demultiplexador STS. Entre esses dois dispositivos, no entanto, o SONET usa multiplexadores de inserção/remoção que podem substituir um sinal por outro. Precisamos deixar claro que essa não é uma demultiplexação/multiplexação no sentido convencional. Um multiplexador de inserção/remoção opera na camada de linha. Um multiplexador de inserção/remoção não adiciona informações de controle das camadas de seção, linha ou caminho. Ele funciona quase como um *switch*; remove um sinal STS-1 e insere outro. O tipo de sinal na entrada e saída de um multiplexador de inserção/remoção permanece o mesmo (ambos STS-3 ou ambos STS-12, por exemplo). O multiplexador de inserção/remoção (ADM), apenas remove os *bytes* correspondentes e os substitui pelos novos *bytes* (incluindo os *bytes* de controle das camadas de seção e de linha).

Redes SONET

Usando equipamentos SONET, podemos criar uma rede SONET que pode ser usada como um *backbone* de alta velocidade transportando dados provenientes de outras redes, como ATM ou IP. Podemos dividir as redes SONET basicamente em três categorias: redes lineares, em anel e em malha.

Rede linear

Uma rede linear é normalmente composta por multiplexadores e demultiplexadores STS, alguns geradores e alguns multiplexadores de inserção/remoção, conforme mostra a Figura 5.72.

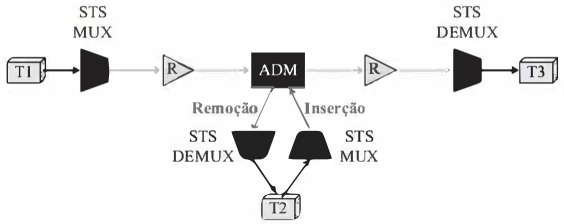


Figura 5.72 Uma rede SONET linear.

Redes em anel

ADMs permitem a criação de redes SONET em anel. Anéis SONET podem ser usados tanto em uma configuração unidirecional como bidirecional. Em cada caso, podemos acrescentar anéis extras para tornar a rede autorrecuperável, ou seja, capaz de se recuperar automaticamente de uma falha na linha. A Figura 5.73 mostra uma rede em anel.

Embora tenhamos colocado um emissor e três receptores na figura, muitas outras configurações são possíveis. O emissor usa uma conexão bidirecional para enviar dados para ambos os anéis simultaneamente; o receptor usa *switches* de seleção para escolher o anel com uma melhor qualidade de sinal. Usamos um multiplexador STS e três demultiplexadores STS para enfatizar que os nós operam na camada de caminho.

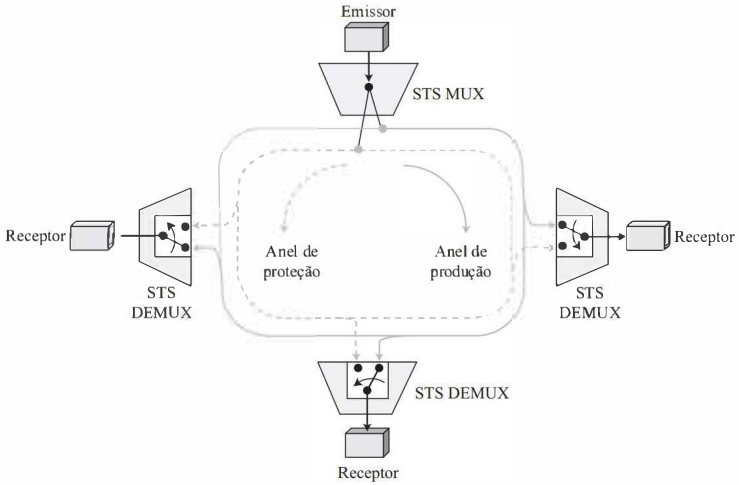
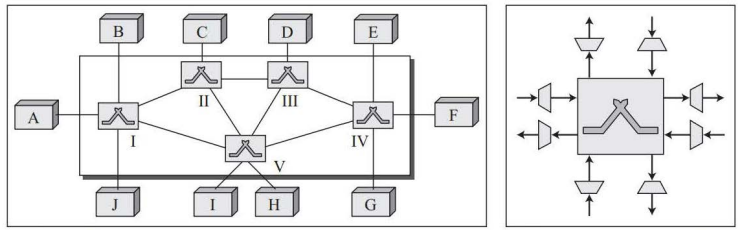


Figura 5.73 Um anel comutado unidirecional.

Redes em malha

Um problema com as redes em anel é a falta de escalabilidade. Quando o tráfego em um anel aumenta, precisamos atualizar não apenas as linhas, mas também os ADMs. Nessa situação, uma rede em malha com *switches* provavelmente apresentaria um melhor desempenho. Um *switch* em uma rede em malha é conhecido como *switch* de conexão cruzada. Um *switch* de conexão cruzada, assim como outros *switches* que discutimos anteriormente, tem portas de entrada e de saída. Em uma porta de entrada, o *switch* recebe um sinal OC-*n*, o transforma em um sinal STS-*n*, o demultiplexa nos sinais STS-1 correspondentes e envia cada sinal STS-1 para a porta de saída apropriada. Uma porta de saída recebe os sinais STS-1 provenientes de diferentes portas de entrada, os multiplexa em um sinal STS-*n*, e cria um sinal OC-*n* para a transmissão. A Figura 5.74 mostra uma rede SONET em malha e a estrutura de *switch*.



(*a*) Rede SONET em malha (*b*) *Switch* de conexão cruzada

Figura 5.74 Uma rede SONET em malha.

Tributários virtuais

O SONET foi projetado para transportar dados em banda larga. Conforme discutido mais adiante, as taxas de transferência de dados atuais na hierarquia digital (DS-1 a DS-3) são, entretanto, inferiores às do STS-1. Para tornar o SONET compatível com a hierarquia atual, o projeto de seus quadros inclui um sistema de **Tributários Virtuais** (VTs – Virtual Tributaries), conforme mostra a Figura 5.75. Um tributário virtual é uma carga parcial de dados que pode ser inserida em uma conexão STS-1 e combinada com outras cargas parciais de dados para preencher o quadro. Em vez de usar todas as 86 colunas de carga útil de um quadro STS-1 para os dados de uma única origem, podemos subdividir a Carga Útil do Envelope SONET (SPE – SONET Payload Envelope) e chamar cada componente de um VT.

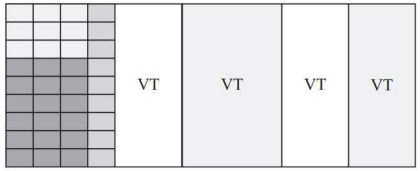


Figura 5.75 Tributários virtuais.

Foram definidos quatro tipos de VTs para acomodar hierarquias digitais existentes. Perceba que o número de colunas permitidas para cada tipo de VT pode ser determinado dobrando-se o número de identificação do tipo (o VT1.5 apresenta três colunas, o VT2 apresenta quatro colunas, etc.). O VT1.5 acomoda o serviço DS-1 dos Estados Unidos (1.544 Mbps). O VT2 acomoda o serviço europeu CEPT-1 (2.048 Mbps). O VT3 acomoda o serviço DS-1C (DS-1 fracionário, que opera a 3.152 Mbps). O VT6 acomoda o serviço DS-2 (6.312 Mbps).

Quando dois ou mais tributários são inseridos em um único quadro STS-1, eles são intercalados coluna a coluna. O SONET fornece mecanismos para identificar cada VT e separá-los sem demultiplexar o fluxo todo. A discussão sobre esses mecanismos e as questões de controle por trás deles está além do escopo deste livro.

5.6.3 Rede comutada: ATM

O **Modo de Transferência Assíncrona** (ATM – Asynchronous Transfer Mode) é um protocolo voltado a redes de longa distância comutadas e é baseado no protocolo de *comutação de células* projetado pelo Fórum ATM e adotado pelo ITU-T. A combinação de ATM com SONET permite a interconexão em alta velocidade de todas as redes do mundo. Na realidade, o ATM pode ser pensado como a "autoestrada" no termo "superautoestrada da informação"*.

O ATM utiliza multiplexação estatística (assíncrona) por divisão de tempo – razão pela qual ele é chamado de Modo de Transferência Assíncrona – para multiplexar células provenientes de diferentes canais. Ele usa faixas de tempo (*slots*) de tamanho fixo (o tamanho de uma célula). Os multiplexadores ATM preenchem uma faixa com uma célula proveniente de qualquer canal de entrada que tenha uma célula; a faixa permanece vazia se nenhum dos canais tiver uma célula para enviar. A Figura 5.76 mostra como as células provenientes de três entradas são multiplexadas. No primeiro sinal do relógio, o canal 2 não tem células (faixa de entrada vazia),

* N. de T.: O termo "superautoestrada da informação" (*information superhighway*) foi cunhado por Albert Gore na década de 1990, referindo-se aos sistemas de comunicação digital na Internet.

de modo que o multiplexador preenche a faixa com uma célula proveniente do terceiro canal. Após todas as células provenientes de todos os canais terem sido multiplexadas, as faixas de saída ficam vazias.

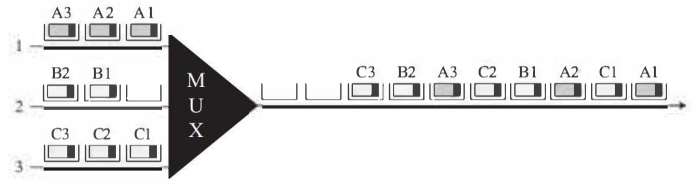


Figura 5.76 Multiplexação no ATM.

