Capítulo 2: Links Diretos

Problema: Conectando a uma rede

No Capítulo 1, vimos que as redes consistem em links que interconectam nós. Um dos problemas fundamentais que enfrentamos é como conectar dois nós. Também introduzimos a abstração de "nuvem" para representar uma rede sem revelar todas as suas complexidades internas. Portanto, também precisamos abordar o problema semelhante de conectar um host a uma nuvem. Este é, na verdade, o problema que todo Provedor de Serviços de Internet (ISP) enfrenta quando deseja conectar um novo cliente à sua rede.

Quer queiramos construir uma rede trivial de dois nós com um único link ou conectar o bilionésimo host a uma rede existente, como a internet, precisamos abordar um conjunto comum de questões. Primeiro, precisamos de um meio físico para fazer a conexão. O meio pode ser um pedaço de fio, um pedaço de fibra óptica ou algum meio menos tangível (como o ar) através do qual a radiação eletromagnética (por exemplo, ondas de rádio) possa ser transmitida. Pode cobrir uma área pequena (por exemplo, um prédio comercial) ou uma área ampla (por exemplo, transcontinental).

No entanto, conectar dois nós com um meio adequado é apenas o primeiro passo. Cinco problemas adicionais devem ser resolvidos antes que os nós possam trocar pacotes com sucesso e, uma vez resolvidos, teremos fornecido conectividade *de Camada 2* (L2) (usando terminologia da arquitetura OSI).

O primeiro é *codificar* bits no meio de transmissão para que eles possam ser entendidos por um nó receptor. O segundo é a questão de delinear a sequência de bits transmitidos pelo link em mensagens completas que podem ser entregues ao nó final. Este é o problema *de enquadramento* , e as mensagens entregues aos hosts finais são

frequentemente chamadas de *quadros* (ou às vezes *pacotes*). Terceiro, como os quadros às vezes são corrompidos durante a transmissão, é necessário detectar esses erros e tomar as medidas apropriadas; este é o problema *de detecção de erros* . O quarto problema é fazer um link parecer confiável, apesar do fato de corromper quadros de tempos em tempos. Finalmente, nos casos em que o link é compartilhado por vários hosts — como geralmente é o caso com links sem fio, por exemplo — é necessário mediar o acesso a esse link. Este é o problema *de controle de acesso à mídia* .

Embora essas cinco questões — codificação, enquadramento, detecção de erros, entrega confiável e mediação de acesso — possam ser discutidas em abstrato, elas são problemas bastante reais, abordados de diferentes maneiras por diferentes tecnologias de rede. Este capítulo considera essas questões no contexto de tecnologias de rede específicas: enlaces de fibra óptica ponto a ponto (dos quais o SONET é o exemplo predominante); redes de Acesso Múltiplo com Detecção de Operadora (CSMA) (das quais a Ethernet clássica e o Wi-Fi são os exemplos mais famosos); fibra óptica até a residência (dos quais o PON é o padrão dominante); e redes sem fio móveis (onde o 4G está rapidamente se transformando em 5G).

O objetivo deste capítulo é, simultaneamente, analisar a tecnologia disponível em nível de enlace e explorar essas cinco questões fundamentais. Examinaremos o que é necessário para tornar uma ampla variedade de diferentes mídias físicas e tecnologias de enlace úteis como blocos de construção para a construção de redes robustas e escaláveis.

2.1 Panorama Tecnológico

Antes de nos aprofundarmos nos desafios descritos na declaração do problema no início deste capítulo, é útil primeiro conhecer o cenário, que inclui uma ampla gama de

tecnologias de enlace. Isso se deve, em parte, às diversas circunstâncias em que os usuários tentam conectar seus dispositivos.

Em uma extremidade do espectro, as operadoras de rede que constroem redes globais precisam lidar com links que se estendem por centenas ou milhares de quilômetros, conectando roteadores do tamanho de geladeiras. Na outra extremidade do espectro, um usuário típico encontra links principalmente como uma forma de conectar um computador à internet existente. Às vezes, esse link será um link sem fio (Wi-Fi) em uma cafeteria; às vezes, um link Ethernet em um prédio comercial ou universidade; às vezes, um smartphone conectado a uma rede celular; para uma parcela cada vez maior da população, é um link de fibra óptica fornecido por um ISP; e muitos outros usam algum tipo de fio ou cabo de cobre para se conectar. Felizmente, existem muitas estratégias comuns usadas nesses tipos aparentemente díspares de links para que todos possam se tornar confiáveis e úteis para camadas superiores na pilha de protocolos. Este capítulo examina essas estratégias.

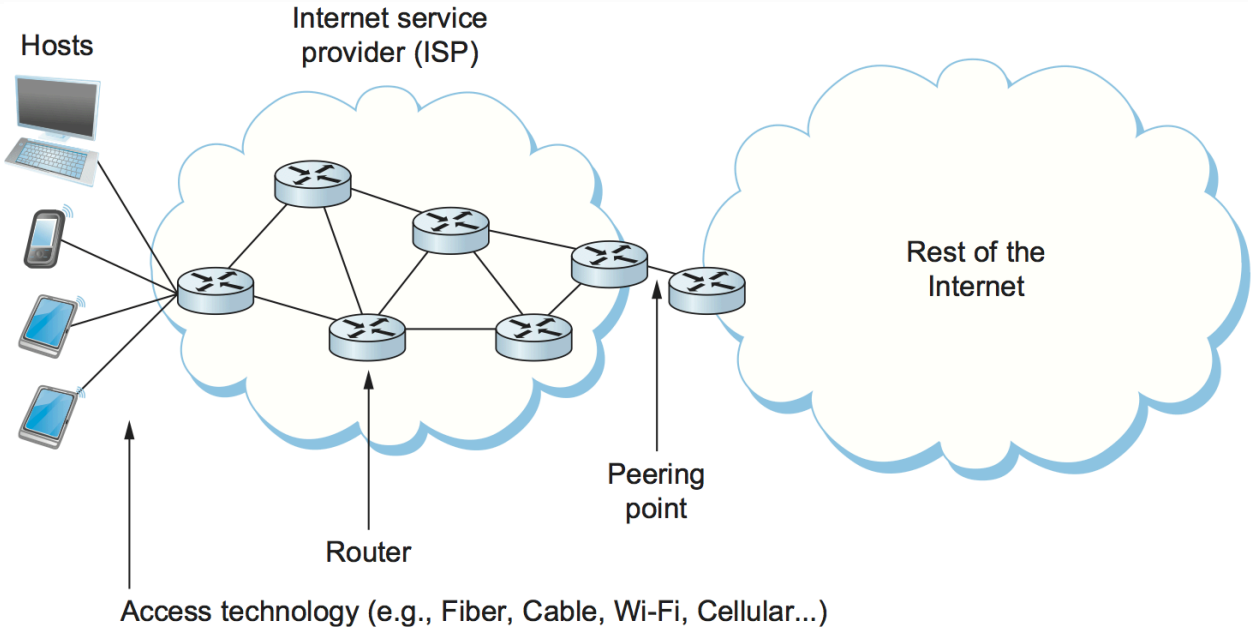

Figura 21. *Visão da Internet por um usuário final.*

A Figura 21 ilustra vários tipos de links que podem ser encontrados na internet atual. À esquerda, vemos uma variedade de dispositivos de usuário final, desde smartphones e tablets até computadores completos, conectados por diversos meios a um ISP. Embora esses links possam usar tecnologias diferentes, todos parecem iguais nesta imagem — uma linha reta conectando um dispositivo a um roteador. Há links que conectam roteadores dentro do ISP, bem como links que conectam o ISP ao "resto da internet", que consiste em vários outros ISPs e os hosts aos quais eles se conectam.

Todos esses links parecem iguais não apenas porque não somos muito bons artistas, mas porque parte do papel de uma arquitetura de rede é fornecer uma abstração comum de algo tão complexo e diverso quanto um link. A ideia é que seu laptop ou smartphone não precise se importar com o tipo de link ao qual está conectado — a única coisa que importa é que ele tenha um link com a internet. Da mesma forma, um roteador não precisa se importar com o tipo de link que o conecta a outros roteadores — ele pode enviar um pacote pelo link com uma expectativa bastante alta de que o pacote chegará à outra extremidade do link.

Como fazemos com que todos esses diferentes tipos de links pareçam suficientemente semelhantes para usuários finais e roteadores? Essencialmente, temos que lidar com todas as limitações físicas e deficiências dos links que existem no mundo real. Esboçamos algumas dessas questões na declaração do problema de abertura deste capítulo, mas antes de podermos discuti-las, precisamos primeiro introduzir um pouco de física simples. Todos esses links são feitos de algum material físico que pode propagar sinais, como ondas de rádio ou outros tipos de radiação eletromagnética, mas o que realmente queremos fazer é enviar *bits* . Nas seções posteriores deste capítulo, veremos como codificar bits para transmissão em um meio físico, seguido pelas outras questões mencionadas acima. Ao final deste capítulo, entenderemos como enviar pacotes completos por praticamente qualquer tipo de link, independentemente do meio físico envolvido.

Uma maneira de caracterizar os links, então, é pelo meio que eles usam — normalmente, fio de cobre de alguma forma, como par trançado (alguns Ethernets e telefones fixos) e coaxial (cabo); fibra óptica, que é usada tanto para fibra óptica até a residência quanto para muitos links de longa distância na espinha dorsal da Internet; ou ar/espaço livre para links sem fio.

Outra característica importante do elo é a *frequência* , medida em hertz, com a qual as ondas eletromagnéticas oscilam. A distância entre um par de máximos ou mínimos adjacentes de uma onda, normalmente medida em metros, é chamada de *comprimento de onda* da onda . Como todas as ondas eletromagnéticas viajam à velocidade da luz (que por sua vez depende do meio), essa velocidade dividida pela frequência da onda é igual ao seu comprimento de onda. Já vimos o exemplo de uma linha telefônica de nível de voz, que transporta sinais eletromagnéticos contínuos variando entre 300 Hz e 3300 Hz; uma onda de 300 Hz viajando através do cobre teria um comprimento de onda de

VelocidadeDaLuzEmCobre / Frequência

= 2/3 × 3 × 10^8 / 300

= 667 × 10^3 *metros*

Em geral, as ondas eletromagnéticas abrangem uma faixa de frequências muito mais ampla, desde ondas de rádio, luz infravermelha, luz visível, raios X e raios gama. A Figura 22 ilustra o espectro eletromagnético e mostra quais meios são comumente usados para transportar quais faixas de frequência.

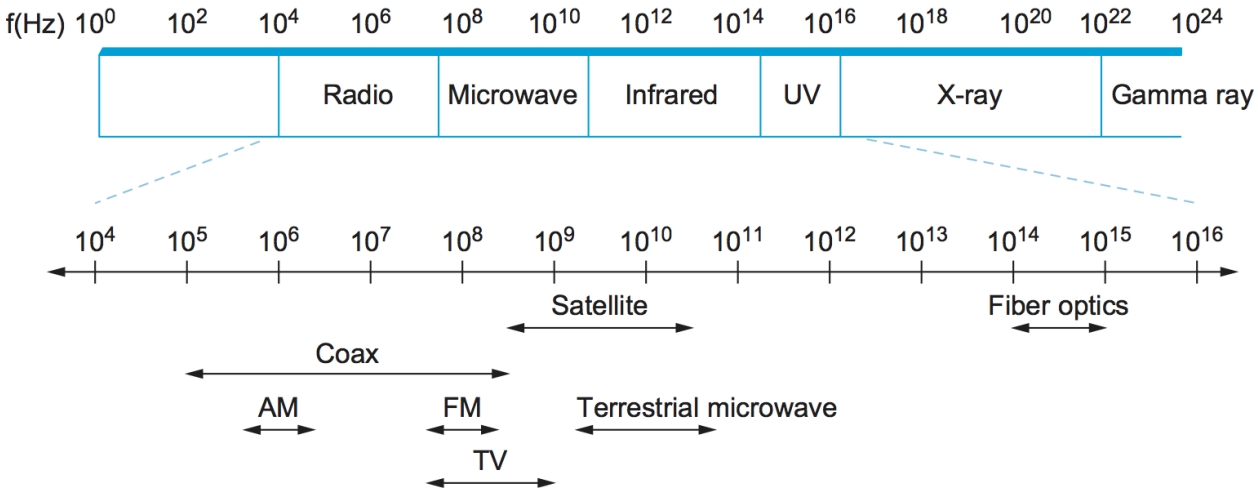

Figura 22. *Espectro eletromagnético.*

O que a Figura 22 não mostra é onde a rede celular se encaixa. Isso é um pouco complicado porque as faixas de frequência específicas licenciadas para redes celulares variam ao redor do mundo, e ainda mais complicado pelo fato de que as operadoras de rede frequentemente suportam simultaneamente tecnologias antigas/legadas e tecnologias novas/de próxima geração, cada uma ocupando uma faixa de frequência diferente. O resumo geral é que as tecnologias celulares tradicionais variam de 700 MHz a 2.400 MHz, com novas alocações de espectro médio agora acontecendo em 6 GHz e alocações de ondas milimétricas (mmWave) abrindo acima de 24 GHz. Essa faixa de mmWave provavelmente se tornará uma parte importante da rede móvel 5G.

Até agora, entendemos que um enlace é um meio físico que transporta sinais na forma de ondas eletromagnéticas. Tais enlaces fornecem a base para a transmissão de todos os tipos de informação, incluindo o tipo de dado que estamos interessados em transmitir — dados binários (1s e 0s). Dizemos que os dados binários estão *codificados* no sinal. O problema da codificação de dados binários em sinais eletromagnéticos é um tópico complexo. Para ajudar a tornar o tópico mais administrável, podemos considerá-lo dividido em duas camadas. A camada inferior trata da *modulação* — a variação da frequência, amplitude ou fase do sinal para efetuar a transmissão de informações. Um exemplo simples de modulação é variar a potência (amplitude) de um

único comprimento de onda. Intuitivamente, isso equivale a ligar e desligar uma luz. Como a questão da modulação é secundária à nossa discussão sobre enlaces como um bloco de construção para redes de computadores, simplesmente assumimos que é possível transmitir um par de sinais distinguíveis — pense neles como um sinal "alto" e um sinal "baixo" — e consideramos apenas a camada superior, que se ocupa do problema muito mais simples de codificar dados binários nesses dois sinais. A próxima seção discute tais codificações.

Outra maneira de classificar links é em termos de como eles são usados. Várias questões econômicas e de implantação tendem a influenciar onde diferentes tipos de link são encontrados. A maioria dos consumidores interage com a Internet por meio de redes sem fio (que eles encontram em cafeterias, aeroportos, universidades, etc.) ou por meio dos chamados links *de última milha (ou alternativamente, redes de acesso*) fornecidos por um ISP, conforme ilustrado na Figura 21. Esses tipos de link estão resumidos na Tabela 2. Eles normalmente são escolhidos porque são maneiras econômicas de alcançar milhões de consumidores. DSL (Digital Subscriber Line), por exemplo, é uma tecnologia mais antiga que foi implantada sobre os fios de cobre de par trançado existentes que já existiam para serviços telefônicos antigos; G.Fast é uma tecnologia baseada em cobre normalmente usada em prédios de apartamentos multifamiliares, e PON (Passive Optical Network) é uma tecnologia mais nova que é comumente usada para conectar casas e empresas por fibra implantada recentemente.