Arquitetura

Redes ATM são redes de comutação de células. Os dispositivos de acesso do usuário, denominados terminais ATM, são conectados por meio de uma **Interface Usuário-Rede** (UNI – User-to-Network Interface) aos *switches* dentro da rede. Os *switches* são conectados por meio de **Interfaces Rede-Rede** (NNIs – Network-to-Network Interfaces). A Figura 5.77 mostra um exemplo de uma rede ATM.

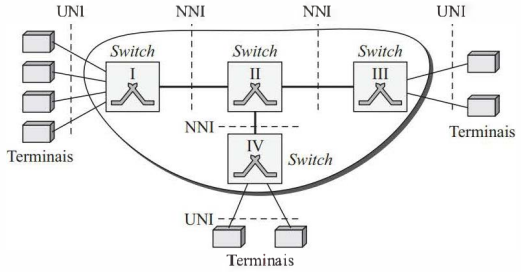


Figura 5.77 Arquitetura de uma rede ATM.

Conexão virtual

A conexão entre dois terminais é realizada por meio de caminhos de transmissão, caminhos virtuais e circuitos virtuais. Um **Caminho de Transmissão** (TP – Transmission Path) é uma conexão física (fios, cabo, satélite, e assim por diante) entre um terminal e um *switch* ou entre dois *switches*. Imagine os dois *switches* como duas cidades. Um caminho de transmissão pode consistir no conjunto de todas as rodovias que conectam diretamente as duas cidades.

Um caminho de transmissão é dividido em diversos caminhos virtuais. Um **Caminho Virtual** (VP – Virtual Path) fornece uma conexão ou um conjunto de conexões entre dois *switches*. Um caminho virtual pode ser imaginado como uma estrada que liga duas cidades. Cada estrada é um caminho virtual; o conjunto de todas as estradas é o caminho de transmissão.

Redes de células são baseadas em **Circuitos Virtuais** (VCs – Virtual Circuits). Todas as células pertencentes a uma única mensagem seguem o mesmo circuito virtual e conservam a sua ordenação

original até chegarem ao destino. O circuito virtual pode ser imaginado como as pistas de uma rodovia (caminho virtual). A Figura 5.78 mostra a relação entre um caminho de transmissão (uma ligação física), caminhos virtuais (uma combinação de circuitos virtuais que são agrupados porque partes de seus caminhos coincidem), e circuitos virtuais que conectam logicamente dois pontos.

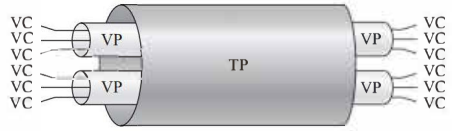
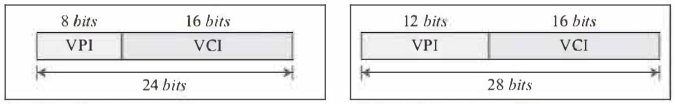


Figura 5.78 TP, VPs e VCs.

Identificadores Em uma rede de circuitos virtuais, para que seja possível rotear os dados de um terminal até outro, as conexões virtuais precisam ser identificados. Com esse objetivo, os projetistas do ATM criaram um identificador hierárquico com dois níveis: um Identificador de Caminho Virtual (VPI – Virtual-Path Identifier) e um Identificador de Circuito Virtual (VCI – Virtual-Circuit Identifier). O VPI define o VP específico e o VCI define um VC em particular dentro do VP. O VPI é o mesmo para todas as conexões virtuais agrupadas (logicamente) em um VP.

Os comprimentos dos VPIs para as UNIs e para as NNIS são diferentes. Em uma UNI, o VPI tem 8 *bits*, enquanto em uma NNI, o VPI tem 12 *bits*. O comprimento do VCI é o mesmo em ambas as interfaces (16 *bits*). Podemos, portanto, dizer que uma conexão virtual é identificada por 24 *bits* em uma UNI e por 28 *bits* em uma NNI, conforme mostra a Figura 5.79.

A ideia por trás de dividir um identificador de circuito virtual em duas partes é permitir o roteamento hierárquico. A maioria dos *switches* em uma rede ATM típica faz o roteamento usando VPIs. Os *switches* na borda da rede, que interagem diretamente com os dispositivos terminais, usam tanto os VPIs como os VCIs.



(*a*) VPI e VCI em uma UNI (*b*) VPI e VCI em uma NNI

Figura 5.79 Identificadores de conexão virtual em UNIs e NNIs.

Células A unidade básica de dados em uma rede ATM é denominada uma célula. Uma célula tem apenas 53 *bytes* de comprimento, com 5 *bytes* atribuídos ao cabeçalho e 48 *bytes* transportando a carga útil (os dados do usuário podem ser menores do que 48 *bytes*). A maior parte do cabeçalho é ocupada pelo VPI e pelo VCI, que definem a conexão virtual pela qual uma célula deve viajar de um terminal até um *switch* ou de um *switch* até outro. A Figura 5.80 mostra a estrutura das células.

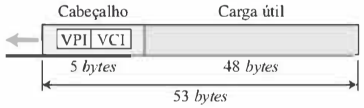


Figura 5.80 Uma célula ATM.

Estabelecimento e encerramento de conexões

O ATM usa dois tipos de conexões: PVC e SVC.

PVC No Circuito Virtual Permanente (PVC – Permanent Virtual-Circuit) é estabelecida entre dois terminais pelo provedor da rede. Os VPIs e os VCIs são definidos para as conexões permanentes, e os valores são inseridos nas tabelas de cada *switch*.

SVC No Circuito Virtual Comutado (SVC – Switched Virtual-Circuit), cada vez que um terminal deseja se conectar a outro, um novo circuito virtual deve ser estabelecido. O ATM não pode realizar essa tarefa sozinho. Ele precisa dos endereços da camada de rede e dos serviços de outro protocolo (tal como o IP). O mecanismo de sinalização desse outro protocolo cria uma solicitação de conexão usando os endereços da camada de rede dos dois terminais. O mecanismo exato depende do protocolo da camada de rede.

Comutação

O ATM usa *switches* para rotear as células de um terminal de origem ao terminal de destino. Um *switch* encaminha a célula usando tanto os VPIs como os VCIs. O encaminhamento requer o identificador completo. A Figura 5.81 mostra como um *switch* de um PVC encaminha as células. Uma célula cujo VPI é 153 e cujo VCI é 67 chega à interface (porta) 1 do *switch*. O *switch* verifica sua tabela de comutação, que contém seis informações por linha: número da interface de chegada, VPI de entrada, VCI de entrada, número da interface de saída correspondente, o novo VPI e o novo VCI. O *switch* encontra a linha da tabela com interface 1, VPI 153 e VCI 67, e então descobre que essa combinação corresponde à interface de saída 3, com VPI 140 e VCI 92. Ele altera os valores de VPI e VCI no cabeçalho para 140 e 92, respectivamente, e envia a célula pela interface 3.

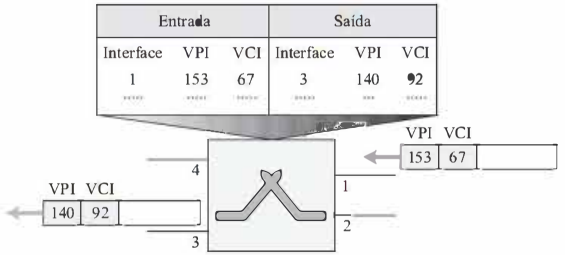


Figura 5.81 Roteamento com um *switch*.

Camadas ATM

O padrão ATM define três camadas. Elas são, de cima para baixo, a camada de adaptação ATM, a camada ATM e a camada física, conforme mostra a Figura 5.82. Os terminais usam as três camadas, enquanto os *switches* usam apenas as duas camadas inferiores.

Camada AAL **A Camada de Adaptação ATM** (AAL – ATM Adaptation Layer) foi projetada para dar suporte a dois conceitos do ATM. Primeiro, o ATM deve aceitar qualquer tipo de carga útil, seja ela um quadro de dados ou um fluxo de *bits*. Um quadro de dados pode vir de um protocolo da camada superior que cria quadros claramente delimitados para serem enviados para uma rede de transporte, tal como o ATM. Um bom exemplo é a Internet. O ATM também deve ser capaz de transportar dados multimídia. Ele pode aceitar fluxos contínuos de *bits* e quebrá-los em grupos para serem encapsulados em uma célula na camada ATM. A camada AAL usa duas subcamadas para realizar essas tarefas.

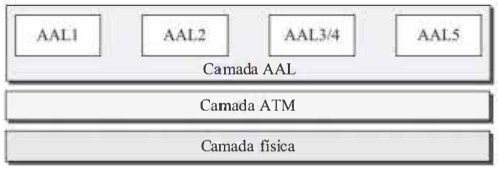


Figura 5.82 Camadas ATM.

Sejam os dados um quadro de dados ou um fluxo de *bits*, a carga útil deve ser dividida em segmentos de 48 *bytes* para serem transportados por uma célula. No destino, esses segmentos precisam ser remontados para recriar a carga útil original. A camada AAL define uma subcamada, conhecida como subcamada **Segmentação e Remontagem** (SAR – Segmentation And Reassembly), para fazer isso. A segmentação é feita na origem, enquanto a remontagem acontece no destino.