Serviço	**Largura de banda**
DSL (cobre)	até 100 Mbps

G.Fast (cobre)	até 1 Gbps
PON (óptico)	até 10 Gbps

E, claro, há também a rede *móvel* ou *celular* (também conhecida como 4G, mas que está evoluindo rapidamente para 5G) que conecta nossos dispositivos móveis à internet. Essa tecnologia também pode servir como a única conexão de internet em nossas casas ou escritórios, mas tem o benefício adicional de nos permitir manter a conexão à internet enquanto nos deslocamos de um lugar para outro.

Essas tecnologias de exemplo são opções comuns para a conexão de última milha para sua casa ou empresa, mas não são suficientes para construir uma rede completa do zero. Para isso, você também precisará de alguns links *de backbone* de longa distância para interconectar cidades. Os links de backbone modernos são quase exclusivamente de fibra óptica e normalmente utilizam uma tecnologia chamada SONET (Rede Óptica Síncrona), que foi originalmente desenvolvida para atender aos exigentes requisitos de gerenciamento das operadoras de telefonia.

Por fim, além dos links de última milha, backbone e móveis, existem os links que você encontra dentro de um prédio ou campus — geralmente chamados de *redes locais* (LANs). Ethernet e seu primo sem fio, o Wi-Fi, são as tecnologias dominantes nesse setor.

Esta análise dos tipos de enlace não é de forma alguma exaustiva, mas deve ter dado a você uma ideia da diversidade de tipos de enlace existentes e algumas das razões para essa diversidade. Nas próximas seções, veremos como os protocolos de rede podem aproveitar essa diversidade e apresentar uma visão consistente da rede para

camadas superiores, apesar de toda a complexidade de baixo nível e dos fatores econômicos.

2.2 Codificação

O primeiro passo para transformar nós e enlaces em blocos de construção utilizáveis é entender como conectá-los de forma que bits possam ser transmitidos de um nó para o outro. Conforme mencionado na seção anterior, os sinais se propagam por enlaces físicos. A tarefa, portanto, é codificar os dados binários que o nó de origem deseja enviar nos sinais que os enlaces são capazes de transportar e, em seguida, decodificar o sinal de volta nos dados binários correspondentes no nó receptor. Ignoramos os detalhes da modulação e assumimos que estamos trabalhando com dois sinais discretos: alto e baixo. Na prática, esses sinais podem corresponder a duas tensões diferentes em um enlace de cobre, dois níveis de potência diferentes em um enlace óptico ou duas amplitudes diferentes em uma transmissão de rádio.

A maioria das funções discutidas neste capítulo é realizada por um *adaptador de rede* — um componente de hardware que conecta um nó a um enlace. O adaptador de rede contém um componente de sinalização que, na verdade, codifica bits em sinais no nó emissor e decodifica sinais em bits no nó receptor. Assim, como ilustrado na Figura 23 , os sinais trafegam por um enlace entre dois componentes de sinalização e os bits fluem entre os adaptadores de rede.

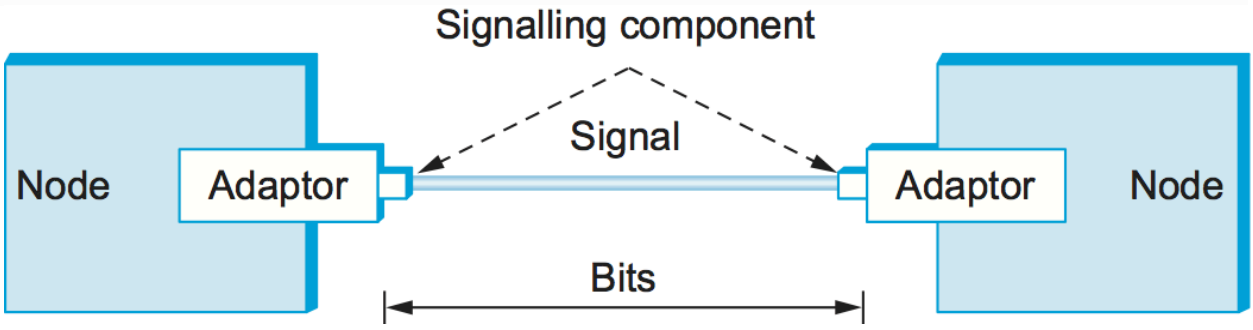

Figura 23. *Os sinais trafegam entre os componentes de sinalização; os bits fluem entre os adaptadores.*

Voltemos ao problema da codificação de bits em sinais. A solução óbvia é mapear o valor de dado 1 para o sinal alto e o valor de dado 0 para o sinal baixo. Este é exatamente o mapeamento usado por um esquema de codificação chamado, enigmaticamente, *de não retorno a zero* (NRZ). Por exemplo, a Figura 24 representa esquematicamente o sinal codificado em NRZ (embaixo) que corresponde à transmissão de uma sequência específica de bits (em cima).

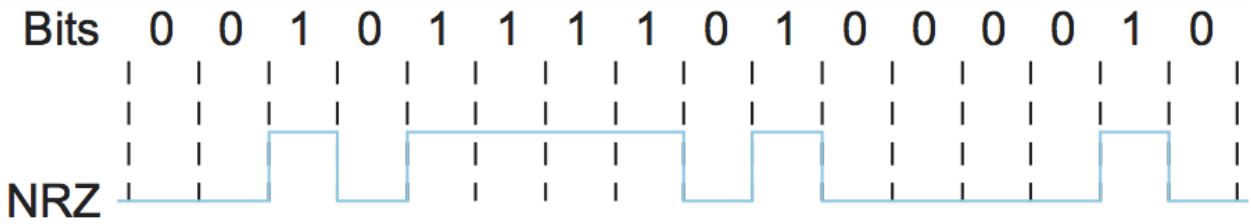

Figura 24. *Codificação NRZ de um fluxo de bits.*

O problema com NRZ é que uma sequência de vários 1s consecutivos significa que o sinal permanece alto no link por um longo período de tempo; similarmente, vários 0s consecutivos significam que o sinal permanece baixo por um longo tempo. Existem dois problemas fundamentais causados por longas sequências de 1s ou 0s. O primeiro é que isso leva a uma situação conhecida como *desvio de linha de base* . Especificamente, o receptor mantém uma média do sinal que viu até agora e então usa essa média para distinguir entre sinais baixos e altos. Sempre que o sinal for significativamente menor do que essa média, o receptor conclui que acabou de ver um 0; da mesma forma, um sinal significativamente maior do que a média é interpretado como 1. O problema, é claro, é que muitos 1s ou 0s consecutivos fazem com que essa média mude, tornando mais difícil detectar uma mudança significativa no sinal.

O segundo problema é que transições frequentes de alta para baixa e *vice-versa* são necessárias para permitir *a recuperação do relógio* . Intuitivamente, o problema da recuperação do relógio é que tanto o processo de codificação quanto o de

decodificação são controlados por um relógio — a cada ciclo de relógio, o remetente transmite um bit e o destinatário recupera um bit. Os relógios do remetente e do destinatário precisam ser sincronizados precisamente para que o destinatário recupere os mesmos bits que o remetente transmite. Se o relógio do destinatário for, mesmo que ligeiramente mais rápido ou mais lento que o relógio do remetente, ele não decodifica o sinal corretamente. Você poderia imaginar enviar o relógio para o destinatário por um fio separado, mas isso normalmente é evitado porque torna o custo do cabeamento duas vezes maior. Então, em vez disso, o destinatário deriva o relógio do sinal recebido — o processo de recuperação do relógio. Sempre que o sinal muda, como em uma transição de 1 para 0 ou de 0 para 1, o destinatário sabe que está em um limite de ciclo de relógio e pode se ressincronizar. No entanto, um longo período de tempo sem tal transição leva ao desvio do relógio. Portanto, a recuperação do relógio depende de muitas transições no sinal, não importa quais dados estejam sendo enviados.

Uma abordagem que aborda esse problema, chamada de *não retorno a zero invertido* (NRZI), faz com que o remetente faça uma transição do sinal atual para codificar um 1 e permaneça no sinal atual para codificar um 0. Isso resolve o problema de 1s consecutivos, mas obviamente não faz nada para 0s consecutivos. NRZI é ilustrado na Figura 25. Uma alternativa, chamada *codificação Manchester* , faz um trabalho mais explícito de mesclar o relógio com o sinal transmitindo o OU exclusivo dos dados codificados por NRZ e o relógio. (Pense no relógio local como um sinal interno que alterna de baixo para alto; um par baixo/alto é considerado um ciclo de relógio.) A codificação Manchester também é ilustrada na Figura 25. Observe que a codificação Manchester resulta em 0 sendo codificado como uma transição de baixo para alto e 1 sendo codificado como uma transição de alto para baixo. Como ambos os 0s e 1s resultam em uma transição para o sinal, o relógio pode ser efetivamente recuperado no receptor. (Há também uma variante da codificação Manchester, chamada *Manchester Diferencial* , na qual um 1 é codificado com a primeira metade do sinal igual à última

metade do sinal do bit anterior e um 0 é codificado com a primeira metade do sinal oposta à última metade do sinal do bit anterior.)

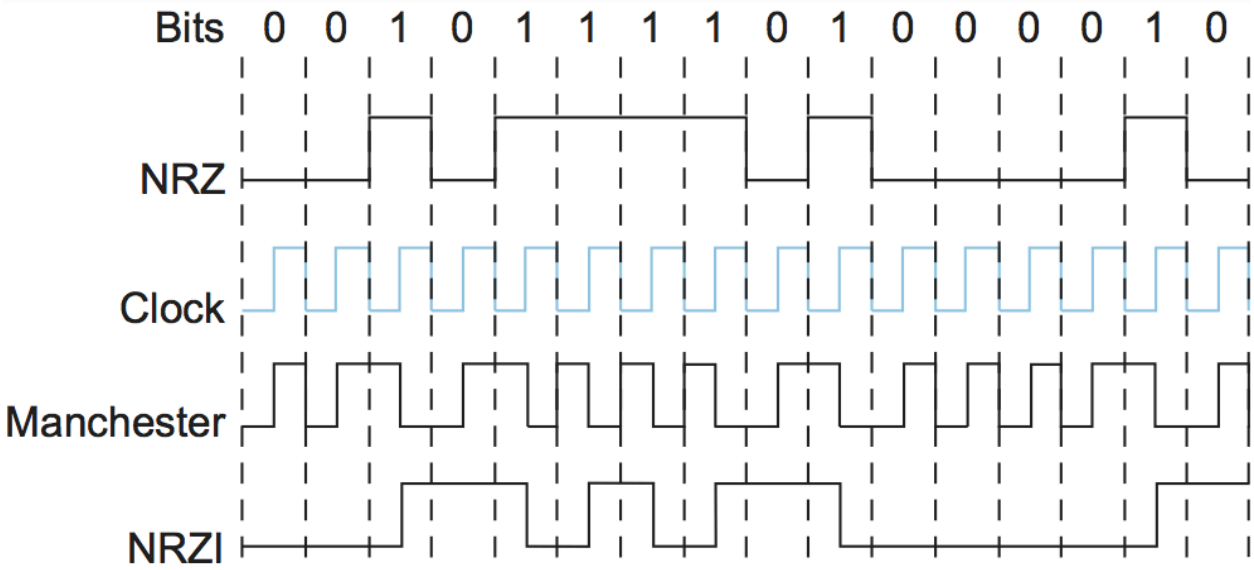

Figura 25. *Diferentes estratégias de codificação.*

O problema com o esquema de codificação Manchester é que ele dobra a taxa na qual as transições de sinal são feitas no link, o que significa que o receptor tem metade do tempo para detectar cada pulso do sinal. A taxa na qual o sinal muda é chamada de *taxa de transmissão* do link . No caso da codificação Manchester, a taxa de bits é metade da taxa de transmissão, então a codificação é considerada apenas 50% eficiente. Tenha em mente que se o receptor tivesse sido capaz de acompanhar a taxa de transmissão mais rápida exigida pela codificação Manchester na Figura 25 , então tanto NRZ quanto NRZI poderiam ter transmitido o dobro de bits no mesmo período de tempo.

Observe que a taxa de bits não é necessariamente menor ou igual à taxa de transmissão, como sugere a codificação Manchester. Se o esquema de modulação for capaz de utilizar (e reconhecer) quatro sinais diferentes, em oposição a apenas dois (por exemplo, "alto" e "baixo"), então é possível codificar dois bits em cada intervalo de clock, resultando em uma taxa de bits que é o dobro da taxa de transmissão. Da

mesma forma, ser capaz de modular entre oito sinais diferentes significa ser capaz de transmitir três bits por intervalo de clock. Em geral, é importante ter em mente que temos uma modulação simplificada demais, que é muito mais sofisticada do que transmitir sinais "altos" e "baixos". Não é incomum variar uma combinação de fase e amplitude de um sinal, tornando possível codificar 16 ou até 64 padrões diferentes (frequentemente chamados de *símbolos*) durante cada intervalo de clock. *QAM (Quadrature Amplitude Modulation)* é um exemplo amplamente utilizado de tal esquema de modulação.

Uma codificação final que consideramos, chamada *4B/5B* , tenta resolver a ineficiência da codificação Manchester sem sofrer com o problema de ter durações estendidas de sinais altos ou baixos. A ideia de 4B/5B é inserir bits extras no fluxo de bits para quebrar longas sequências de 0s ou 1s. Especificamente, cada 4 bits de dados reais são codificados em um código de 5 bits que é então transmitido ao receptor; daí o nome 4B/5B. Os códigos de 5 bits são selecionados de tal forma que cada um tenha no máximo um 0 à esquerda e no máximo dois 0s à direita. Assim, quando enviados consecutivamente, nenhum par de códigos de 5 bits resulta em mais de três 0s consecutivos sendo transmitidos. Os códigos de 5 bits resultantes são então transmitidos usando a codificação NRZI, o que explica por que o código se preocupa apenas com 0s consecutivos — NRZI já resolve o problema de 1s consecutivos. Observe que a codificação 4B/5B resulta em 80% de eficiência.

Símbolo de dados de 4 bits	**Código de 5 bits**
0000	11110

0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110

1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

A Tabela 3 apresenta os códigos de 5 bits que correspondem a cada um dos 16 símbolos de dados de 4 bits possíveis. Observe que, como 5 bits são suficientes para codificar 32 códigos diferentes, e estamos usando apenas 16 deles para dados, restam 16 códigos que podemos usar para outros propósitos. Destes, o código`11111`é usado quando a linha está ociosa, o código`00000` corresponde a quando a linha está inativa e`00100`é interpretado como parada. Dos 13 códigos restantes, 7 não são válidos porque violam a regra "um 0 à esquerda, dois 0 à direita", e os outros 6 representam vários símbolos de controle. Alguns dos protocolos de enquadramento descritos posteriormente neste capítulo utilizam esses símbolos de controle.

2.3 Enquadramento

Agora que vimos como transmitir uma sequência de bits por um link ponto a ponto — de adaptador para adaptador — vamos considerar o cenário da Figura 26. Lembre-se do Capítulo 1 que estamos focando em redes comutadas por pacotes, o que significa

que blocos de dados (chamados de *quadros* neste nível), não fluxos de bits, são trocados entre os nós. É o adaptador de rede que permite que os nós troquem quadros. Quando o nó A deseja transmitir um quadro para o nó B, ele diz ao seu adaptador para transmitir um quadro da memória do nó. Isso resulta em uma sequência de bits sendo enviada pelo link. O adaptador no nó B então coleta a sequência de bits que chega ao link e deposita o quadro correspondente na memória de B. Reconhecer exatamente qual conjunto de bits constitui um quadro — isto é, determinar onde o quadro começa e termina — é o desafio central enfrentado pelo adaptador.

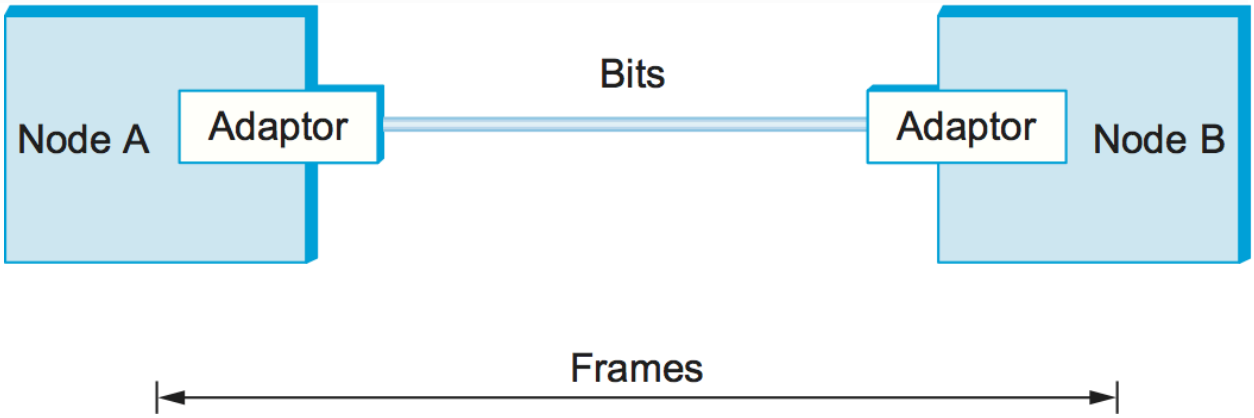

Figura 26. *Fluxo de bits entre adaptadores e quadros entre hosts.*

Existem várias maneiras de abordar o problema do enquadramento. Esta seção utiliza três protocolos diferentes para ilustrar os vários pontos no espaço de projeto. Observe que, embora discutamos o enquadramento no contexto de enlaces ponto a ponto, o problema é fundamental e também deve ser abordado em redes de acesso múltiplo, como Ethernet e Wi-Fi.

2.3.1 Protocolos Orientados a Bytes (PPP)

Uma das abordagens mais antigas para enquadramento — que tem suas raízes na conexão de terminais a mainframes — é visualizar cada quadro como uma coleção de bytes (caracteres) em vez de uma coleção de bits. Exemplos iniciais desses protocolos *orientados a bytes* são o protocolo de Comunicação Síncrona Binária (BISYNC),

desenvolvido pela IBM no final da década de 1960, e o Protocolo de Mensagem de Comunicação de Dados Digitais (DDCMP), usado no DECNET da Digital Equipment Corporation. (Antigamente, grandes empresas de informática como IBM e DEC também construíam redes privadas para seus clientes.) O amplamente utilizado Protocolo Ponto a Ponto (PPP) é um exemplo recente dessa abordagem.

Em um nível mais alto, existem duas abordagens para enquadramento orientado a bytes. A primeira é usar caracteres especiais conhecidos como *caracteres sentinela* para indicar onde os quadros começam e terminam. A ideia é denotar o início de um quadro enviando um caractere SYN (sincronização) especial. A porção de dados do quadro às vezes fica contida entre mais dois caracteres especiais: STX (início do texto) e ETX (fim do texto). O BISYNC usou essa abordagem. O problema com a abordagem sentinela, é claro, é que um dos caracteres especiais pode aparecer na porção de dados do quadro. A maneira padrão de superar esse problema é "escapar" o caractere precedendo-o com um caractere DLE (escape de link de dados) sempre que ele aparecer no corpo de um quadro; o caractere DLE também é escapado (precedendo-o com um DLE extra) no corpo do quadro. (Programadores C podem notar que isso é análogo à maneira como uma aspa é escapada pela barra invertida quando ocorre dentro de uma string.) Essa abordagem é frequentemente chamada de *preenchimento de caracteres* porque caracteres extras são inseridos na parte de dados do quadro.

A alternativa para detectar o fim de um quadro com um valor sentinela é incluir o número de bytes no quadro no início do quadro, no cabeçalho do quadro. O DDCMP usou essa abordagem. Um perigo com essa abordagem é que um erro de transmissão pode corromper o campo de contagem, nesse caso o fim do quadro não seria detectado corretamente. (Um problema semelhante existe com a abordagem baseada em sentinela se o campo ETX for corrompido.) Se isso acontecer, o receptor acumulará tantos bytes quanto o campo de contagem ruim indicar e, em seguida, usará o campo de detecção de erro para determinar se o quadro é ruim. Isso às vezes é chamado de *erro de enquadramento* . O receptor então esperará até ver o próximo caractere SYN para começar a coletar os bytes que compõem o próximo quadro. Portanto, é possível

que um erro de enquadramento faça com que quadros consecutivos sejam recebidos incorretamente.

O Protocolo Ponto a Ponto (PPP), comumente usado para transportar pacotes de Protocolo de Internet por vários tipos de links ponto a ponto, utiliza sentinelas e preenchimento de caracteres. O formato de um quadro PPP é apresentado na Figura 27 .

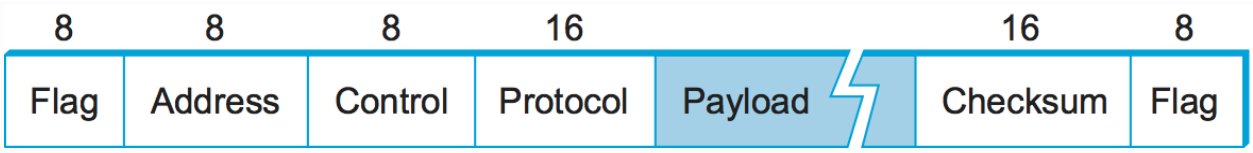

Figura 27. *Formato de quadro PPP.*

Esta figura é a primeira de muitas que você verá neste livro, usadas para ilustrar formatos de quadros ou pacotes, portanto, algumas palavras de explicação são necessárias. Mostramos um pacote como uma sequência de campos rotulados. Acima de cada campo, há um número que indica o comprimento desse campo em bits. Observe que os pacotes são transmitidos começando pelo campo mais à esquerda.

O caractere especial de início de texto, denotado como o `Flag`campo, é `01111110`. Os campos `Address`e `Control`geralmente contêm valores padrão e, portanto, não são interessantes. O campo (Protocolo) é usado para demultiplexação; ele identifica o protocolo de alto nível, como IP. O tamanho do payload do quadro pode ser negociado, mas é de 1500 bytes por padrão. O `Checksum`campo tem 2 (por padrão) ou 4 bytes de comprimento. Observe que, apesar do nome comum, este campo é, na verdade, um CRC e não uma soma de verificação (conforme descrito na próxima seção).

O formato de quadro PPP é incomum, pois vários tamanhos de campo são negociados em vez de fixos. Essa negociação é conduzida por um protocolo chamado Protocolo de Controle de Link (LCP). PPP e LCP trabalham em conjunto: o LCP envia mensagens de controle encapsuladas em quadros PPP — tais mensagens são denotadas por um identificador LCP no campo PPP (Protocolo) — e então altera o formato do quadro PPP

com base nas informações contidas nessas mensagens de controle. O LCP também está envolvido no estabelecimento de um link entre dois pares quando ambos os lados detectam que a comunicação através do link é possível (por exemplo, quando cada receptor óptico detecta um sinal de entrada da fibra à qual se conecta).

2.3.2 Protocolos Orientados a Bits (HDLC)

Ao contrário dos protocolos orientados a bytes, um protocolo orientado a bits não se preocupa com limites de bytes — ele simplesmente vê o quadro como uma coleção de bits. Esses bits podem vir de algum conjunto de caracteres, como ASCII; podem ser valores de pixel em uma imagem; ou podem ser instruções e operandos de um arquivo executável. O protocolo Synchronous Data Link Control (SDLC), desenvolvido pela IBM, é um exemplo de protocolo orientado a bits; o SDLC foi posteriormente padronizado pela ISO como o protocolo High-Level Data Link Control (HDLC). Na discussão a seguir, usamos o HDLC como exemplo; seu formato de quadro é apresentado na Figura 28 .

HDLC denota o início e o fim de um quadro com a sequência de bits distinta `01111110`. Essa sequência também é transmitida durante qualquer momento em que o link esteja ocioso, para que o remetente e o destinatário possam manter seus relógios sincronizados. Dessa forma, ambos os protocolos utilizam essencialmente a abordagem sentinela. Como essa sequência pode aparecer em qualquer lugar no corpo do quadro — na verdade, os bits `01111110`podem cruzar limites de bytes —, os protocolos orientados a bits utilizam o análogo do caractere DLE, uma técnica conhecida como *preenchimento de bits* .

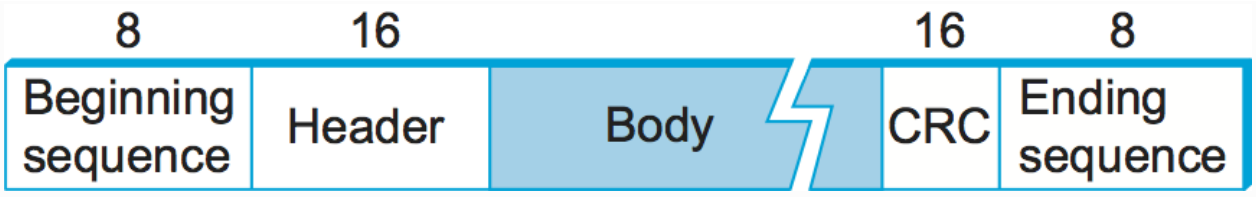

Figura 28. *Formato de quadro HDLC.*

O preenchimento de bits no protocolo HDLC funciona da seguinte maneira. No lado do envio, sempre que cinco 1s consecutivos forem transmitidos do corpo da mensagem (ou seja, excluindo quando o remetente está tentando transmitir a `01111110`sequência distinta), o remetente insere um 0 antes de transmitir o próximo bit. No lado do recebimento, se cinco 1s consecutivos chegarem, o receptor toma sua decisão com base no próximo bit que vê (ou seja, o bit após os cinco 1s). Se o próximo bit for 0, ele deve ter sido preenchido e, portanto, o receptor o remove. Se o próximo bit for 1, uma de duas coisas é verdadeira: ou este é o marcador de fim de quadro ou um erro foi introduzido no fluxo de bits. Ao olhar para o *próximo* bit, o receptor pode distinguir entre esses dois casos. Se ele vir um 0 (ou seja, os últimos 8 bits que ele olhou são `01111110`), então é o marcador de fim de quadro; Se ele vir um 1 (ou seja, os últimos 8 bits que ele examinou são `01111111`), então deve ter havido um erro e todo o quadro é descartado. Neste último caso, o receptor tem que esperar pelo próximo `01111110`antes de poder começar a receber novamente e, como consequência, existe a possibilidade de o receptor falhar em receber dois quadros consecutivos. Obviamente, ainda existem maneiras pelas quais erros de enquadramento podem passar despercebidos, como quando um padrão de fim de quadro espúrio inteiro é gerado por erros, mas essas falhas são relativamente improváveis. Maneiras robustas de detectar erros são discutidas em uma seção posterior.

Uma característica interessante do preenchimento de bits, assim como do preenchimento de caracteres, é que o tamanho de um quadro depende dos dados que estão sendo enviados na carga útil do quadro. De fato, não é possível fazer com que todos os quadros tenham exatamente o mesmo tamanho, visto que os dados que podem ser transportados em qualquer quadro são arbitrários. (Para se convencer disso, considere o que acontece se o último byte do corpo de um quadro for o caractere ETX.) Uma forma de enquadramento que garante que todos os quadros tenham o mesmo tamanho é descrita na próxima subseção.

2.3.3 Enquadramento baseado em relógio (SONET)

Uma terceira abordagem para enquadramento é exemplificada pelo padrão Synchronous Optical Network (SONET). Na ausência de um termo genérico amplamente aceito, nos referimos a essa abordagem simplesmente como *enquadramento baseado em clock* . O SONET foi proposto inicialmente pela Bell Communications Research (Bellcore) e, em seguida, desenvolvido pelo American National Standards Institute (ANSI) para transmissão digital por fibra óptica; desde então, foi adotado pela ITU-T. O SONET tem sido, por muitos anos, o padrão dominante para transmissão de dados de longa distância por redes ópticas.

Um ponto importante a ser destacado sobre o SONET antes de prosseguirmos é que a especificação completa é substancialmente maior do que este livro. Portanto, a discussão a seguir abordará necessariamente apenas os pontos altos do padrão. Além disso, o SONET aborda tanto o problema do enquadramento quanto o da codificação. Ele também aborda um problema muito importante para as operadoras de telefonia: a multiplexação de vários links de baixa velocidade em um link de alta velocidade. (Na verdade, grande parte do projeto do SONET reflete o fato de que as operadoras de telefonia precisam se preocupar com a multiplexação de um grande número de canais de 64 kbps, tradicionalmente usados para chamadas telefônicas.) Começaremos com a abordagem do SONET para o enquadramento e discutiremos as outras questões a seguir.

Assim como nos esquemas de enquadramento discutidos anteriormente, um quadro SONET possui algumas informações especiais que informam ao receptor onde o quadro começa e termina; no entanto, as semelhanças se limitam a isso. Notavelmente, não é utilizado o preenchimento de bits, de modo que o comprimento de um quadro não depende dos dados enviados. Portanto, a pergunta a ser feita é: "Como o receptor sabe onde cada quadro começa e termina?". Consideramos essa questão para o link SONET de menor velocidade, conhecido como STS-1 e que opera a 51,84 Mbps. Um quadro STS-1 é mostrado na Figura 29. Ele é organizado em 9 linhas de 90 bytes cada, e os primeiros 3 bytes de cada linha são overhead, com o restante disponível para dados que estão sendo transmitidos pelo link. Os primeiros 2 bytes do

quadro contêm um padrão de bits especial, e são esses bytes que permitem ao receptor determinar onde o quadro começa. No entanto, como o preenchimento de bits não é utilizado, não há razão para que esse padrão não apareça ocasionalmente na parte de carga útil do quadro. Para se proteger contra isso, o receptor procura o padrão de bits especial consistentemente, esperando vê-lo aparecer uma vez a cada 810 bytes, já que cada quadro tem 9 × 90 = 810 bytes de comprimento. Quando o padrão especial aparece no lugar certo várias vezes, o receptor conclui que está em sincronia e pode então interpretar o quadro corretamente.