Antes de os dados serem segmentados pela subcamada SAR, eles devem ser preparados para garantir sua integridade. Isso é feito por uma subcamada denominada **Subcamada de Convergência** (CS – Convergence Sublayer). O ATM define quatro versões da camada AAL: *AAL1*, *AAL2*, *AAL3/4* e *AAL5*. Discutimos aqui apenas a AAL5, que é utilizada na Internet atual.

A subcamada ALL5 foi projetada para aplicações da Internet. Ela também é conhecida como **Camada de Adaptação Simples e Eficiente** (SEAL – Simple and Efficient Adaptation Layer). A AAL5 assume que todas as células que pertencem a uma única mensagem viajam sequencialmente e que as funções de controle são incluídas nas camadas superiores do aplicativo que está enviando os dados. A Figura 5.83 mostra a subcamada AAL5.

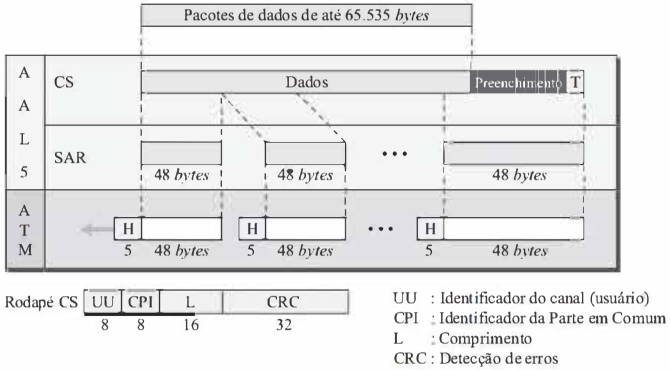


Figura 5.83 AAL5

O pacote na camada CS usa um rodapé com quatro campos. O campo UU é o identificador usuário a usuário. O IPC é o identificador da parte em comum. O campo C define o comprimento dos dados originais. O campo CRC é um campo de dois *bytes* usado para verificação de erros nos dados completos.

Camada ATM A *camada ATM* fornece serviços de roteamento, gerenciamento de tráfego, comutação e multiplexação. Ela processa o tráfego de saída, aceitando segmentos de 48 *bytes* das subcamadas AAL e transformando-os em células de 53 *bytes* por meio da adição de um cabeçalho de 5 *bytes*.

Camada física Como na Ethernet e nas LANs sem fio, as células ATM podem ser transportadas por qualquer tipo de camada física.

Controle de congestionamento e qualidade de serviço

O ATM inclui mecanismos de controle de congestionamento e qualidade de serviço bastante avançados.

5.7 DISPOSITIVOS DE CONEXÃO

Estações e redes normalmente não operam isoladamente. Usamos dispositivos de conexão para interligar estações de modo a criar uma rede ou para interligar redes com o objetivo de criar uma internet. Os dispositivos de conexão podem operar em diferentes camadas do modelo Internet. Discutiremos três tipos de dispositivos de conexão: repetidores e *hubs*, *switches* da camada de enlace (ou comutadores da camada 2) e roteadores (ou comutadores da camada 3). Os repetidores e os *hubs* operam na primeira camada do modelo Internet. *Switches* da camada de enlace operam nas duas primeiras camadas. Roteadores operam nas três primeiras camadas.

5.7.1 Repetidores e *hubs*

Um **repetidor** é um dispositivo que opera apenas na camada física. Os sinais que transportam informações dentro de uma rede são capazes de percorrer uma distância fixa antes que a atenuação ponha em perigo a integridade dos dados. Um repetidor recebe um sinal e, antes que ele fique muito fraco, ou seja, corrompido, tal dispositivo *regenera* e *retemporiza* a sequência original de *bits*. Em seguida, o repetidor envia o sinal restaurado. No passado, quando LANs Ethernet usavam uma topologia em barramento, repetidores eram usados para conectar dois segmentos de uma LAN, de modo a superar as restrições de comprimento do cabo coaxial. Atualmente, entretanto, LANs Ethernet usam topologias em estrela. Em uma topologia em estrela, um repetidor é um dispositivo de múltiplas portas, geralmente conhecido como *hub*, que pode servir como ponto de conexão e, ao mesmo tempo, funcionar como um repetidor. A Figura 5.84 mostra que, quando um pacote que vai da estação A para a estação B chega ao *hub*, o sinal que representa o quadro é regenerado para remover qualquer ruído que possa corromper os dados, mas o *hub* encaminha o pacote por todas as portas de saída, exceto por aquela na qual o sinal foi recebido. Em outras palavras, o quadro é transmitido via *broadcast*. Todas as estações na LAN recebem o quadro, mas apenas a estação B o conserva. As outras estações o descartam. A Figura 5.84 mostra o papel de um *hub* em uma LAN comutada.

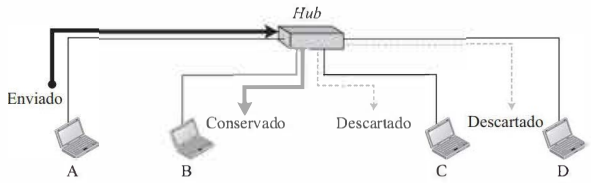


Figura 5.84 Repetidor ou *hub*.

A figura mostra definitivamente que um *hub* não apresenta qualquer capacidade de filtragem; ele não tem a inteligência necessária para determinar a porta pela qual o quadro deve sair.

Um *hub* não apresenta funcionalidades de filtragem.

Hubs e repetidores são dispositivos da camada física. Eles não apresentam um endereço da camada de enlace e não verificam o endereço da camada de enlace presente no quadro recebido. Apenas regeneram os *bits* corrompidos e enviam os quadros pelas portas de saída.

5.7.2 *Switches*

Um *switch da camada de enlace*, ou simplesmente *switch*, opera nas camadas física e de enlace de dados. Como um dispositivo da camada física, ele regenera o sinal recebido. Como um dispositivo da camada de enlace, ele pode verificar os endereços MAC (de origem e de destino) contidos no quadro.

Filtragem

Podemos nos perguntar qual é a diferença em termos de funcionalidade entre um *switch* e um *hub*. Um *switch* possui capacidade de *filtragem*; pode verificar o endereço de destino de um quadro e decidir por qual porta de saída ele deve ser enviado.

Um *switch* possui uma tabela usada em decisões de filtragem.

Vamos dar um exemplo. Na Figura 5.85, temos uma LAN com quatro estações conectadas a um *switch*. Se um quadro destinado à estação de endereço 71:2B:13:45:61:42 chegar à porta 1 do *switch*, ele consulta sua tabela para determinar a porta de saída.

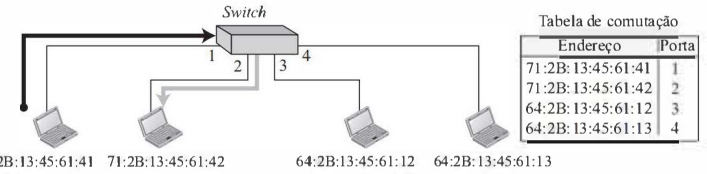


Tabela de comutação

Endereço	Porta
71:2B:13:45:61:41	1
71:2B:13:45:61:42	2
64:2B:13:45:61:12	3
64:2B:13:45:61:13	4

Figura 5.85 *Switch*

De acordo com a tabela, os quadros destinados a 71:2B:13:45:61:42 devem ser enviados somente pela porta 2 e, portanto, não há necessidade de transmitir o quadro às outras portas.

Um *switch* não altera os endereços da camada de enlace (MAC) em um quadro.

Switches transparentes

Um *switch transparente* é um *switch* cujas estações a ele conectadas não tomam ciência de sua existência. Se um *switch* é adicionado ou removido do sistema, não é necessário reconfigurar as estações. De acordo com a especificação do IEEE 802.1d, um sistema equipado com *switches* transparentes deve satisfazer três requisitos:

- Os quadros devem ser encaminhados de uma estação para outra.
- A tabela de encaminhamento é criada automaticamente usando informações sobre a movimentação de quadro na rede (aprendizado).
- Laços devem ser evitados no sistema.

Encaminhamento

Um *switch* transparente deve encaminhar corretamente os quadros, conforme discutido na seção anterior.

Aprendizado

Os primeiros *switches* inventados apresentavam tabelas de encaminhamento estáticas. O administrador do sistema precisava inserir manualmente cada entrada na tabela durante a configuração do *switch*. Embora o processo fosse simples, ele não era prático. Se uma estação fosse adicionada ou removida, a tabela precisava ser modificada manualmente. O mesmo acontecia se o endereço MAC de uma estação mudasse, o que não é um evento raro. Por exemplo, colocar uma nova interface de rede significa ter um novo endereço MAC.

Uma solução mais adequada é substituir a tabela estática por uma tabela dinâmica que mapeia endereços das portas (interfaces) automaticamente. Para criar uma tabela dinâmica, precisamos de um *switch* que aprenda gradualmente com as informações de movimentação dos quadros. Para isso, o *switch* inspeciona os endereços de destino e de origem do quadro. O endereço de destino é usado para a decisão de encaminhamento (busca na tabela); já o endereço de origem é usado para adicionar entradas na tabela e para fins de atualização. Detalharemos esse processo usando a Figura 5.86.