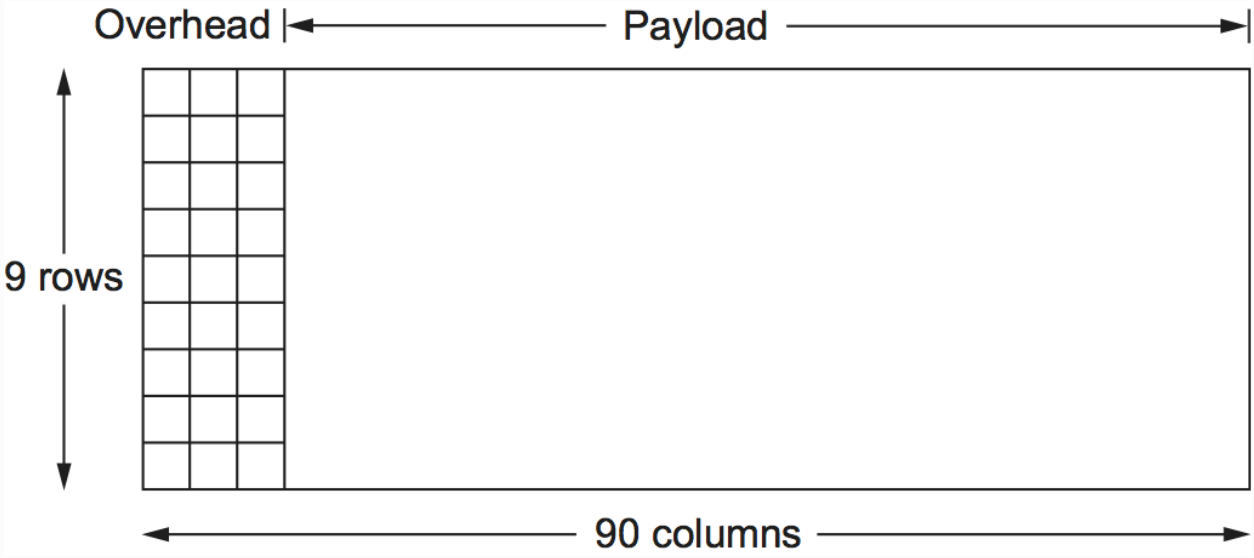

Figura 29. *Um quadro SONET STS-1.*

Um dos aspectos que não descrevemos devido à complexidade do SONET é o uso detalhado de todos os outros bytes de overhead. Parte dessa complexidade pode ser atribuída ao fato de o SONET ser executado na rede óptica da operadora, e não apenas em um único enlace. (Lembre-se de que estamos ignorando o fato de que as operadoras implementam uma rede e, em vez disso, estamos nos concentrando no fato de que podemos alugar um enlace SONET delas e, em seguida, usar esse enlace para construir nossa própria rede comutada por pacotes.) Uma complexidade adicional advém do fato de o SONET fornecer um conjunto consideravelmente mais rico de serviços do que apenas a transferência de dados. Por exemplo, 64 kbps da capacidade de um enlace SONET são reservados para um canal de voz usado para manutenção.

Os bytes de overhead de um quadro SONET são codificados usando NRZ, a codificação simples descrita na seção anterior, onde 1s são altos e 0s são baixos. No entanto, para garantir que haja transições suficientes para permitir que o receptor recupere o clock do remetente, os bytes de carga útil são *embaralhados* . Isso é feito calculando o OR exclusivo (XOR) dos dados a serem transmitidos e pelo uso de um padrão de bits bem conhecido. O padrão de bits, que tem 127 bits de comprimento, tem muitas transições de 1 a 0, de modo que a aplicação de XOR com os dados transmitidos provavelmente produzirá um sinal com transições suficientes para permitir a recuperação do clock.

O SONET suporta a multiplexação de múltiplos links de baixa velocidade da seguinte maneira. Um dado link SONET opera em uma de um conjunto finito de taxas possíveis, variando de 51,84 Mbps (STS-1) a 39.813.120 Mbps (STS-768). 1 Observe que todas essas taxas são múltiplos inteiros de STS-1. A importância para o enquadramento é que um único quadro SONET pode conter subquadros para múltiplos canais de taxa mais baixa. Uma segunda característica relacionada é que cada quadro tem 125 μs de comprimento. Isso significa que em taxas STS-1, um quadro SONET tem 810 bytes de comprimento, enquanto em taxas STS-3, cada quadro SONET tem 2.430 bytes de comprimento. Observe a sinergia entre essas duas características: 3 × 810 = 2.430, o que significa que três quadros STS-1 cabem exatamente em um único quadro STS-3.

1

STS significa *Sinal de Transporte Síncrono* , que é como a SONET se refere a quadros. Existe um termo paralelo — *Portadora Óptica* (OC) — usado para se referir ao sinal óptico subjacente que transporta quadros SONET. Dizemos que esses dois termos são paralelos porque STS-3 e OC-3, para usar um exemplo concreto, ambos implicam uma taxa de transmissão de 155,52 Mbps. Como estamos focados em enquadramento aqui, vamos nos ater ao STS, mas é mais provável que você ouça alguém se referir a um link óptico pelo seu nome "OC".

Intuitivamente, o quadro STS-N pode ser considerado como composto por N quadros STS-1, onde os bytes desses quadros são intercalados; ou seja, um byte do primeiro quadro é transmitido, depois um byte do segundo quadro é transmitido, e assim por diante. A razão para intercalar os bytes de cada quadro STS-N é garantir que os bytes em cada quadro STS-1 sejam uniformemente ritmados; ou seja, os bytes chegam ao receptor a uma taxa de 51 Mbps, em vez de todos agrupados durante uma única transmissão específica.

1/Nth

do intervalo de 125 µs.

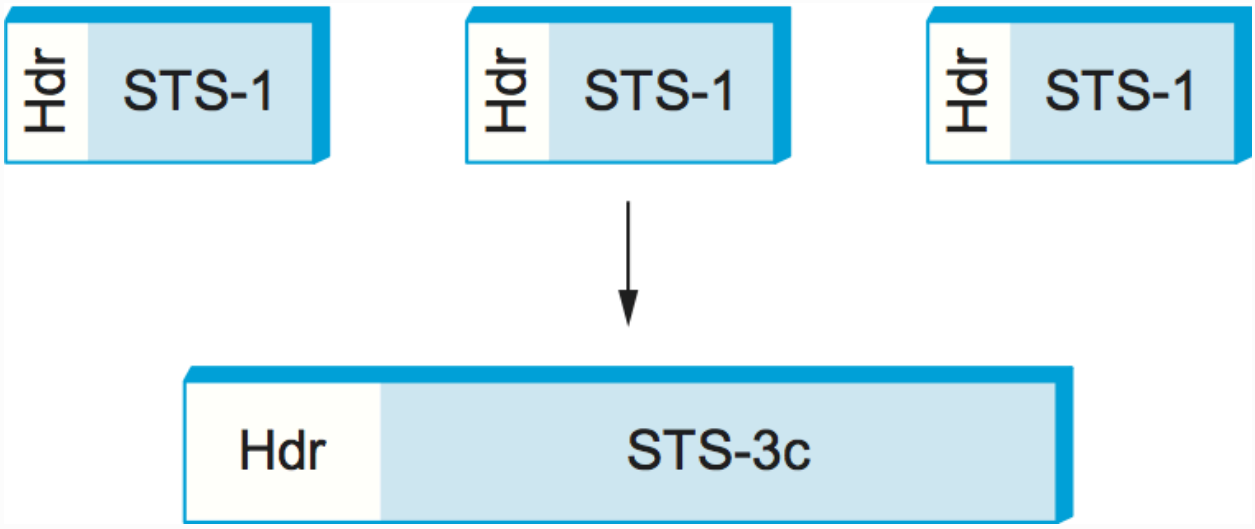

Figura 30. *Três quadros STS-1 multiplexados em um quadro STS-3c.*

Embora seja preciso visualizar um sinal STS-N como sendo usado para multiplexar N quadros STS-1, a carga útil desses quadros STS-1 pode ser vinculada para formar uma carga útil STS-N maior; tal link é denotado STS-Nc (de *concatenated*). Um dos campos no overhead é usado para esse propósito. A Figura 30 descreve esquematicamente a concatenação no caso de três quadros STS-1 sendo concatenados em um único quadro STS-3c. A importância de um link SONET ser designado como STS-3c em vez de STS-3 é que, no primeiro caso, o usuário do link pode visualizá-lo como um único

canal de 155,25 Mbps, enquanto um STS-3 deveria realmente ser visualizado como três links de 51,84 Mbps que compartilham uma fibra.

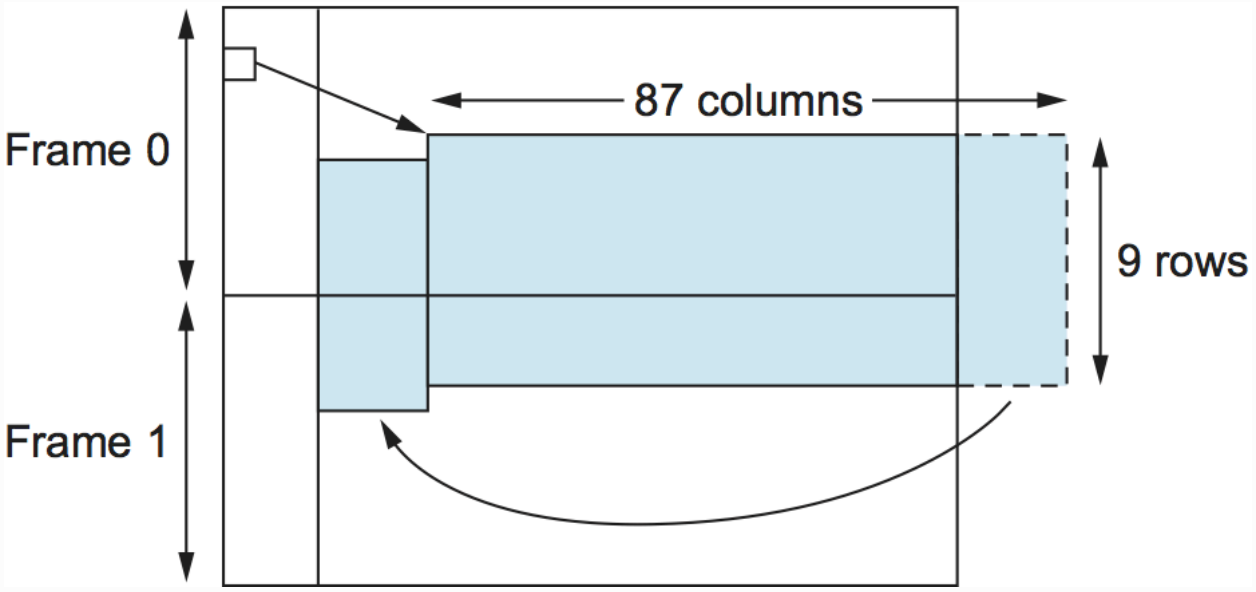

Figura 31. *Quadros SONET fora de fase.*

Finalmente, a descrição anterior do SONET é excessivamente simplista, pois pressupõe que a carga útil de cada quadro esteja completamente contida no quadro. (Por que não estaria?) Na verdade, deveríamos visualizar o quadro STS-1 descrito anteriormente como um simples espaço reservado para o quadro, onde a carga útil real pode *flutuar* entre os limites do quadro. Essa situação é ilustrada na Figura 31. Aqui, vemos tanto a carga útil STS-1 flutuando entre dois quadros STS-1 quanto a carga útil deslocada alguns bytes para a direita e, portanto, encapsulada. Um dos campos na sobrecarga do quadro aponta para o início da carga útil. O valor desse recurso é que ele simplifica a tarefa de sincronizar os relógios usados nas redes das operadoras, algo com que as operadoras gastam muito tempo se preocupando.

2.4 Detecção de erros

Conforme discutido no Capítulo 1, erros de bits às vezes são introduzidos em quadros. Isso acontece, por exemplo, devido a interferência elétrica ou ruído térmico. Embora erros sejam raros, especialmente em enlaces ópticos, algum mecanismo é necessário

para detectá-los e, assim, tomar medidas corretivas. Caso contrário, o usuário final fica se perguntando por que o programa em C, compilado com sucesso há pouco, agora apresenta um erro de sintaxe, quando tudo o que aconteceu nesse ínterim foi que ele foi copiado através de um sistema de arquivos de rede.

Há uma longa história de técnicas para lidar com erros de bits em sistemas computacionais, que remonta pelo menos à década de 1940. Os códigos de Hamming e Reed-Solomon são dois exemplos notáveis, desenvolvidos para uso em leitores de cartões perfurados, no armazenamento de dados em discos magnéticos e nas primeiras memórias de núcleo. Esta seção descreve algumas das técnicas de detecção de erros mais comumente usadas em redes.

Detectar erros é apenas uma parte do problema. A outra parte é corrigi-los após a detecção. Duas abordagens básicas podem ser adotadas quando o destinatário de uma mensagem detecta um erro. Uma delas é notificar o remetente de que a mensagem foi corrompida para que ele possa retransmitir uma cópia da mensagem. Se erros de bits forem raros, é muito provável que a cópia retransmitida esteja livre de erros. Alternativamente, alguns tipos de algoritmos de detecção de erros permitem que o destinatário reconstrua a mensagem correta mesmo após ela ter sido corrompida; tais algoritmos dependem de *códigos de correção de erros* , discutidos a seguir.

Uma das técnicas mais comuns para detectar erros de transmissão é uma técnica conhecida como *verificação de redundância cíclica* (CRC). Ela é usada em quase todos os protocolos de nível de enlace discutidos neste capítulo. Esta seção descreve o algoritmo CRC básico, mas antes de discutir essa abordagem, descrevemos primeiro o esquema *de soma de verificação* mais simples usado por vários protocolos de internet.

A ideia básica por trás de qualquer esquema de detecção de erros é adicionar informações redundantes a um quadro que possam ser usadas para determinar se

erros foram introduzidos. Em casos extremos, poderíamos imaginar a transmissão de duas cópias completas dos dados. Se as duas cópias forem idênticas no receptor, então é provável que ambas estejam corretas. Se forem diferentes, então um erro foi introduzido em uma (ou ambas) delas, e elas devem ser descartadas. Este é um esquema de detecção de erros bastante ruim por dois motivos. Primeiro, ele envia

$$n$$

bits redundantes para um

$$n$$

mensagem de bits. Em segundo lugar, muitos erros passarão despercebidos — qualquer erro que corrompa as mesmas posições de bits na primeira e na segunda cópias da mensagem. Em geral, o objetivo dos códigos de detecção de erros é fornecer uma alta probabilidade de detecção de erros combinada com um número relativamente baixo de bits redundantes.