Construção gradual da tabela

Endereço	Porta

(*a*) Original

Endereço	Porta
71:2B:13:45:61:41	1

(*b*) Após A enviar um quadro para D

Endereço	Porta
71:2B:13:45:61:41	1
64:2B:13:45:61:13	4

(*c*) Após D enviar um quadro para B

Endereço	Porta
71:2B:13:45:61:41	1
64:2B:13:45:61:13	4
71:2B:13:45:61:42	2

(*d*) Após B enviar um quadro para A

Endereço	Porta
71:2B:13:45:61:41	1
64:2B:13:45:61:13	4
71:2B:13:45:61:42	2
64:2B:13:45:61:12	3

(*e*) Após C enviar um quadro para D

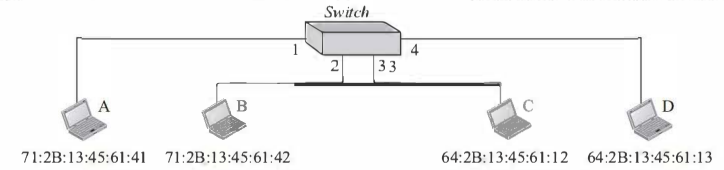


Figura 5.86 Aprendizado no *switch*.

1. Quando a estação A envia um quadro para a estação D, o *switch* não possui entradas nem para D nem para A. O quadro é enviado por todas as três portas, ou seja, inunda a rede. Entretanto, verificando o endereço de origem, o *switch* aprende que a estação A deve estar conectada à porta 1. Isto significa que quadros destinados a A, no futuro, devem sair pela porta 1. O *switch* adiciona esta entrada à sua tabela, que agora tem a sua primeira entrada.
2. Quando a estação D envia um quadro para a estação B, o *switch* não tem uma entrada para B, de modo que ele inunda a rede novamente. No entanto, adiciona uma entrada a mais à tabela, relacionada à estação D.
3. O processo de aprendizagem continua até que a tabela apresente informações sobre todas as portas.

No entanto, note que o processo de aprendizagem pode levar um longo tempo. Por exemplo, se uma estação não enviar quadros (uma situação rara), a estação nunca terá uma entrada na tabela.

5.7.3 Roteadores

Discutimos sobre roteadores no Capítulo 4. Neste capítulo, mencionamos os roteadores com o objetivo de compará-los com um *switch* e um *hub*. Um *roteador* é um dispositivo com três camadas; ele opera nas camadas: física, de enlace de dados e de rede. Como um dispositivo de camada física, ele regenera o sinal que recebe. Como um dispositivo da camada de enlace, ele verifica os endereços físicos (de origem e de destino) contidos no pacote. Como um dispositivo da camada de rede, um roteador verifica os endereços da camada de rede.

Um roteador é um dispositivo de três camadas (física, de enlace de dados e de rede).

Um roteador pode conectar redes. Em outras palavras, um roteador é um dispositivo de interconexão; conecta redes independentes para criar uma internet. De acordo com essa definição, duas redes conectadas por um roteador constituem uma internet.

Existem três grandes diferenças entre um roteador e um *hub* ou um *switch*.

1. Um roteador tem um endereço físico e um endereço lógico (IP) para cada uma de suas interfaces.
2. Um roteador atua apenas sobre os pacotes nos quais o endereço da camada de enlace de destino corresponde ao endereço da interface na qual o pacote chega.
3. Um roteador altera o endereço da camada de enlace do pacote (tanto de origem como de destino), quando ele encaminha o pacote.

Usemos a Figura 5.87 como exemplo: considere uma organização que tem dois edifícios separados com uma LAN Ethernet Gigabit instalada em cada prédio. A organização usa *switches* em cada LAN. As duas LANs podem ser conectadas para formar uma grande LAN usando a tecnologia Ethernet 10-Gigabit, acelerando a conexão com a Ethernet e a conexão com o servidor da organização. Um roteador pode, então, conectar o sistema todo à Internet.

Um roteador, como vimos no Capítulo 4, alterará o endereço MAC que ele recebe porque os endereços MAC têm jurisdição local apenas.

Um roteador altera os endereços da camada de enlace de um pacote.

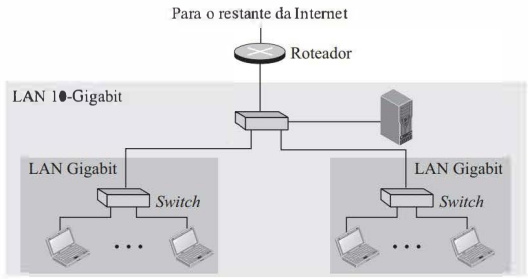


Figura 5.87 Exemplo de roteamento.

5.8 MATERIAL DO FINAL DO CAPÍTULO

5.8.1 Leitura recomendada

Para obter mais detalhes sobre os assuntos discutidos neste capítulo, recomendamos os seguintes livros. Os itens entre colchetes [...] referem-se à lista de referências no fim do texto.

Books

Diversos livros excelentes discutem as questões da camada de enlace. Entre eles, recomendamos a leitura de [Ham 80], [Zar 02], [Ror 96], [Tan 03], [GW 02], [For 03], [KMK 04], [Sta 04], [Kes 02], [PD 03], [Kei 02], [Spu 00], [KCK 98], [Sau 98], [Izz 00], [Per 00] e [WV 00].

RFCs

Uma discussão sobre o uso da soma de verificação na Internet pode ser encontrada na RFC 1141.

5.8.2 Termos-chave

- acesso aleatório
- acesso controlado
- Acesso Múltiplo com Detecção de Portadora (CSMA – Carrier Sense Multiple Access)
- Acesso Múltiplo com Detecção de Portadora / Detecção de Colisão (CSMA/CD – Carrier Sense Multiple Access/Collision Detection)
- Acesso Múltiplo com Detecção de Portadora / Prevenção de Colisão (CSMA/CA – Carrier Sense Multiple Access/Collision Avoidance)
- ALOHA
- ALOHA puro
- Antena Comunitária de TV (CATV – Community Antenna TV)
- Camada de Adaptação ATM (AAL – ATM Adaptation Layer)
- Camada de Adaptação Simples e Eficiente (SEAL – Simple and Efficient Adaptation Layer)
- Caminho de Transmissão (TP – Transmission Path)
- Caminho Virtual (VP – Virtual Path)
- canalização
- Circuito Virtual (VC – Virtual Circuit)
- código de paridade
- complemento de um
- contenção
- Controle de Acesso ao Meio (MAC – Media Access Control)

- Controle de Enlace de Dados (DLC – Data Link Control)
- Controle de Enlace de Dados de Alto Nível (HDLC – High-level Data Link Control)
- Controle de Enlace Lógico (LLC – Logical Link Control)
- demodulador
- distância de Hamming
- erro de um único *bit*
- erro em rajada
- Ethernet 10-Gigabit
- Ethernet Gigabit
- Ethernet Padrão
- Fast Ethernet
- *hub*
- Interface Rede-Rede (NNI – Network-to-Network Interface)
- Interface Usuário-Rede (UNI – User-to-Network Interface)
- Linha de Assinante Digital (DSL – Digital Subscriber Line)
- modem
- Modo de Transferência Assíncrona (ATM – Asynchronous Transfer Mode)
- Modo Balanceado Assíncrono (ABM – Asynchronous Balanced Mode)
- modulador
- Multiplexação por Divisão de Tempo (TDM – Time-Division Multiplexing)
- palavra de dados
- palavra de código
- Passagem de Ficha (*Token Passing*)
- Placa de Interface de Rede (NIC – Network Interface Card)
- Portadora Óptica (OC – Optical Carrier)
- preenchimento de *bits*
- preenchimento de *byte*
- Projeto 802
- Protocolo de Autenticação por Desafio-Resposta (CHAP – Challenge Handshake Authentication Protocol)
- Protocolo de Controle de Enlace (LCP – Link Control Protocol)
- Protocolo de Controle de Rede da Internet (IPCP – Internet Protocol Control Protocol)
- Protocolo Ponto a Ponto (PPP – Point-to-Point Protocol)
- Protocolo de Resolução de Endereços (ARP – Address Resolution Protocol)
- Rede Híbrida Fibra-Coaxial (HFC – Hybrid Fiber-Coaxial)
- Rede Local Virtual (VLAN – Virtual Local Area Network)
- Rede Óptica Síncrona (SONET – Synchronous Optical Network)
- Segmentação e Remontagem (SAR – Segmentation And Reassembly)
- Sinal de Transporte Síncrono (STS – Synchronous Transport Signal)
- síndrome
- *slotted* ALOHA
- Subcamada de Convergência (CS – Convergence Sublayer)
- *switch*
- Tributários Virtuais (VT – Virtual Tributaries)
- varredura (*polling*)
- Verificação de Redundância Cíclica (CRC – Cyclic Redundancy Check)

5.8.3 Resumo

Podemos considerar a camada de enlace de dados como sendo duas subcamadas. A subcamada superior é responsável pelo controle do enlace de dados, enquanto a inferior é responsável por resolver o acesso ao meio compartilhado. A subcamada de Controle de Enlace de Dados (DLC – Data Link Control) lida com o projeto e os procedimentos para a comunicação entre dois nós adjacentes: comunicação nó a nó. Essa subcamada é responsável por realizar o enquadramento e controlar erros. O controle de erros consiste em mecanismos para lidar com a corrupção dos dados durante a transmissão. Discutimos dois protocolos da camada de enlace neste capítulo: o HDLC e o PPP. O Controle de Enlace de Dados de Alto Nível (HDLC – High-level Data Link Control) é um protocolo orientado a *bits* usado para comunicação por meio de enlaces ponto a ponto e multiponto.

Entretanto, o protocolo mais comum para acesso ponto a ponto é o Protocolo Ponto a Ponto (PPP – Point-to-Point Protocol ou), que é um protocolo orientado a *bytes*.

Muitos protocolos formais foram criados para lidar com o acesso a um enlace compartilhado. Podemos categorizá-los em três grupos: protocolos de acesso aleatório, de acesso controlado e de canalização. Nos métodos de acesso aleatório ou de contenção, nenhuma estação tem prioridade ou controle sobre outra estação. No acesso controlado, as estações devem consultar umas às outras para descobrir qual estação tem direito de enviar dados. A canalização é um método de acesso múltiplo no qual a largura de banda disponível em um enlace é compartilhada no tempo, na frequência ou por meio de códigos, entre diferentes estações.

Na camada de enlace de dados, usamos endereçamento da camada de enlace. O sistema normalmente determina o endereço da camada de enlace do próximo nó usando o Protocolo de Resolução de Endereços (ARP – Address Resolution Protocol).

A Ethernet é o protocolo de rede local mais utilizado atualmente. A camada de enlace de dados da Ethernet consiste na subcamada LLC e na subcamada MAC. A subcamada MAC é responsável pela operação do método de acesso CSMA/CD e pelo enquadramento. Cada estação em uma rede Ethernet tem um endereço de 48 *bits* único impresso em sua placa de interface de rede (NIC – Network Interface Card). Uma Rede Local Virtual (VLAN – Virtual Local Area Network) é configurada via *software*, não por meio do cabeamento físico. A associação de máquinas a uma VLAN pode ser baseada em números de portas, endereços MAC, endereços IP, endereços IP *multicast*, ou uma combinação dessas características. VLANs são eficientes em termos de custo e tempo, podem reduzir o tráfego na rede, e podem ser usadas como uma medida adicional de segurança.

A necessidade de transmitir dados digitais levou à invenção do modem discado. Devido à necessidade de alta velocidade de recepção e envio de dados, as empresas de telefonia introduziram uma nova tecnologia, a *Linha de Assinante Digital* (DSL – Digital Subscriber Line). As redes a cabo foram originalmente criadas para proporcionar melhor acesso a programas de TV. As empresas de TV a cabo também descobriram um mercado promissor no fornecimento de acesso à Internet, usando alguns dos canais originalmente projetados para vídeo. O padrão Rede Óptica Síncrona (SONET – Synchronous Optical Network) foi desenvolvido pela ANSI para redes de fibra óptica. O Modo de Transferência Assíncrona (ATM – Asynchronous Transfer Mode) é um protocolo de comutação de células que, em combinação com a tecnologia SONET, permite conexões de alta velocidade. Uma célula é um pequeno bloco de informação de tamanho fixo. O pacote de dados do ATM é uma célula composta por 53 *bytes*. O padrão ATM define três camadas: a camada Camada de Adaptação ATM (AAL – ATM Adaptation Layer), a camada ATM e a camada física.

Um *hub* é um dispositivo de conexão que opera na camada física do modelo Internet. Um *switch* é um dispositivo de conexão que opera nas camadas físicas e de enlace de dados do modelo Internet. Um *switch* transparente pode encaminhar e filtrar quadros, construindo sua tabela de encaminhamento de forma automática. Um roteador é um dispositivo de conexão que opera nas três primeiras camadas da pilha de protocolos TCP/IP.

5.9 ATIVIDADES PRÁTICAS

5.9.1 Testes

Diversos testes interativos relativos a este capítulo podem ser encontrados no *site* www.mhhe.com/forouzan. Recomendamos que o estudante realize os testes para verificar o seu entendimento do material apresentado antes de continuar com as atividades práticas.

Questões

5-1 Descreva as diferenças entre a comunicação na camada de rede e a comunicação na camada de enlace de dados.

5-2 Descreva as diferenças entre um enlace ponto a ponto e um enlace de *broadcast*.

5-3 Explique por que é necessário utilizar marcadores quando usamos quadros de tamanho variável.

5-4 Explique por que não podemos aplicar preenchimento de *bits* no contexto de enquadramento orientado a caracteres para alterar um *byte* usado como marcador aparecendo no texto.

5-5 Quais as diferenças entre um erro de um único *bit* e um erro em rajada?

5-6 Qual é a definição de um código de bloco linear?

5-7 Em um código de bloco, uma palavra de dados tem 20 *bits* e a palavra de código correspondente tem 25 *bits*. Quais são os valores de *k*, *r* e *n* de acordo com as definições dadas no texto? Quantos *bits* redundantes são adicionados a cada palavra de dados?

5-8 Em uma palavra de código, acrescentamos dois *bits* redundantes em cada palavra de dados de oito *bits*. Determine o número de

a. palavras de código válidas.

b. palavras de código inválidas.

5-9 O que significa a distância de Hamming mínima no contexto de códigos corretores de erros?

5-10 Se quisermos detectar erros de dois *bits*, qual deve ser a distância de Hamming mínima?

5-11 A classe de código para detecção (e correção) de erros conhecida como código de Hamming consiste em códigos nos quais $d_{min} = 3$. Esse código pode detectar até dois erros (ou corrigir um único erro). Nele, os valores de *n*, *k* e *r* satisfazem as relações: $n = 2^r - 1$ e $k = n - r$. Encontre o número de *bits* nas palavras de dados e nas palavras de código se $r = 3$.

5-12 No cálculo do CRC, se a palavra de dados tem 5 *bits* e a palavra de código tem 8 *bits*, quantos 0s precisam ser adicionados à palavra de dados para gerar o dividendo? Qual é o tamanho do resto? Qual é o tamanho do divisor?

5-13 No cálculo do CRC, quais dos seguintes geradores (divisores) garantem a detecção de um erro de um único *bit*?

a. 101 **b.** 100 **c.** 1

5-14 No cálculo do CRC, quais dos seguintes geradores (divisores) garantem a detecção de um número ímpar de erros?

a. 10111 **b.** 101101 **c.** 111

5-15 No cálculo do CRC, escolhemos o gerador 1100101. Qual é a probabilidade de detectarmos um erro em rajada apresentando cada um dos seguintes comprimentos?

a. 5 **b.** 7 **c.** 10

5-16 Considere que estamos enviando conjuntos de dados de 16 *bits* de comprimento. Se dois conjuntos de dados forem trocados durante a transmissão, a soma de verificação (*checksum*) tradicional pode detectar esse erro? Explique.

5-17 O valor de uma soma de verificação (*checksum*) tradicional pode ser composto apenas por 0s (em binário)? Explique sua resposta.

5-18 Explique como o algoritmo de Fletcher (Figura 5.18) atribui pesos aos conjuntos de dados durante o cálculo da soma de verificação (*checksum*).

5-19 Explique por que há apenas um campo de endereço (e não dois) em um quadro HDLC.

5-20 Quais das seguintes opções são protocolos de acesso aleatório?

a. CSMA/CD

b. Varredura

c. TDMA

5-21 As estações em uma rede usando ALOHA puro enviam quadros de 1.000 *bits* de comprimento a uma taxa de transferência de 1 Mbps. Qual é o período vulnerável para essa rede?

5-22 Em uma rede usando ALOHA puro com G = 1/2, como a taxa de transferência é afetada em cada um dos seguintes casos?

a. G é aumentado para 1.

b. G é reduzido para 1/4.

5-23 Para entender a utilidade do parâmetro K na Figura 5.40, determine a probabilidade de que uma estação seja capaz de transmitir imediatamente em cada um dos seguintes casos:

a. Após uma falha.

b. Após três falhas.

5-24 Com base na Figura 5.30, como interpretamos a situação de *sucesso* em uma rede ALOHA?

5-25 Considere que o atraso de propagação em uma rede cujo meio é compartilhado seja de 5 µs e que o tempo de transmissão do quadro seja de 10 µs.

a. Quanto tempo leva para que o primeiro *bit* chegue ao destino?

b. Após a chegada do primeiro *bit*, quanto tempo leva para que o último *bit* chegue ao destino?

c. Quanto tempo a rede fica ocupada com esse quadro (vulnerável a colisões)?

5-26 Considere que o atraso de propagação em uma rede cujo meio é compartilhado seja de 3 µs e que o tempo de transmissão do quadro seja de 5 µs. A colisão pode ser detectada independentemente de onde ela ocorrer?

5-27 Duas estações em duas redes diferentes podem ter o mesmo endereço da camada de enlace?

5-28 Explique por que a colisão é um problema em protocolos de acesso aleatório, mas não o é em protocolos de acesso controlado ou de canalização.

5-29 Precisamos de um protocolo de acesso múltiplo quando usamos o serviço DSL fornecido pela companhia telefônica para acessar a Internet? Por quê?

5-30 O tamanho do pacote ARP é fixo? Explique.

5-31 Qual o tamanho de um pacote ARP quando o protocolo de rede é o IPv4 e o *hardware* é do tipo Ethernet?

5-32 Qual é o tamanho de um quadro Ethernet transportando um pacote ARP cujo tamanho é 28 *bytes*?

5-33 Em um quadro Ethernet, quais as diferenças entre o campo de preâmbulo e o campo SFD?

5-34 Para que serve uma placa de rede (NIC)?

5-35 Por que não é necessário usar CSMA/CD em uma LAN Ethernet *full-duplex*?

5-36 Compare as taxas de transferência de dados da Ethernet Padrão, Fast Ethernet, Ethernet Gigabit e Ethernet 10-Gigabit.

5-37 Quais são as implementações comuns da Ethernet Padrão?

5-38 Quais são as implementações comuns da Ethernet Gigabit?

5-39 O que é a tecnologia de modem discado? Liste alguns dos padrões comuns de modem discutidos neste capítulo e forneça suas taxas de transferência de dados.