Felizmente, podemos fazer muito melhor do que este esquema simples. Em geral, podemos fornecer uma capacidade de detecção de erros bastante robusta, enviando apenas

$$k$$

bits redundantes para um

$$n$$

mensagem de bits, onde

k

é muito menor que

n

Em uma Ethernet, por exemplo, um quadro transportando até 12.000 bits (1.500 bytes) de dados requer apenas um código CRC de 32 bits, ou como é comumente expresso, usa CRC-32. Esse código detectará a grande maioria dos erros, como veremos a seguir.

Dizemos que os bits extras que enviamos são redundantes porque não acrescentam nenhuma informação nova à mensagem. Em vez disso, eles são derivados diretamente da mensagem original usando um algoritmo bem definido. Tanto o remetente quanto o destinatário sabem exatamente qual é esse algoritmo. O remetente aplica o algoritmo à mensagem para gerar os bits redundantes. Em seguida, ele transmite a mensagem e esses poucos bits extras. Quando o destinatário aplica o mesmo algoritmo à mensagem recebida, ele deve (na ausência de erros) obter o mesmo resultado que o remetente. Ele compara o resultado com o que lhe foi enviado pelo remetente. Se eles corresponderem, ele pode concluir (com alta probabilidade) que nenhum erro foi introduzido na mensagem durante a transmissão. Se não corresponderem, ele pode ter certeza de que a mensagem ou os bits redundantes foram corrompidos e deve tomar as medidas apropriadas — ou seja, descartar a mensagem ou corrigi-la, se possível.

Uma observação sobre a terminologia para esses bits extras. Em geral, eles são chamados de *códigos de detecção de erros* . Em casos específicos, quando o algoritmo para criar o código é baseado em adição, eles podem ser chamados de *soma de verificação* . Veremos que a soma de verificação da Internet tem um nome apropriado: é uma verificação de erros que usa um algoritmo de soma. Infelizmente, a

palavra *soma de verificação* é frequentemente usada de forma imprecisa para significar qualquer forma de código de detecção de erros, incluindo CRCs. Isso pode ser confuso, por isso recomendamos que você use a palavra *soma de verificação* apenas para se aplicar a códigos que realmente usam adição e que use *código de detecção de erros* para se referir à classe geral de códigos descritos nesta seção.

2.4.1 Algoritmo de soma de verificação da Internet

Nossa primeira abordagem para detecção de erros é exemplificada pela soma de verificação da Internet. Embora não seja usada no nível do link, ela oferece o mesmo tipo de funcionalidade que os CRCs, por isso a discutiremos aqui.

A ideia por trás do checksum da internet é muito simples: você soma todas as palavras transmitidas e, em seguida, transmite o resultado dessa soma. O resultado é o checksum. O receptor realiza o mesmo cálculo nos dados recebidos e compara o resultado com o checksum recebido. Se algum dado transmitido, incluindo o próprio checksum, estiver corrompido, os resultados não corresponderão, então o receptor sabe que ocorreu um erro.

Você pode imaginar muitas variações diferentes da ideia básica de uma soma de verificação. O esquema exato usado pelos protocolos de internet funciona da seguinte maneira. Considere os dados que estão sendo verificados como uma sequência de números inteiros de 16 bits. Some-os usando a aritmética de complemento para unidades de 16 bits (explicada abaixo) e, em seguida, calcule o complemento para unidades do resultado. Esse número de 16 bits é a soma de verificação.

Na aritmética de complemento de uns, um inteiro negativo (-x) é representado como o complemento de x; isto é, cada bit de x é invertido. Ao somar números na aritmética de complemento de uns, um carryout do bit mais significativo precisa ser adicionado ao resultado. Considere, por exemplo, a adição de -5 e -2 na aritmética de complemento

de uns em inteiros de 4 bits: +5 é 0101, então -5 é 1010; +2 é 0010, então -2 é 1101. Se somarmos 1010 e 1101, ignorando o carry, obtemos 0111. Na aritmética de complemento de uns, o fato de que essa operação causou um carry do bit mais significativo nos faz incrementar o resultado, dando 1000, que é a representação em complemento de uns de -7 (obtida pela inversão dos bits em 0111), como seria de se esperar.

A rotina a seguir fornece uma implementação direta do algoritmo de soma de verificação da internet. O `count`argumento fornece o comprimento de , `buf`medido em unidades de 16 bits. A rotina assume que `buf` já foi preenchido com 0s até um limite de 16 bits.

```
u_short
cksum(u_short *buf, int count)
{
    register u_long sum = 0;

    while (count--)
    {
        sum += *buf++;
        if (sum & 0xFFFF0000)
        {
            /* carry occurred, so wrap around */
            sum &= 0xFFFF;
            sum++;
        }
    }
    return ~(sum & 0xFFFF);
}
```

Este código garante que o cálculo utilize a aritmética de complemento de um em vez do complemento de dois, usado na maioria das máquinas. Observe a `if`instrução dentro do `while`loop. Se houver um transporte para os 16 bits superiores de `sum`, então incrementamos `sum`exatamente como no exemplo anterior.

Comparado ao nosso código de repetição, este algoritmo tem uma boa pontuação por usar um pequeno número de bits redundantes — apenas 16 para uma mensagem de qualquer tamanho —, mas não tem uma pontuação extremamente alta em termos de

força de detecção de erros. Por exemplo, um par de erros de bit único, um dos quais incrementa uma palavra e o outro decrementa outra palavra na mesma quantidade, não será detectado. A razão para usar um algoritmo como este, apesar de sua proteção relativamente fraca contra erros (em comparação com um CRC, por exemplo), é simples: este algoritmo é muito mais fácil de implementar em software. A experiência sugere que uma soma de verificação deste formato era adequada, mas uma das razões pelas quais é adequada é que esta soma de verificação é a última linha de defesa em um protocolo ponta a ponta. A maioria dos erros é detectada por algoritmos de detecção de erros mais fortes, como CRCs, no nível do enlace.

2.4.2 Verificação de redundância cíclica

Já deve estar claro que um dos principais objetivos do projeto de algoritmos de detecção de erros é maximizar a probabilidade de detecção de erros usando apenas um pequeno número de bits redundantes. As verificações de redundância cíclica utilizam uma matemática bastante poderosa para atingir esse objetivo. Por exemplo, um CRC de 32 bits oferece forte proteção contra erros de bits comuns em mensagens com milhares de bytes de comprimento. A base teórica da verificação de redundância cíclica está enraizada em um ramo da matemática chamado *campos finitos* . Embora isso possa parecer assustador, os conceitos básicos podem ser facilmente compreendidos.

Para começar, pense em uma mensagem (n+1)-bit como sendo representada por um

$$n$$

polinômio de grau, ou seja, um polinômio cujo termo de ordem mais alta é

$$xn$$

A mensagem é representada por um polinômio usando o valor de cada bit da mensagem como coeficiente para cada termo do polinômio, começando pelo bit mais

significativo para representar o termo de ordem mais alta. Por exemplo, uma mensagem de 8 bits composta pelos bits 10011010 corresponde ao polinômio

$$M(x)=(1\times x7)+(0\times x6)+(0\times x5)+(1\times x4)+(1\times x3)+(0\times x2)+(1\times x1)+(0\times x0)$$

$$M(x)=x7+x4+x3+x1$$

Podemos, portanto, pensar em um emissor e um receptor trocando polinômios entre si.

Para fins de cálculo de um CRC, um remetente e um destinatário devem concordar com um polinômio *divisor* ,

$C(x)$

.

$C(x)$

é um polinômio de grau

k

. Por exemplo, suponha

$C(x)=x3+x2+1$

. Nesse caso,

$k=3$

. A resposta à pergunta “Onde é que

$C(x)$

vem?” é, na maioria dos casos práticos, “Você procura em um livro”. Na verdade, a escolha de

$$C(x)$$

tem um impacto significativo nos tipos de erros que podem ser detectados com confiabilidade, como discutiremos a seguir. Há um punhado de polinômios divisores que são escolhas muito boas para diversos ambientes, e a escolha exata normalmente é feita como parte do projeto do protocolo. Por exemplo, o padrão Ethernet usa um polinômio bem conhecido de grau 32.

Quando um remetente deseja transmitir uma mensagem

$$M(x)$$

que tem n+1 bits de comprimento, o que é realmente enviado é a mensagem de (n+1) bits mais

$$k$$

bits. Chamamos a mensagem completa transmitida, incluindo os bits redundantes,

$$P(x)$$

O que vamos fazer é inventar para fazer o polinômio que representa

$$P(x)$$

exatamente divisível por

$$C(x)$$

; explicamos como isso é alcançado a seguir. Se

$$P(x)$$

é transmitido por um link e não há erros introduzidos durante a transmissão, então o receptor deve ser capaz de dividir

$$P(x)$$

por

$$C(x)$$

exatamente, deixando um resto de zero. Por outro lado, se algum erro for introduzido em

$$P(x)$$

durante a transmissão, então, com toda a probabilidade, o polinômio recebido não será mais exatamente divisível por

$$C(x)$$

, e assim o receptor obterá um resto diferente de zero, o que implica que ocorreu um erro.

Será útil entender o seguinte se você souber um pouco sobre aritmética polinomial; ela é apenas ligeiramente diferente da aritmética de inteiros normal. Estamos lidando com uma classe especial de aritmética polinomial aqui, onde os coeficientes podem ser apenas um ou zero, e as operações sobre os coeficientes são realizadas usando aritmética de módulo 2. Isso é chamado de "aritmética polinomial módulo 2". Como este é um livro de redes, não um texto de matemática, vamos nos concentrar nas

principais propriedades desse tipo de aritmética para nossos propósitos (que pedimos que você aceite com fé):

- Qualquer polinômio
- B(x)
- pode ser dividido por um polinômio divisor
- C(x)
- se
- B(x)
- é de grau mais elevado do que
- C(x)
- .
- Qualquer polinômio
- B(x)
- pode ser dividido uma vez por um polinômio divisor
- C(x)
- se
- B(x)
- é do mesmo grau que
- C(x)
- .
- O restante obtido quando
- B(x)
- é dividido por
- C(x)
- é obtido executando a operação OU exclusiva (XOR) em cada par de coeficientes correspondentes.

Por exemplo, o polinômio

$$x3+1$$

pode ser dividido por

$$x3+x2+1$$

(porque ambos são de grau 3) e o restante seria

$$0\times x3+1\times x2+0\times x1+0\times x0=x2$$

(obtido pela operação XOR dos coeficientes de cada termo). Em termos de mensagens, poderíamos dizer que 1001 pode ser dividido por 1101 e deixa um resto de 0100. Você deve conseguir ver que o resto é apenas o OU exclusivo bit a bit das duas mensagens.

Agora que conhecemos as regras básicas para a divisão de polinômios, podemos realizar a divisão longa, necessária para lidar com mensagens mais longas. Um exemplo segue abaixo.

Lembre-se de que queríamos criar um polinômio para transmissão derivado da mensagem original

$$M(x)$$

, é

$$k$$

bits mais longos que

$$M(x)$$

, e é exatamente divisível por

$C(x)$

. Podemos fazer isso da seguinte maneira:

1. Multiplicar
2. $M(x)$
3. por
4. x^k
5. ; isto é, adicionar
6. k
7. zeros no final da mensagem. Chame esta mensagem de extensão zero
8. $T(x)$
9. .
10. Dividir
11. $T(x)$
12. por
13. $C(x)$
14. e encontre o resto.
15. Subtraia o restante de
16. $T(x)$
17. .

Deveria ser óbvio que o que resta neste ponto é uma mensagem que é exatamente divisível por

$C(x)$

. Podemos também notar que a mensagem resultante consiste em

$M(x)$

seguido pelo restante obtido no passo 2, pois quando subtraímos o restante (que não pode ser maior que

k

bits de comprimento), estávamos apenas fazendo XOR com o

k

zeros adicionados na etapa 1. Esta parte ficará mais clara com um exemplo.

Considere a mensagem

$x7+x4+x3+x1$

, ou 10011010. Começamos multiplicando por

$x3$

, já que nosso polinômio divisor é de grau 3. Isso dá 10011010000. Dividimos isso por

$C(x)$

, que corresponde a 1101 neste caso. A Figura 32 mostra a operação de divisão longa polinomial. Dadas as regras da aritmética polinomial descritas acima, a operação de divisão longa ocorre de forma muito semelhante à divisão de números inteiros. Assim, na primeira etapa do nosso exemplo, vemos que o divisor 1101 divide uma vez os quatro primeiros bits da mensagem (1001), uma vez que são do mesmo grau, e deixa um resto de 100 (1101 XOR 1001). A próxima etapa é reduzir um dígito do polinômio da mensagem até obtermos outro polinômio com o mesmo grau que

C(x)

, neste caso 1001. Calculamos o resto novamente (100) e continuamos até que o cálculo esteja completo. Observe que o "resultado" da divisão longa, que aparece no topo do cálculo, não é realmente de grande interesse — o que importa é o resto no final.

Você pode ver na parte inferior da Figura 32 que o restante do cálculo do exemplo é 101. Portanto, sabemos que 10011010000 menos 101 seria exatamente divisível por

C(x)

, e é isso que enviamos. A operação de menos na aritmética polinomial é a operação lógica XOR, então, na verdade, enviamos 10011010101. Como observado acima, esta é apenas a mensagem original com o restante do cálculo da divisão longa anexado a ela. O destinatário divide o polinômio recebido por

C(x)

e, se o resultado for 0, conclui-se que não houve erros. Se o resultado for diferente de zero, pode ser necessário descartar a mensagem corrompida; com alguns códigos, pode ser possível *corrigir* um pequeno erro (por exemplo, se o erro afetou apenas um bit). Um código que permite a correção de erros é chamado de *código de correção de erros* (ECC).

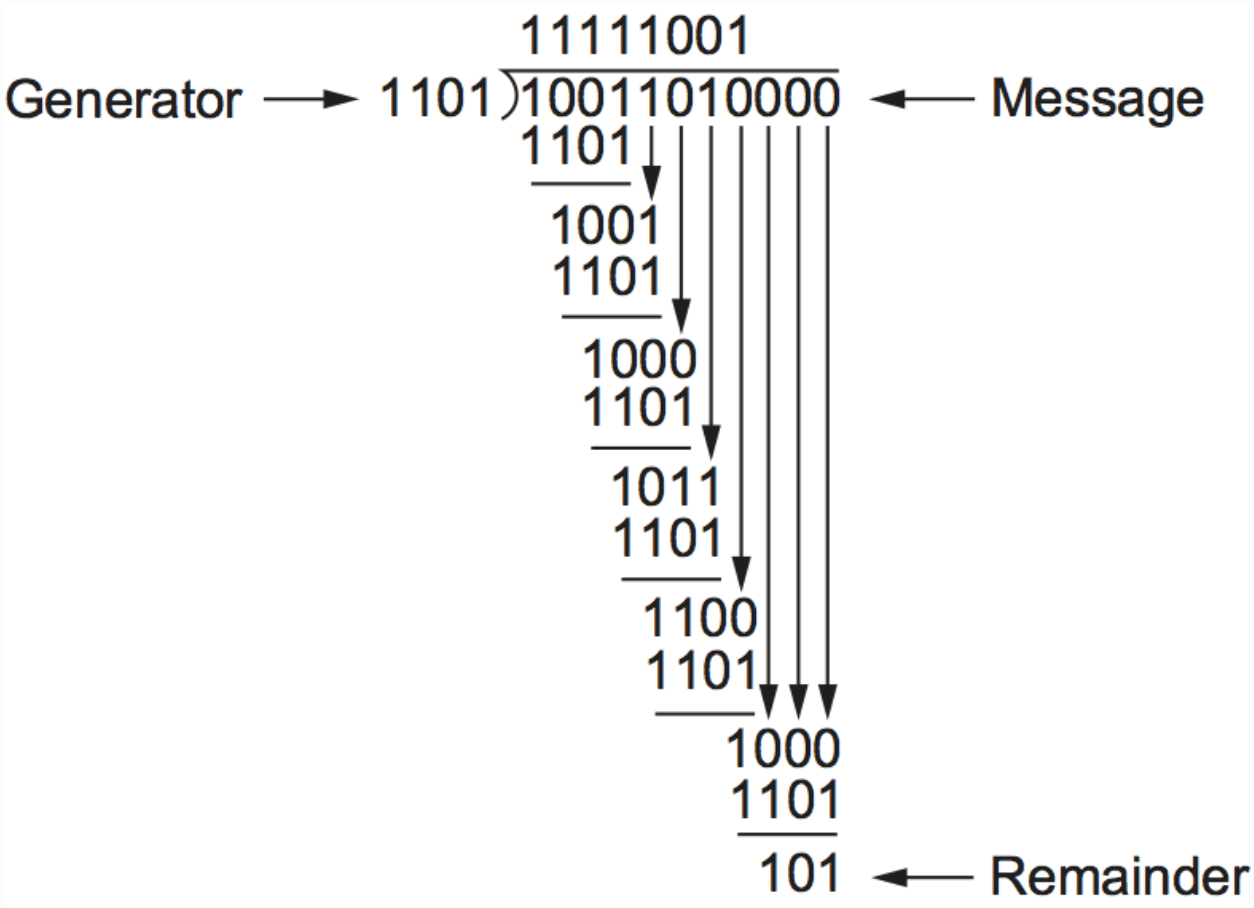

Figura 32. *Cálculo de CRC usando divisão longa polinomial.*

Agora vamos considerar a questão de onde o polinômio

$$C(x)$$

vem de. Intuitivamente, a ideia é selecionar esse polinômio de forma que seja muito improvável que ele se divida uniformemente em uma mensagem que contenha erros introduzidos. Se a mensagem transmitida for

$$P(x)$$

, podemos pensar na introdução de erros como a adição de outro polinômio

$$E(x)$$

, para que o destinatário veja

$$P(x)+E(x)$$

. A única maneira de um erro passar despercebido seria se a mensagem recebida pudesse ser dividida igualmente por

$$C(x)$$

, e já que sabemos que

$$P(x)$$

pode ser dividido igualmente por

$$C(x)$$

, isso só poderia acontecer se

$$E(x)$$

pode ser dividido igualmente por

$$C(x)$$

O truque é escolher

$$C(x)$$

o que torna isso muito improvável para tipos comuns de erros.

Um tipo comum de erro é um erro de bit único, que pode ser expresso como

E(x)=xi

quando afeta a posição do bit *i* . Se selecionarmos

C(x)

de modo que o primeiro e o último termo (ou seja, o

xk

e

x0

termos) são diferentes de zero, então já temos um polinômio de dois termos que não pode ser dividido uniformemente em um termo

E(x)

. Tal

C(x)

pode, portanto, detectar todos os erros de bit único. Em geral, é possível provar que os seguintes tipos de erros podem ser detectados por um

C(x)

com as propriedades declaradas:

- Todos os erros de bit único, desde que o
- xk

- e
- x0
- os termos têm coeficientes diferentes de zero
- Todos os erros de bit duplo, desde que
- C(x)
- tem um fator com pelo menos três termos
- Qualquer número ímpar de erros, desde que
- C(x)
- contém o fator
- (x+1)
- Qualquer erro de “explosão” (ou seja, sequência de bits consecutivos com erros) para o qual o comprimento da explosão é menor que
- k
- bits (a maioria dos erros de burst de comprimento maior que
- k
- bits também podem ser detectados.)

Seis versões de

C(x)

são amplamente utilizados em protocolos de nível de link. Por exemplo, a Ethernet utiliza CRC-32, que é definido da seguinte forma:

- CRC-32 =
- x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x+1

Mencionamos que é possível usar códigos que não apenas detectam a presença de erros, mas também permitem que eles sejam corrigidos. Como os detalhes desses códigos exigem matemática ainda mais complexa do que a necessária para entender

os CRCs, não nos deteremos neles aqui. No entanto, vale a pena considerar os méritos da correção versus detecção.

À primeira vista, parece que a correção é sempre melhor, já que com a detecção somos forçados a descartar a mensagem e, em geral, solicitar que outra cópia seja transmitida. Isso consome largura de banda e pode introduzir latência enquanto se espera pela retransmissão. No entanto, há uma desvantagem na correção, pois geralmente requer um número maior de bits redundantes para enviar um código de correção de erros que seja tão forte (ou seja, capaz de lidar com a mesma gama de erros) quanto um código que apenas detecta erros. Assim, enquanto a detecção de erros requer o envio de mais bits quando ocorrem erros, a correção de erros requer o envio de mais bits *o tempo todo* . Como resultado, a correção de erros tende a ser mais útil quando (1) os erros são bastante prováveis, como podem ser, por exemplo, em um ambiente sem fio, ou (2) o custo da retransmissão é muito alto, por exemplo, devido à latência envolvida na retransmissão de um pacote por um link de satélite.

O uso de códigos de correção de erros em redes é às vezes chamado de *correção antecipada de erros* (FEC), pois a correção de erros é realizada "antecipadamente" pelo envio de informações adicionais, em vez de esperar que os erros aconteçam e tratá-los posteriormente por retransmissão. A FEC é comumente usada em redes sem fio, como a 802.11.

Por fim, notamos que o algoritmo CRC, embora aparentemente complexo, é facilmente implementado em hardware usando um

k

registrador de deslocamento de bits e portas XOR. O número de bits no registrador de deslocamento é igual ao grau do polinômio gerador (

k

). A Figura 33 mostra o hardware que seria utilizado para o gerador

$$x3+x2+1$$

do nosso exemplo anterior. A mensagem é deslocada da esquerda para a direita, começando com o bit mais significativo e terminando com a sequência de

$$k$$

zeros que são anexados à mensagem, assim como no exemplo da divisão longa. Quando todos os bits foram deslocados e submetidos a um XOR apropriado, o registrador contém o restante — ou seja, o CRC (bit mais significativo à direita). A posição das portas XOR é determinada da seguinte forma: se os bits no registrador de deslocamento forem rotulados de 0 a

$$k−1$$

, da esquerda para a direita, então coloque uma porta XOR na frente do bit

$$n$$

se houver um termo

$$xn$$

no polinômio gerador. Assim, vemos uma porta XOR na frente das posições 0 e 2 para o polinômio gerador

$$x3+x2+x0$$

.

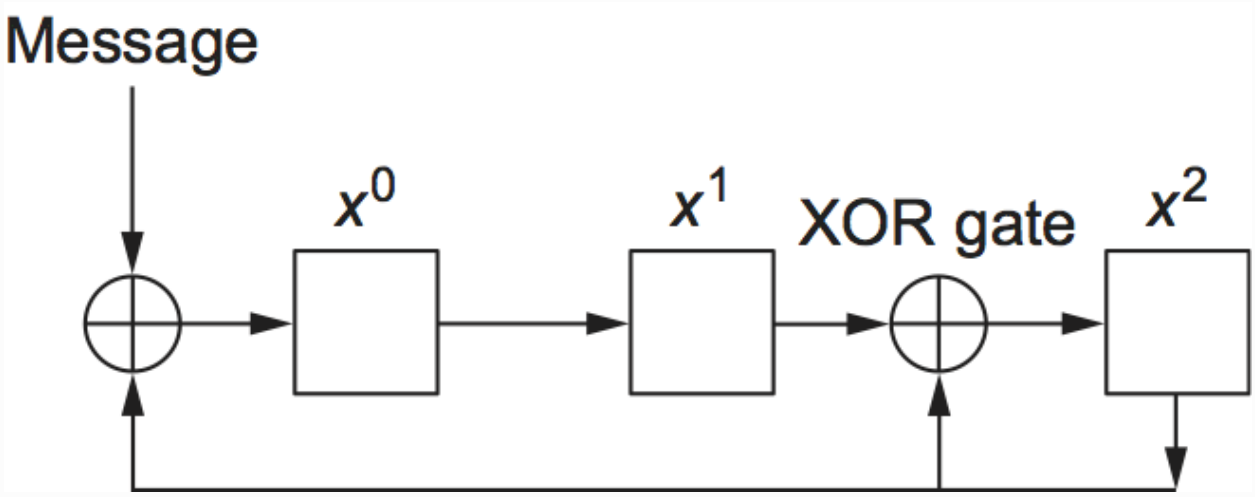

Figura 33. *Cálculo de CRC usando registrador de deslocamento.*

2.5 Transmissão confiável

Como vimos na seção anterior, os quadros às vezes são corrompidos durante o trânsito, sendo um código de erro como o CRC usado para detectar tais erros. Embora alguns códigos de erro sejam fortes o suficiente para corrigir erros, na prática, a sobrecarga costuma ser grande demais para lidar com a gama de erros de bits e rajadas que podem ser introduzidos em um enlace de rede. Mesmo quando códigos de correção de erros são usados (por exemplo, em enlaces sem fio), alguns erros serão graves demais para serem corrigidos. Como resultado, alguns quadros corrompidos precisam ser descartados. Um protocolo em nível de enlace que deseja entregar quadros de forma confiável precisa, de alguma forma, se recuperar desses quadros descartados (perdidos).

Vale ressaltar que a confiabilidade é uma função que *pode* ser fornecida no nível do enlace, mas muitas tecnologias de enlace modernas omitem essa função. Além disso, a entrega confiável é frequentemente fornecida em níveis mais altos, incluindo tanto o transporte quanto, às vezes, a camada de aplicação. Exatamente onde ela deve ser fornecida é uma questão controversa e depende de muitos fatores. Descrevemos aqui

os princípios básicos da entrega confiável, visto que os princípios são comuns a todas as camadas, mas você deve estar ciente de que não estamos falando apenas de uma função da camada de enlace.

Uma entrega confiável geralmente é realizada usando uma combinação de dois mecanismos fundamentais — *confirmações* e *timeouts* . Uma confirmação (ACK, abreviado) é um pequeno quadro de controle que um protocolo envia de volta ao seu par informando que recebeu um quadro anterior. Por quadro de controle, queremos dizer um cabeçalho sem dados, embora um protocolo possa *adicionar* um ACK a um quadro de dados, ele simplesmente está enviando na direção oposta. O recebimento de uma confirmação indica ao remetente do quadro original que seu quadro foi entregue com sucesso. Se o remetente não receber uma confirmação após um período de tempo razoável, ele *retransmite* o quadro original. Essa ação de esperar um período de tempo razoável é chamada de *timeout* .

A estratégia geral de usar confirmações e timeouts para implementar entrega confiável é às vezes chamada de *solicitação de repetição automática* (ARQ, abreviação). Esta seção descreve três algoritmos ARQ diferentes usando linguagem genérica; ou seja, não fornecemos informações detalhadas sobre os campos de cabeçalho de um protocolo específico.

2.5.1 Parar e esperar

O esquema ARQ mais simples é o algoritmo *stop-and-wait* . A ideia do stop-and-wait é direta: após transmitir um quadro, o remetente aguarda uma confirmação antes de transmitir o próximo quadro. Se a confirmação não chegar após um determinado período de tempo, o remetente expira e retransmite o quadro original.

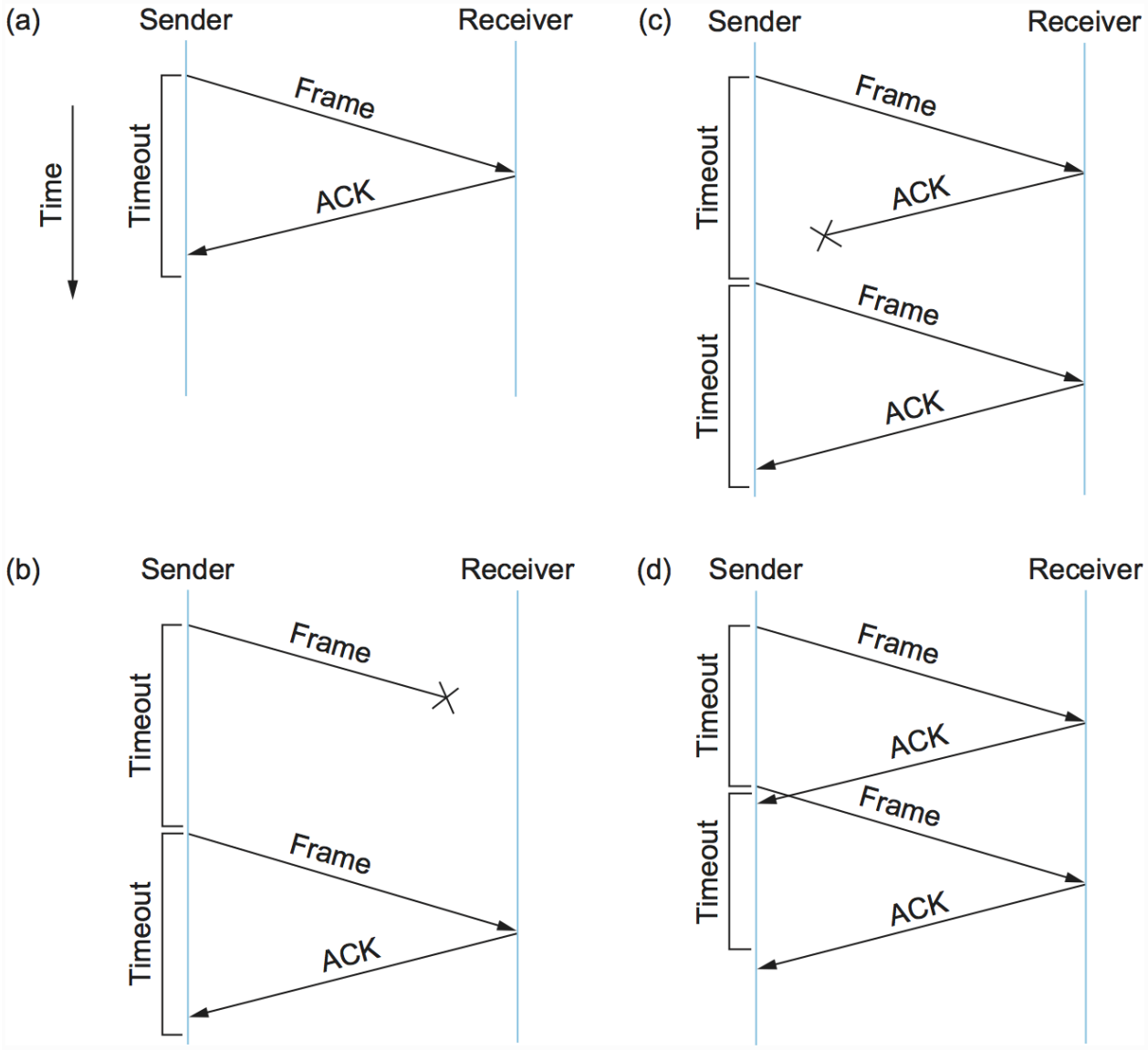

Figura 34. *Linha do tempo mostrando quatro cenários diferentes para o algoritmo stop-and-wait. (a) O ACK é recebido antes que o temporizador expire; (b) o quadro original é perdido; (c) o ACK é perdido; (d) o tempo limite dispara muito cedo.*

A Figura 34 ilustra linhas do tempo para quatro cenários diferentes resultantes desse algoritmo básico. O lado emissor é representado à esquerda, o lado receptor é representado à direita, e o tempo flui de cima para baixo. A Figura 34(a) mostra a situação em que o ACK é recebido antes do tempo expirar; (b) e (c) mostram a situação em que o quadro original e o ACK, respectivamente, são perdidos; e (d) mostra a situação em que o tempo limite é disparado muito cedo. Lembre-se de que por "perdido" queremos dizer que o quadro foi corrompido durante o trânsito, que essa

corrupção foi detectada por um código de erro no receptor e que o quadro foi posteriormente descartado.