5-40 Compare uma rede a cabo tradicional com uma rede híbrida fibra-coaxial, explicando suas diferenças e semelhanças.

5-41 Quatro estações são conectadas a um *hub* em uma rede Ethernet. As distâncias entre o *hub* e as estações são de 300 m, 400 m, 500 m e 700 m, respectivamente. Qual é o comprimento da rede que devemos considerar para o cálculo de T_p?

5-42 O que significa dizer que um *switch* de camada de enlace é capaz de filtrar o tráfego? Por que a filtragem é importante?

5-43 Como uma VLAN pode levar a uma economia de tempo e dinheiro?

5-44 Como uma VLAN pode reduzir o tráfego na rede?

5-45 Por que uma rede SONET é considerada uma rede síncrona?

5-46 Discuta as funções de cada camada do SONET.

5-47 No ATM, qual é a relação entre TPs, VPs e VCs?

Problemas

5-1 Faça o preenchimento de *byte* da seguinte carga útil de um quadro, considerando que E é o *byte* de escape, F é o marcador de fim de quadro e D é um *byte* de dados diferente de um caractere de escape ou de um marcador de fim de quadro.

D	E	D	D	F	D	D	E	E	D	F	D

5-2 Faça o preenchimento de *byte* da seguinte carga útil de um quadro:

```
0001111111001111101000111111111110000111
```

5-3 Qual é número máximo de *bits* corrompidos por um ruído em rajada de 2 ms quando os

dados são transmitidos a cada uma das seguintes taxas de transferência?

a. 1.500 bps **c.** 100 kbps
b. 12 kbps **d.** 100 Mbps

5-4 Considere que p seja a probabilidade de que um *bit* em uma unidade de dados seja corrompido durante a transmissão. Para uma unidade de dados de n *bits*, determine a probabilidade de que x *bits* sejam corrompidos em cada um dos casos a seguir.

a. $n = 8, x = 1, p = 0{,}2$
b. $n = 16, x = 3, p = 0{,}3$
c. $n = 32, x = 10, p = 0{,}4$

5-5 A operação de XOR (OU-Exclusivo) é uma das operações mais utilizadas no cálculo de palavras de código. Aplique a operação de XOR sobre as duas sequências de *bits* a seguir. Interprete os resultados.

a. $(10001) \oplus (10001)$
b. $(11100) \oplus (00000)$
c. $(10011) \oplus (11111)$

5-6 Na Tabela 5.1, o remetente envia a palavra de dados 10. Um erro em rajada de comprimento igual a 3 bits corrompe a palavra de código. O receptor é capaz de detectar o erro? Explique sua resposta.

5-7 Usando o código da Tabela 5.2, qual é a palavra de dados correspondente a cada uma das seguintes palavras de código recebidas?

a. 01011 **b.** 11111 **c.** 00000 **d.** 11011

5-8 Prove que o código representado pelas palavras de código a seguir não é linear. Para isso, basta que você encontre um caso que viole a linearidade.

{(00000), (01011), (10111), (11111)}

5-9 Qual é a distância de Hamming para cada um dos seguintes pares de palavras de código?

a. d (10000, 00000)
b. d (10101, 10000)
c. d (00000, 11111)
d. d (00000, 00000)

5-10 Embora seja possível provar formalmente que o código da Tabela 5.3 é tanto linear como cíclico, utilize os dois testes a seguir para provar parcialmente esse fato:

a. Teste a propriedade cíclica sobre a palavra de código 0101100.
b. Teste a propriedade linear sobre as palavras de código 0010110 e 1111111.

5-11 Com relação ao CRC-8 mostrado na Tabela 5.4, responda às seguintes perguntas:

a. Ele é capaz de detectar um erro de um único *bit*? Explique sua resposta.
b. Ele é capaz de detectar um erro em rajada de tamanho 6? Explique sua resposta.
c. Qual é a probabilidade de ele detectar um erro em rajada de tamanho 9?
d. Qual é a probabilidade de ele detectar um erro em rajada de tamanho 15?

5-12 Considerando que estamos usando paridade par, determine o *bit* de paridade para cada um dos seguintes conjuntos de dados.

a. 1001011 **c.** 1000000
b. 0001100 **d.** 1110111

5-13 Um simples *bit* de verificação de paridade, que é normalmente inserido no final da palavra (transformando um caractere ASCII de 7 *bits* em um *byte*), não é capaz de detectar um número par de erros. Por exemplo, erros em dois, quatro, seis ou oito *bits* não podem ser detectados usando essa técnica. Uma solução melhor é organizar os caracteres em uma tabela e criar paridades de linhas e colunas. Os *bits* de paridade das linhas são enviados juntamente com os *bytes*, enquanto os *bits* de paridade das colunas são enviados como um *byte* extra (Figura 5.88).

	C1	C2	C3	C4	C5	C6	C7	
R1	1	1	0	0	1	1	1	1
R2	1	0	1	1	1	0	1	1
R3	0	1	1	1	0	0	1	0
R4	0	1	0	1	0	0	1	1
	0	1	0	1	0	1	0	1

Paridades das linhas

Paridades das colunas

Rn: Linha *n*
Cm: Coluna *m*

Figura 5.88 Esquema para o Problema 5-13.

Mostre como os seguintes erros podem ser detectados:

a. Um erro na posição (L3, C3).
b. Dois erros nas posições (L3, C4) e (L3, C6).
c. Três erros nas posições (L2, C4), (L2, C5) e (R3, C4).

d. Quatro erros nas posições (L1, C2), (L1, C6), (L3, C2) e (L3, C6).

5-14 Dada a palavra de dados 101001111 e o divisor 10111, mostre como é feita a geração da palavra de código do CRC no lado do remetente (usando divisão binária).

5-15 Considere que um pacote é composto por apenas quatro palavras de 16 *bits*, $(A7A2)_{16}$, $(CABF)_{16}$, $(903A)_{16}$ e $(A123)_{16}$. Simule manualmente o algoritmo da Figura 5.17 para calcular a soma de verificação (*checksum*).

5-16 O cálculo tradicional da soma de verificação (*checksum*) precisa ser feito usando aritmética em complemento de um. Computadores e calculadoras atuais são projetados para realizar cálculos usando aritmética em complemento de dois. Uma forma de calcular a soma de verificação tradicional é somar os números usando aritmética em complemento de dois, determinar o quociente e o resto da divisão do resultado por 2^{16}, e então somar o quociente e o resto para obter a soma em complemento de um. A soma de verificação pode ser calculada subtraindo-se a soma do valor $2^{16} - 1$. Use esse método para calcular a soma de verificação dos quatro números a seguir: 43.689, 64.463, 45.112 e 59.683.

5-17 Esse problema mostra um caso especial na manipulação da soma de verificação. Um remetente tem dois conjuntos de dados para enviar: $(4567)_{16}$ e $(BA98)_{16}$. Qual é o valor da soma de verificação (*checksum*)?

5-18 Simule manualmente o algoritmo de Fletcher (Figura 5.18) para calcular a soma de verificação (*checksum*) dos *bytes* a seguir: $(2B)_{16}$, $(3F)_{16}$, $(6A)_{16}$ e $(AF)_{16}$. Mostre também que o resultado é uma soma de verificação ponderada.

5-19 Simule manualmente o algoritmo de Adler (Figura 5.19) para calcular a soma de verificação (*checksum*) das palavras a seguir: $(FBFF)_{16}$ e $(EFAA)_{16}$. Mostre também que o resultado é uma soma de verificação ponderada.

5-20 Como um exemplo prático, determine o valor do campo de soma de verificação (*checksum*) para o pequeno datagrama mostrado na Figura 5.89. Como a soma de verificação de um datagrama IP é calculado apenas para o cabeçalho, mostramos apenas os valores desses campos.

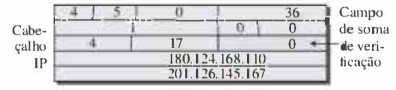


Figura 5.89 Esquema para o Problema 5-20.

5-21 Considere que um datagrama apresentando o cabeçalho da Figura 5.90 foi recebido. O valor da soma de verificação (*checksum*) é 65.207, conforme mostrado. Verifique se o cabeçalho foi corrompido durante a transmissão.

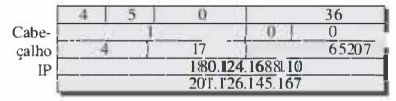


Figura 5.90 Esquema para o Problema 5-21.

5-22 Um exemplo de uma soma de verificação (*checksum*) ponderada é o código ISBN-10 que podemos encontrar na capa traseira de alguns livros. No ISBN-10, há nove dígitos decimais que definem o país, a editora e o livro. O décimo dígito (mais à direita) é um dígito de verificação. O código $D_1D_2D_3D_4D_5D_6D_7D_8D_9C$, satisfaz a seguinte propriedade.

$$[(10 \times D_1) + (9 \times D_2) + (8 \times D_3) + \ldots + (2 \times D_9) + (1 \times C)] \bmod 11 = 0$$

Em outras palavras, os pesos são 10, 9, ..., 1. Se o valor calculado para C for 10, a letra X é usada em seu lugar. Ao substituir cada peso w pelo seu complemento usando aritmética módulo 11 ($11 - w$), pode-se demonstrar que o dígito verificador pode ser calculado conforme mostrado a seguir.

$$C = [(1 \times D_1) + (2 \times D_2) + (3 \times D_3) + \ldots + (9 \times D_9)] \bmod 11$$

Calcule o dígito verificador para o seguinte ISBN-10: **0-07-296775-C.**

5-23 Um código ISBN-13, uma nova versão do ISBN-10, é outro exemplo de uma soma de verificação (*checksum*) ponderada com 13 dígitos, no qual há 12 dígitos decimais que definem o livro e o último dígito é o dígito verificador. O código, $D_1D_2D_3D_4D_5D_6D_7D_8D_9D_{11}D_{12}C$ satisfaz a seguinte propriedade.

$$[(1 \times D_1) + (3 \times D_2) + (1 \times D_3) + \ldots + (3 \times D_{12}) + (1 \times C)] \bmod 10 = 0$$

Em outras palavras, os pesos são 1 e 3, alternadamente. Usando essa descrição, calcule o dígito verificador para o seguinte ISBN-13: **978-0-07-296775-C**.