As linhas do tempo dos pacotes mostradas nesta seção são exemplos de uma ferramenta frequentemente utilizada no ensino, explicação e projeto de protocolos. Elas são úteis porque capturam visualmente o comportamento de um sistema distribuído ao longo do tempo — algo que pode ser bastante difícil de analisar. Ao projetar um protocolo, muitas vezes é preciso estar preparado para o inesperado — uma falha do sistema, uma mensagem se perde ou algo que se esperava que acontecesse rapidamente acaba demorando muito. Esses tipos de diagramas podem frequentemente nos ajudar a entender o que pode dar errado nesses casos e, assim, ajudar o projetista de protocolos a estar preparado para qualquer eventualidade.

Há uma sutileza importante no algoritmo stop-and-wait. Suponha que o remetente envie um quadro e o destinatário o reconheça, mas a confirmação seja perdida ou atrase a chegada. Essa situação é ilustrada nas linhas de tempo (c) e (d) da Figura 34. Em ambos os casos, o remetente expira e retransmite o quadro original, mas o destinatário pensará que é o próximo quadro, uma vez que recebeu e reconheceu corretamente o primeiro quadro. Isso tem o potencial de causar a entrega de cópias duplicadas de um quadro. Para resolver esse problema, o cabeçalho de um protocolo stop-and-wait geralmente inclui um número de sequência de 1 bit — ou seja, o número de sequência pode assumir os valores 0 e 1 — e os números de sequência usados para cada quadro se alternam, conforme ilustrado na Figura 35 . Assim, quando o remetente retransmite o quadro 0, o receptor pode determinar que está vendo uma segunda cópia do quadro 0 em vez da primeira cópia do quadro 1 e, portanto, pode ignorá-la (o receptor ainda a reconhece, caso o primeiro ACK tenha sido perdido).

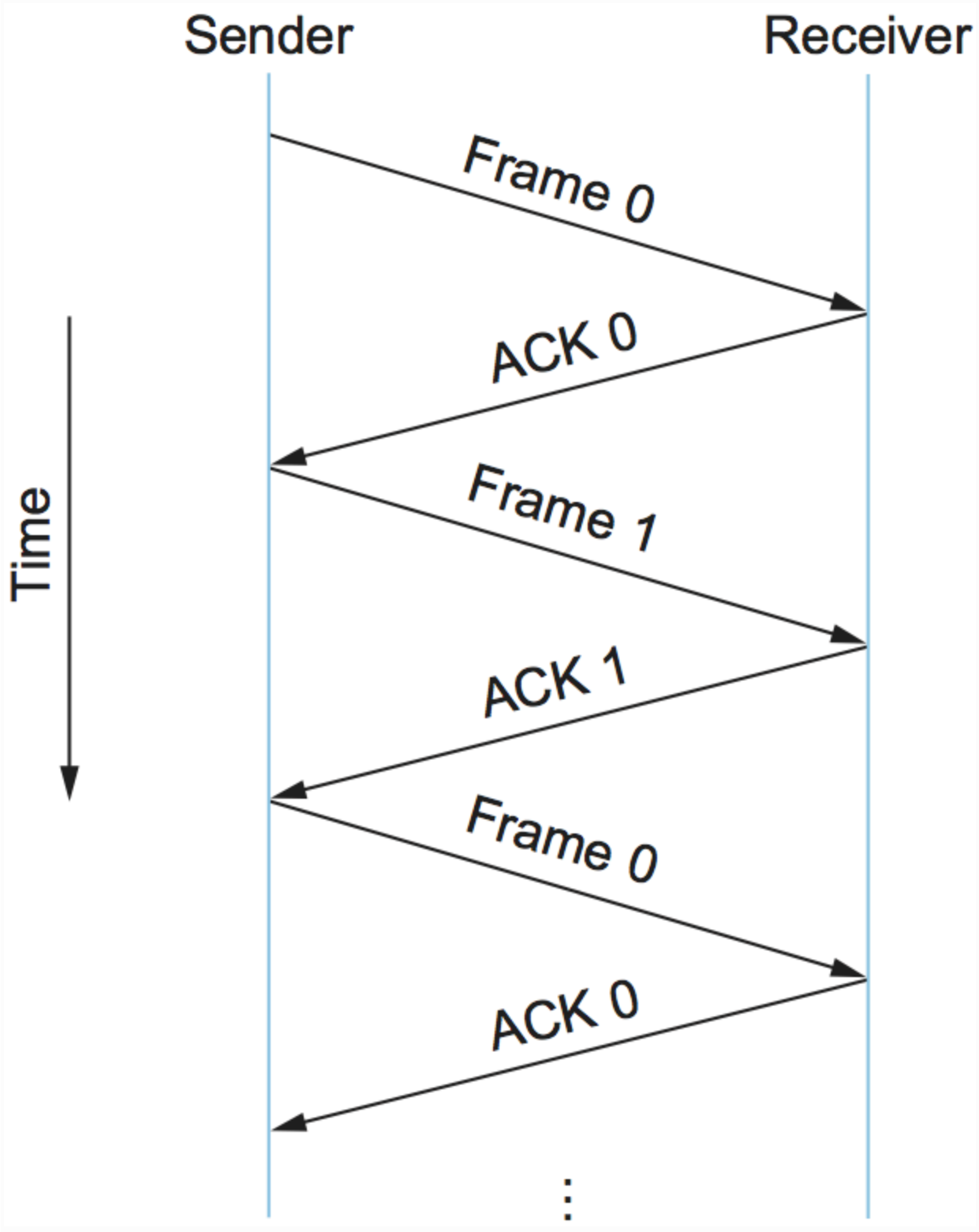

Figura 35. *Linha do tempo para parar e esperar com número de sequência de 1 bit.*

A principal deficiência do algoritmo stop-and-wait é que ele permite que o remetente tenha apenas um quadro pendente no enlace por vez, e este pode estar muito abaixo da capacidade do enlace. Considere, por exemplo, um enlace de 1,5 Mbps com um tempo de ida e volta de 45 ms. Este enlace tem um produto atraso x largura de banda de 67,5 Kb, ou aproximadamente 8 KB. Como o remetente pode enviar apenas um quadro por RTT, e assumindo um tamanho de quadro de 1 KB, isso implica uma taxa máxima de envio de

Bits por quadro / Tempo por quadro = 1024 x 8 / 0,045 = 182 kbps

ou cerca de um oitavo da capacidade do link. Para usar o link ao máximo, gostaríamos que o remetente pudesse transmitir até oito quadros antes de ter que esperar por uma confirmação.

Conclusão principal

A importância do produto atraso × largura de banda é que ele representa a quantidade de dados que poderia estar em trânsito. Gostaríamos de poder enviar essa quantidade de dados sem esperar pela primeira confirmação. O princípio em vigor aqui é frequentemente chamado de " *manter o canal cheio"* . Os algoritmos apresentados nas duas subseções a seguir fazem exatamente isso. [Próximo]

2.5.2 Janela deslizante

Considere novamente o cenário em que o link tem um produto atraso x largura de banda de 8 KB e os quadros têm 1 KB de tamanho. Gostaríamos que o remetente estivesse pronto para transmitir o nono quadro praticamente no mesmo momento em que o ACK do primeiro quadro chega. O algoritmo que nos permite fazer isso é chamado de *janela deslizante* , e um cronograma ilustrativo é apresentado na Figura 36 .

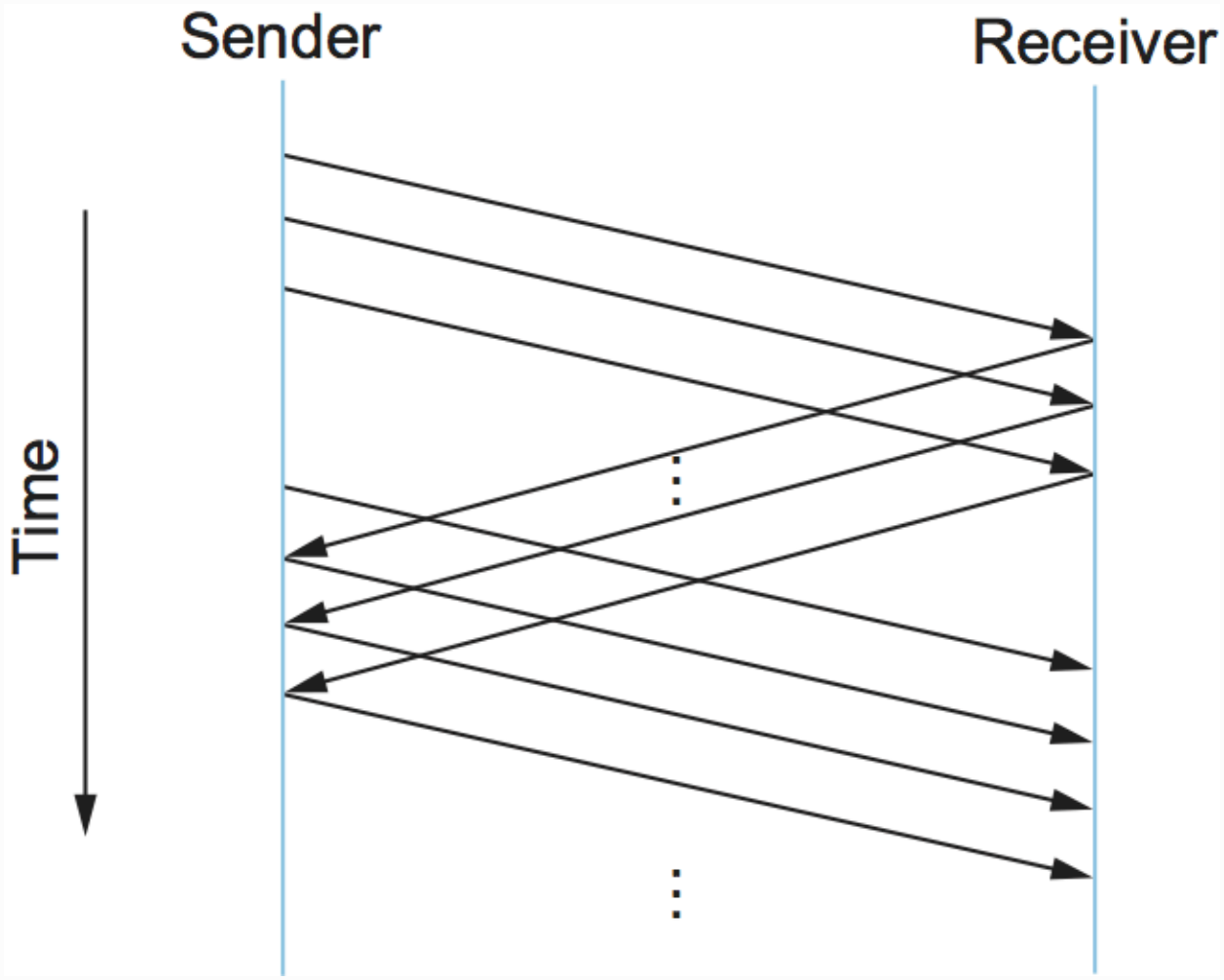

Figura 36. *Linha do tempo para o algoritmo de janela deslizante.*

O Algoritmo da Janela Deslizante

O algoritmo de janela deslizante funciona da seguinte maneira. Primeiro, o remetente atribui um *número de sequência* , denotado `SeqNum`, a cada quadro. Por enquanto, vamos ignorar o fato de que `SeqNum`é implementado por um campo de cabeçalho de tamanho finito e, em vez disso, assumir que ele pode crescer infinitamente. O remetente mantém três variáveis: O *tamanho da janela de envio* , denotado `SWS`, fornece o limite superior para o número de quadros pendentes (não confirmados) que o remetente pode transmitir; `LAR`denota o número de sequência da *última confirmação recebida* ; e `LFS`denota o número de sequência do *último quadro enviado* . O remetente também mantém a seguinte invariante:

```
LFS - LAR <= SWS
```

Esta situação é ilustrada na Figura 37 .

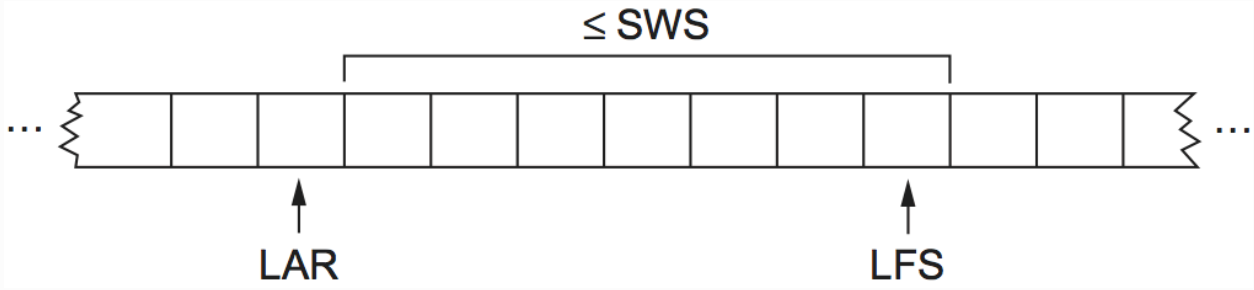

Figura 37. *Janela deslizante no remetente.*

Quando uma confirmação chega, o remetente se move `LAR`para a direita, permitindo assim que transmita outro quadro. Além disso, o remetente associa um temporizador a cada quadro transmitido e retransmite o quadro caso o temporizador expire antes de um ACK ser recebido. Observe que o remetente precisa estar disposto a armazenar em buffer até `SWS`os quadros, pois deve estar preparado para retransmiti-los até que sejam confirmados.