5-24 Para criar uma fórmula de avaliação do desempenho de uma rede de acesso múltiplo, precisamos de um modelo matemático. Quando o número de estações em uma rede é muito grande, a distribuição de Poisson, dada por $p[x] = (e^{-\lambda} \times \lambda^x)/(x!)$, é usada. Nessa fórmula, $p[x]$ é a probabilidade de se gerar um número x de quadros em um período de tempo e λ é o número médio de quadros gerados durante o mesmo período. Usando a distribuição de Poisson:

a. Determine a probabilidade de que uma rede usando ALOHA puro gere x quadros durante o período vulnerável. Lembre-se que o período vulnerável para essa rede é duas vezes o intervalo de transmissão de quadros (T_{fr}).

b. Determine a probabilidade de que uma rede usando *slotted* ALOHA gere x quadros durante o período vulnerável. Lembre-se que o período vulnerável para essa rede é igual ao intervalo de transmissão de quadros (T_{fr}).

5-25 No problema anterior, foi utilizada a distribuição de Poisson para calcular a probabilidade de serem gerados x quadros, em um determinado período de tempo, em redes usando o ALOHA puro e o *slotted* ALOHA. Essa probabilidade foi calculada como $p[x] = (e^{-\lambda} \times \lambda^x)/(x!)$. Nesse problema, queremos calcular a probabilidade de que um quadro em tais redes chegue ao seu destino sem colidir com outros quadros. Para isso, é mais simples considerar que temos G estações, cada uma enviando em média um quadro durante o intervalo de transmissão de quadros (em vez de haver N quadros e uma média de G/N quadros sendo enviados durante o mesmo intervalo). Dessa forma, a probabilidade de sucesso para uma estação equivale à probabilidade de que nenhuma outra estação envie um quadro durante o período vulnerável.

a. Determine a probabilidade de que uma estação em uma rede usando ALOHA puro tenha sucesso em enviar um quadro durante um período vulnerável.

b. Determine a probabilidade de que uma estação em uma rede usando *slotted* ALOHA tenha sucesso em enviar um quadro durante um período vulnerável.

5-26 No problema anterior, descobrimos que a probabilidade de uma estação (em uma rede com G estações) enviar um quadro com sucesso durante um intervalo de tempo vulnerável é $P = e^{-2G}$ para o ALOHA puro e $P = e^{-2G}$ para o *slotted* ALOHA. Nesse problema, queremos determinar a vazão dessas redes, que corresponde à probabilidade de que alguma estação (dentre G estações) consiga enviar um quadro durante o período vulnerável.

a. Determine a vazão de uma rede usando ALOHA puro.

b. Determine a vazão de uma rede usando *slotted* ALOHA.

5-27 No problema anterior, mostramos que a vazão é $V = Ge^{-2G}$ para uma rede usando ALOHA puro e é $V = Ge^{-G}$ para o *slotted* ALOHA. Nesse problema, queremos determinar o valor de G em cada rede que leva à vazão máxima da rede, e então determinar o valor dessa vazão máxima. Isto pode ser feito se calcularmos o valor de G que faz com que a derivada de V em relação a G seja zero.

a. Determine o valor de G que leva à vazão máxima, e então encontre o valor da vazão máxima para uma rede usando ALOHA puro.

b. Determine o valor de G que leva à vazão máxima, e então encontre o valor da vazão máxima para uma rede usando *slotted* ALOHA.

5-28 Uma rede de acesso múltiplo com um grande número de estações pode ser analisada usando a distribuição de Poisson. Quando há um número reduzido de estações em uma rede, entretanto, precisamos utilizar outra abordagem para essa análise. Em uma rede com N estações, consideramos que a probabilidade de que cada estação tenha um quadro para enviar durante o intervalo de transmissão de quadros (T_{fr}) seja p. Nessa rede, uma estação envia seu quadro com sucesso se a estação tiver um quadro para enviar durante o período vulnerável e nenhuma outra estação tiver um quadro para enviar durante esse espaço de tempo.

a. Determine a probabilidade de que uma estação em uma rede usando ALOHA puro consiga enviar um quadro durante o período vulnerável.

b. Determine a probabilidade de que uma estação em uma rede usando *slotted* ALOHA consiga enviar um quadro durante o período vulnerável.

5-29 No problema anterior, determinamos a probabilidade de sucesso de uma estação que tenta enviar um quadro durante o período vulnerável. A vazão de uma rede com um número limitado de estações equivale à probabilidade de que qualquer estação (dentre N estações) seja capaz de enviar um quadro com sucesso. Em outras palavras, a vazão é a soma das N probabilidades de sucesso.

a. Determine a vazão de uma rede usando ALOHA puro.

b. Determine a vazão de uma rede usando *slotted* ALOHA.

5-30 No problema anterior, determinamos a vazão de uma rede usando ALOHA puro e *slotted* ALOHA como sendo $V = Np\ (1 - p)^{2(N-1)}$ e $V = Np\ (1 - p)^{(N-1)}$, respectivamente. Nesse problema, queremos determinar a vazão máxima com relação a p.

a. Determine o valor de p que maximiza a vazão de uma rede usando ALOHA puro, e calcule a vazão máxima quando N é um número muito grande.

b. Determine o valor de p que maximiza a vazão de uma rede usando *slotted* ALOHA, e calcule a vazão máxima quando N é um número muito grande.

5-31 Considere que existam apenas três estações ativas em uma rede usando *slotted* ALOHA: A, B e C. A probabilidade de que cada estação gere um quadro em uma faixa de tempo (*slot*) é dada respectivamente por $p_A = 0{,}2$; $p_B = 0{,}3$ e $p_C = 0{,}4$.

a. Qual é a vazão de cada estação?

b. Qual é a vazão da rede?

5-32 Considere que existam apenas três estações ativas em uma rede usando *slotted* ALOHA: A, B e C. A probabilidade de que cada estação gere um quadro em uma faixa de tempo (*slot*) é dada respectivamente por $p_A = 0{,}2$; $p_B = 0{,}3$ e $p_C = 0{,}4$.

a. Qual é a probabilidade de que alguma estação envie um quadro na primeira faixa de tempo?

b. Qual é a probabilidade de que a estação A envie um quadro com sucesso pela primeira vez na segunda faixa de tempo?

c. Qual é a probabilidade de que a estação C envie um quadro com sucesso pela primeira vez na terceira faixa de tempo?

5-33 Uma rede usando *slotted* ALOHA está operando na sua vazão máxima.

a. Qual é a probabilidade de que uma faixa de tempo esteja vazia?

b. Quantas faixas de tempo, em média, devem aparecer antes de observarmos uma faixa de tempo que esteja vazia?

5-34 Um parâmetro útil em uma LAN é o número de *bits* que cabem em um metro do meio de transmissão ($n_{b/m}$). Determine o valor de $n_{b/m}$, se a taxa de transferência de dados for de 100 Mbps e a velocidade de propagação no médio for de 2×10^8 m/s.

5-35 Outro parâmetro útil em uma LAN é o comprimento de *bit* do meio (L_b), que corresponde ao número de *bits* que o meio pode comportar em qualquer instante. Determine o comprimento de *bit* de uma LAN se a taxa de transferência de dados for de 100 Mbps e o comprimento médio em metros (L_m) para a comunicação entre duas estações for de 200 m. Considere que a velocidade de propagação no meio seja de 2×10^8 m/s.

5-36 Definimos o parâmetro a, o número de quadros que o meio entre duas estações pode comportar, como sendo $a = (T_p)/(T_{fr})$. Outra forma de definir esse parâmetro é $a = L_b/F_b$, onde L_b é o comprimento de *bit* do meio e F_b é o comprimento de quadro do meio. Mostre que essas duas definições são equivalentes.

5-37 Em uma rede em barramento usando CSMA e com uma taxa de transferência de dados de 10 Mbps, uma colisão ocorre 20 µs após o primeiro *bit* do quadro deixar a estação de origem. Qual deve ser o comprimento do quadro de modo que o remetente seja capaz de detectar a colisão?

5-38 Considere que existam apenas duas estações, A e B, em uma rede em barramento usando CSMA/CD. A distância entre as duas estações é de 2000 m e a velocidade de propagação é 2×10^8 m/s. Se a estação A começar a transmitir no instante de tempo t_1:

a. O protocolo permite que a estação B comece a transmitir no instante $t_1 + 8$ µs? Se a resposta for sim, o que vai acontecer?

b. O protocolo permite que a estação B comece a transmitir no instante $t_1 + 11$ µs? Se a resposta for sim, o que vai acontecer?

5-39 Considere que existam apenas duas estações, A e B, em uma rede em barramento usando CSMA/CD 1-persistente, na qual $T_p = 25{,}6$ µs e $T_{tr} = 51{,}2$ µs. A estação A tem um quadro que ela deseja enviar à estação B. O quadro é enviado duas vezes sem sucesso, mas é entregue com sucesso na terceira tentativa. Desenhe um diagrama com a linha do tempo que descreve esse problema. Considere que o valor de R seja 1 e 2, respectivamente, e ignore o tempo necessário para enviar um sinal de interferência (ver a Figura 5.40).

5-40 Para entender a razão pela qual precisamos ter um tamanho mínimo de quadro $T_{tr} = 2 \times T_p$ em uma rede usando CDMA/CD, suponha que temos uma rede em barramento com apenas duas estações, A e B, na qual $T_{tr} = 40$ µs e $T_p = 25$ µs. A estação A começa a enviar um quadro no instante $t = 0{,}0$ µs e a estação B começa a enviar um quadro no instante $t = 23$ µs. Responda às seguintes questões:

a. Os quadros colidem?

b. Se a resposta para o item **a** for sim, a estação A detecta a colisão?

c. Se a resposta para o item **a** for sim, a estação B detecta a colisão?

5-41 Em um barramento usando CSMA/CD 1-persistente no qual $T_p = 50$ µs e $T_{tr} = 120$ µs, existem duas estações, A e B. Elas começam a enviar quadros uma para a outra ao mesmo tempo. Como os quadros colidem, ambas as estações tentam retransmitir. A estação A sorteia R = 0, enquanto a estação B sorteia R = 1. Ignore qualquer outro atraso, incluindo o atraso relativo ao envio de sinais de interferência. Os quadros colidem novamente? Desenhe um diagrama com a linha de tempo que comprove a sua resposta. A geração de um número aleatório ajuda a evitar a colisão nesse caso?

5-42 A variável aleatória R (Figura 5.40) é projetada para gerar atrasos distintos em cada estação após a ocorrência de uma colisão. Para reduzir o número de colisões, espera-se que estações diferentes gerem valores distintos de R. Para isso, determine a probabilidade de que o valor de R seja o mesmo para duas estações após

a. A primeira colisão.

b. A segunda colisão.

5-43 Considere que tenhamos uma rede que usa *slotted* CSMA/CD. Cada estação nesta rede adota um período de contenção, durante o qual a estação abstém-se de acessar o canal compartilhado antes de tentar enviar um quadro. Considere que o período de contenção seja composto por faixas de tempo (*slots*) de contenção. No início de cada faixa, a estação verifica o meio. Se ele estiver livre, a estação envia seu quadro; se o canal estiver ocupado, a estação se abstém de enviar o quadro e espera até o início da próxima faixa. Em outras palavras, a estação espera, em média, por k faixas antes de enviar seu quadro, conforme mostra a Figura 5.91. Observe que o canal pode estar no estado de contenção, de transmissão ou no estado ocioso (quando nenhuma estação tiver um quadro para enviar). Entretanto, se N for um número muito grande, o estado ocioso acaba desaparecendo.

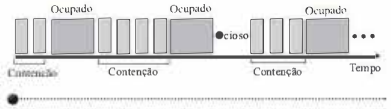


Figura 5.91 Esquema para o Problema 5-43.

a. Qual é a probabilidade de haver uma faixa livre (P_{livre}) se o número de estações for N e se cada estação tiver um quadro para enviar com probabilidade p?

b. Qual é o valor máximo dessa probabilidade quando N é um número muito grande?

c. Qual é a probabilidade de que a j-ésima faixa esteja livre?

d. Qual é o número médio de faixas, k, que uma estação deve esperar antes de encontrar uma faixa livre?

e. Qual é o valor de k quando N (o número de estações) é muito grande?

5-44 Embora o cálculo da vazão em uma rede CDMA/CD seja realmente complicado, podemos calcular a vazão máxima de uma rede *slotted* CDMA/CD usando a especificação descrita no problema anterior. Descobrimos que o número médio de faixas de contenção que uma estação precisa esperar é $k = e$ faixas. Com esse pressuposto, a vazão de uma rede usando *slotted* CDMA/CD é dada por

$V = (T_p)/$ **(tempo durante o qual o canal fica ocupado com um quadro)**

O tempo durante o qual o canal fica ocupado com um quadro corresponde ao tempo que precisamos esperar por uma faixa livre somado ao tempo para transmitir o quadro somado ao atraso de propagação até a estação perceber que não houve colisão. Considere que a duração de uma faixa de contenção seja $2 \times (T_p)$ e que $a = (T_p)/(T_p)$. Note que o parâmetro a é o número de quadros que ocupam o meio de transmissão. Determine a vazão de uma rede usando *slotted* CDMA/CD em função do parâmetro a.

5-45 Considere uma rede usando ALOHA puro com uma taxa de transferência de dados de 10 Mbps. Qual é o número máximo de quadros de 1.000 *bits* que podem ser enviados com sucesso por essa rede?

5-46 Em uma rede usando CDMA/CD com uma taxa de transferência de dados de 10 Mbps, o tamanho mínimo do quadro para a operação correta do processo de detecção de colisão foi calculado como sendo 512 *bits*.

Qual deve ser o tamanho mínimo do quadro se mantivermos o tamanho da rede constante, porém aumentarmos a taxa de transferência de dados para

a. 100 Mbps?

b. 1 Gbps?

c. 10 Gbps?

5-47 Qual é o valor hexadecimal equivalente ao endereço Ethernet a seguir?

```
01011010 00010001 01010101
00011000 10101010 00001111
```

5-48 Como o endereço Ethernet 1A:2B:3C:4D:5E:6F aparece na linha de transmissão, em binário?

5-49 Se um endereço Ethernet de destino é 07:01:02:03:04:05, qual é o seu tipo de endereço (*unicast*, *multicast* ou *broadcast*)?

5-50 Em uma rede usando CSMA/CD, há duas estações, A e B. Suponha que essas duas estações tenham um quadro para enviar durante o intervalo de colisão ($2 \times t_p$) e que elas o façam com probabilidade p_1 e p_2, respectivamente. Responda às seguintes perguntas, considerando $p_1 = 0{,}3$ e $p_2 = 0{,}4$:

a. Qual é a probabilidade de que estação A envie o quadro com sucesso?

b. Qual é a probabilidade de que estação B envie o quadro com sucesso?

c. Qual é a probabilidade de que um quadro seja enviado com sucesso?

5-51 Um roteador cujo endereço IPv4 é 125.45.23.12 e cujo endereço Ethernet é 23:45:AB:4F:67:CD recebeu um pacote destinado a uma estação cujo endereço IP é 125.11.78.10. Mostre os campos no pacote ARP de pedido enviado pelo roteador. Mostre também como fica o encapsulamento do pacote em um quadro Ethernet.

5-52 Na Figura 5.50, mostre as operações no lado de Bob quando a resposta é enviada de Bob para Alice.

5-53 A subcamada MAC da Ethernet recebe 42 *bytes* de dados da camada superior. Quantos *bytes* de enchimento (*padding*) devem ser adicionados aos dados?

5-54 Qual é a razão entre o tamanho dos dados úteis e o tamanho do quadro inteiro no menor quadro Ethernet?

5-55 Suponha que o comprimento de um cabo 10Base5 seja de 2500 m. Se a velocidade de propagação em um cabo coaxial grosso é de 200.000.000 m/s, quanto tempo leva para que um *bit* viaje de uma extremidade da rede até a outra? Considere um atraso de 10 µs no equipamento.

5-56 Que tipo de topologia é usado quando os clientes em uma região usam modems DSL para fins de transferência de dados? Explique.

5-57 Um *switch* da camada de enlace utiliza uma tabela de filtragem; já um roteador usa uma tabela de roteamento. Você pode explicar a diferença?

5.10 EXPERIMENTOS DE SIMULAÇÃO

5.10.1 *Applets*

Criamos alguns *applets* Java para demonstrar alguns dos principais conceitos discutidos neste capítulo. Recomendamos que o estudante ative esses *applets* no *site* www.mhhe.com/forouzan e examine cuidadosamente os protocolos em ação.

5.10.2 Experimentos de laboratório

Nesta seção, usamos o Wireshark para simular dois protocolos: o protocolo Ethernet e o protocolo ARP. As descrições completas desses experimentos de laboratório encontram-se no *site* www.grupoa.com.br.

Lab5-1 Neste laboratório, desejamos examinar o conteúdo de um quadro enviado pela camada de enlace de dados. Queremos descobrir o valor de diferentes campos, como os endereços MAC de origem e de destino, o valor do CRC, o valor do campo de protocolo (que mostra o tipo de carga útil sendo transportado pelo quadro), e assim por diante.

Lab5-2 Neste laboratório, desejamos examinar o conteúdo de um pacote ARP. Queremos capturar os pacotes ARP de pedido e de resposta. Os campos interessantes de examinarmos são aqueles que mostram que tipo de endereços de origem e de destino são usados em cada pacote.

5.11 TAREFAS DE PROGRAMAÇÃO

Para cada uma das tarefas a seguir, escreva um programa em uma linguagem de programação de sua preferência.

Prg5-1 Escreva e teste um programa que simule o preenchimento e a remoção de preenchimento de *byte*, conforme mostrado na Figura 5.5.

Prg5-2 Escreva e teste um programa que simule o preenchimento e a remoção de preenchimento de *bits*, conforme mostrado na Figura 5.7.

Prg5-3 Escreva e teste um programa que simule o diagrama de fluxo da Figura 5.17.

Prg5-4 Escreva e teste um programa que simule o diagrama de fluxo da Figura 5.18.

Prg5-5 Escreva e teste um programa que simule o diagrama de fluxo da Figura 5.19.