O receptor mantém as três variáveis a seguir: O *tamanho da janela de recepção* , denotado por `RWS`, fornece o limite superior para o número de quadros fora de ordem que o receptor está disposto a aceitar; `LAF` denota o número de sequência do *maior quadro aceitável* ; e `LFR`denota o número de sequência do *último quadro recebido* . O receptor também mantém a seguinte invariante:

```
LAF - LFR <= RWS
```

Esta situação é ilustrada na Figura 38 .

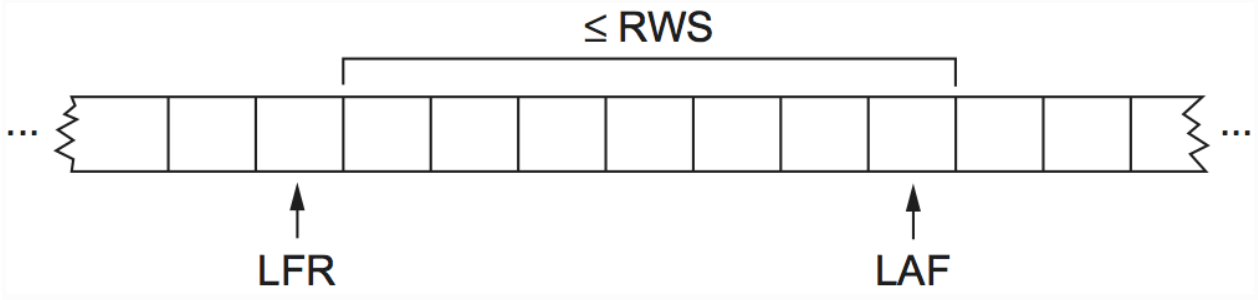

Figura 38. *Janela deslizante no receptor.*

Quando um quadro com número de sequência `SeqNum`chega, o receptor realiza a seguinte ação. Se ou , então o quadro está fora da janela do receptor e é descartado. Se , então o quadro está dentro da janela do receptor e é aceito. Agora, o receptor precisa decidir se envia ou não um ACK. Seja denotar o maior número de sequência ainda não confirmado, de forma que todos os quadros com números de sequência menores ou iguais a tenham sido recebidos. O receptor confirma o recebimento de , mesmo que pacotes com numeração mais alta tenham sido recebidos. Essa confirmação é dita cumulativa. Em seguida, ele define e ajusta .`SeqNum <= LFRSeqNum > LAFLFR < SeqNum <= LAFSeqNumToAckSeqNumToAckSeqNumToAckLFR = SeqNumToAckLAF = LFR + RWS`

Por exemplo, suponha (ou seja, o último ACK que o receptor enviou foi para o número de sequência 5), e . Isso implica que . Se os quadros 7 e 8 chegarem, eles serão armazenados em buffer porque estão dentro da janela do receptor. No entanto, nenhum ACK precisa ser enviado, pois o quadro 6 ainda não chegou. Diz-se que os quadros 7 e 8 chegaram fora de ordem. (Tecnicamente, o receptor poderia reenviar um ACK para o quadro 5 quando os quadros 7 e 8 chegarem.) Se o quadro 6 chegar — talvez esteja atrasado porque foi perdido na primeira vez e teve que ser retransmitido, ou talvez tenha sido simplesmente atrasado — o receptor reconhece o quadro 8, salta para 8 e define como 12. 1 Se o quadro 6 foi de fato perdido, então um tempo limite terá ocorrido no remetente, fazendo com que ele retransmita o quadro 6.`LFR = 5RWS = 4LAF = 9LFRLAF`

1

Embora seja improvável que um pacote possa ser atrasado ou chegar fora de ordem em um link ponto a ponto, esse mesmo algoritmo é usado em conexões multi-hop onde tais atrasos são possíveis.

Observamos que, quando ocorre um timeout, a quantidade de dados em trânsito diminui, pois o remetente não consegue avançar sua janela até que o quadro 6 seja reconhecido. Isso significa que, quando ocorrem perdas de pacotes, esse esquema não mantém mais o canal cheio. Quanto mais tempo leva para perceber a perda de um pacote, mais grave se torna o problema.

Observe que, neste exemplo, o receptor poderia ter enviado uma *confirmação negativa* (NAK) para o quadro 6 assim que o quadro 7 chegasse. No entanto, isso é desnecessário, pois o mecanismo de timeout do remetente é suficiente para detectar essa situação, e o envio de NAKs adiciona complexidade adicional ao receptor. Além disso, como mencionamos, seria legítimo enviar confirmações adicionais do quadro 5 quando os quadros 7 e 8 chegassem; em alguns casos, um remetente pode usar ACKs duplicados como um indício de que um quadro foi perdido. Ambas as abordagens ajudam a melhorar o desempenho, permitindo a detecção antecipada de perdas de pacotes.

Outra variação desse esquema seria usar *confirmações seletivas* . Ou seja, o receptor poderia confirmar exatamente os quadros que recebeu, em vez de apenas o quadro com a numeração mais alta recebida em ordem. Assim, no exemplo acima, o receptor poderia confirmar o recebimento dos quadros 7 e 8. Fornecer mais informações ao remetente torna potencialmente mais fácil para o remetente manter o canal cheio, mas adiciona complexidade à implementação.

O tamanho da janela de envio é selecionado de acordo com a quantidade de quadros que queremos ter pendentes no link em um determinado momento; `SWS`é fácil de calcular para um determinado produto atraso × largura de banda. Por outro lado, o receptor pode definir `RWS`o que quiser. Duas configurações comuns são , que implica

que o receptor não armazenará em buffer nenhum quadro que chegue fora de ordem, e , que implica que o receptor pode armazenar em buffer qualquer um dos quadros que o remetente transmitir. Não faz sentido definir, pois é impossível que mais de 100 quadros cheguem fora de ordem.`RWS = 1RWS = SWSRWS > SWSSWS`

Números de sequência finitos e janela deslizante

Retornamos agora à simplificação que introduzimos no algoritmo — nossa suposição de que os números de sequência podem crescer infinitamente. Na prática, é claro, o número de sequência de um quadro é especificado em um campo de cabeçalho de tamanho finito. Por exemplo, um campo de 3 bits significa que existem oito números de sequência possíveis, de 0 a 7. Isso torna necessário reutilizar números de sequência ou, dito de outra forma, a recombinação de números de sequência. Isso introduz o problema de ser capaz de distinguir entre diferentes encarnações dos mesmos números de sequência, o que implica que o número de números de sequência possíveis deve ser maior do que o número de quadros pendentes permitidos. Por exemplo, o método stop-and-wait permitia um quadro pendente por vez e tinha dois números de sequência distintos.

Suponha que temos um número a mais em nosso espaço de números de sequência do que temos quadros potencialmente pendentes; isto é, , onde é o número de números de sequência disponíveis. Isso é suficiente? A resposta depende de . Se , então é suficiente. Se for igual a , então ter apenas um maior que o tamanho da janela de envio não é bom o suficiente. Para ver isso, considere a situação em que temos os oito números de sequência de 0 a 7, e . Suponha que o remetente transmita quadros de 0 a 6, eles são recebidos com sucesso, mas os ACKs são perdidos. O receptor agora está esperando quadros de 7, 0 a 5, mas o remetente atinge o tempo limite e envia quadros de 0 a 6. Infelizmente, o receptor está esperando a segunda encarnação dos quadros de 0 a 5, mas obtém a primeira encarnação desses quadros. Esta é exatamente a situação que queríamos evitar.`SWS <= MaxSeqNum - 1MaxSeqNumRWSRWS = 1MaxSeqNum >= SWS + 1RWSSWSMaxSeqNumSWS = RWS = 7`

Acontece que o tamanho da janela de envio não pode ser maior que a metade do número de números de sequência disponíveis quando , ou dito mais precisamente,`RWS = SWS`

```
SWS < (MaxSeqNum + 1)/ 2
```

Intuitivamente, o que isso quer dizer é que o protocolo de janela deslizante alterna entre as duas metades do espaço de números de sequência, assim como o stop-and-wait alterna entre os números de sequência 0 e 1. A única diferença é que ele desliza continuamente entre as duas metades em vez de alternar discretamente entre elas.

Observe que esta regra é específica para a situação em que . Deixamos como exercício determinar a regra mais geral que funciona para valores arbitrários de e . Observe também que a relação entre o tamanho da janela e o espaço do número de sequência depende de uma suposição tão óbvia que é fácil de ignorar, ou seja, que os quadros não são reordenados em trânsito. Isso não pode acontecer em um link ponto a ponto direto, pois não há como um quadro ultrapassar outro durante a transmissão. No entanto, veremos o algoritmo de janela deslizante usado em ambientes diferentes e precisaremos elaborar outra regra.`RWS = SWSRWSSWS`

Implementação de Janela Deslizante

As rotinas a seguir ilustram como podemos implementar os lados de envio e recebimento do algoritmo de janela deslizante. As rotinas são extraídas de um protocolo funcional denominado, apropriadamente, Protocolo de Janela Deslizante (SWP). Para não nos preocuparmos com os protocolos adjacentes no grafo de protocolos, denotamos o protocolo acima do SWP como o protocolo de alto nível (HLP) e o protocolo abaixo do SWP como o protocolo de nível de enlace (LLP).

Começamos definindo um par de estruturas de dados. Primeiro, o cabeçalho do quadro é muito simples: ele contém um número de sequência (`SeqNum`) e um número de

confirmação (`AckNum`). Ele também contém um `Flags`campo que indica se o quadro é um ACK ou se contém dados.

```
typedef uint8_t SwpSeqno;

typedef struct {
    SwpSeqno    SeqNum;    /* sequence number of this frame */
    SwpSeqno    AckNum;    /* ack of received frame */
    uint8_t      Flags;    /* up to 8 bits worth of flags */
} SwpHdr;
```

Em seguida, o estado do algoritmo de janela deslizante tem a seguinte estrutura. Para o lado de envio do protocolo, este estado inclui variáveis `LAR`e `LFS`, conforme descrito anteriormente nesta seção, bem como uma fila que contém quadros que foram transmitidos, mas ainda não confirmados (`sendQ`). O estado de envio também inclui um *semáforo de contagem* chamado `sendWindowNotFull`. Veremos como isso é usado abaixo, mas geralmente um semáforo é uma primitiva de sincronização que suporta operações `semWait`e `semSignal`. Cada invocação de `semSignal`incrementa o semáforo em 1, e cada invocação de `semWait`decrementa `s`em 1, com o processo de chamada bloqueado (suspenso) deve decrementar o semáforo fazer com que seu valor se torne menor que 0. Um processo que é bloqueado durante sua chamada para `semWait` será autorizado a retomar assim que `semSignal`operações suficientes tenham sido realizadas para elevar o valor do semáforo acima de 0.

Para o lado receptor do protocolo, o estado inclui a variável `NFE`. Este é o *próximo quadro esperado* , o quadro com um número de sequência um a mais que o último quadro recebido (LFR), descrito anteriormente nesta seção. Há também uma fila que armazena quadros que foram recebidos fora de ordem (`recvQ`). Por fim, embora não mostrado, os tamanhos das janelas deslizantes do remetente e do destinatário são definidos pelas constantes `SWS`e `RWS`, respectivamente.

```
typedef struct {
    /* sender side state: */
    SwpSeqno    LAR;        /* seqno of last ACK received */
    SwpSeqno    LFS;        /* last frame sent */
    Semaphore   sendWindowNotFull;
```

```
    SwpHdr      hdr;        /* pre-initialized header */
    struct sendQ_slot {
        Event   timeout;    /* event associated with send-timeout */
        Msg     msg;
    }   sendQ[SWS];

    /* receiver side state: */
    SwpSeqno    NFE;        /* seqno of next frame expected */
    struct recvQ_slot {
        int     received;   /* is msg valid? */
        Msg     msg;
    }   recvQ[RWS];
} SwpState;
```

O lado de envio do SWP é implementado pelo procedimento `sendSWP`. Esta rotina é bastante simples. Primeiro, `semWait`faz com que este processo seja bloqueado em um semáforo até que seja permitido enviar outro quadro. Uma vez autorizado a prosseguir, `sendSWP`define o número de sequência no cabeçalho do quadro, salva uma cópia do quadro na fila de transmissão (`sendQ`), agenda um evento de tempo limite para lidar com o caso em que o quadro não é confirmado e envia o quadro para o protocolo de nível imediatamente inferior, que denotamos como `LINK`.

Um detalhe que vale a pena notar é a chamada para `store_swp_hdr`imediatamente antes da chamada para `msgAddHdr`. Esta rotina traduz a estrutura C que contém o cabeçalho SWP (`state->hdr`) em uma sequência de bytes que pode ser anexada com segurança ao início da mensagem (`hbuf`). Esta rotina (não mostrada) deve traduzir cada campo inteiro no cabeçalho para a ordem de bytes da rede e remover qualquer preenchimento que o compilador tenha adicionado à estrutura C. A questão da ordem dos bytes não é trivial, mas por enquanto basta assumir que esta rotina coloca o bit mais significativo de um inteiro multipalavra no byte com o endereço mais alto.

Outra complexidade dessa rotina é o uso de `semWait` e o `sendWindowNotFull`semáforo. `sendWindowNotFull`é inicializado com o tamanho da janela deslizante do remetente `SWS`(esta inicialização não é mostrada). Cada vez que o remetente transmite um quadro, a `semWait`operação diminui essa contagem e bloqueia o remetente caso a contagem chegue a 0. Cada vez que um ACK é recebido, a `semSignal`operação invocada em

`deliverSWP`(veja abaixo) incrementa essa contagem, desbloqueando assim qualquer remetente em espera.

```
static int
sendSWP(SwpState *state, Msg *frame)
{
    struct sendQ_slot *slot;
    char hbuf[HLEN];

    /* wait for send window to open */
    semWait(&state->sendWindowNotFull);
    state->hdr.SeqNum = ++state->LFS;
    slot = &state->sendQ[state->hdr.SeqNum % SWS];
    store_swp_hdr(state->hdr, hbuf);
    msgAddHdr(frame, hbuf, HLEN);
    msgSaveCopy(&slot->msg, frame);
    slot->timeout = evSchedule(swpTimeout, slot, SWP_SEND_TIMEOUT);
    return send(LINK, frame);
}
```

Antes de prosseguir para o lado de recebimento do SWP, precisamos reconciliar uma aparente inconsistência. Por um lado, temos dito que um protocolo de alto nível invoca os serviços de um protocolo de baixo nível chamando a `send`operação, então esperaríamos que um protocolo que deseja enviar uma mensagem via SWP chamasse . Por outro lado, o procedimento que implementa a operação de envio do SWP é chamado , e seu primeiro argumento é uma variável de estado ( ). O que acontece? A resposta é que o sistema operacional fornece código de ligação que traduz a chamada genérica para em uma chamada específica do protocolo para . Esse código de ligação mapeia o primeiro argumento para (a variável mágica do protocolo) em um ponteiro de função para e um ponteiro para o estado do protocolo que o SWP precisa para fazer seu trabalho. A razão pela qual temos o protocolo de alto nível invocando indiretamente a função específica do protocolo por meio da chamada de função genérica é que queremos limitar a quantidade de informações que o protocolo de alto nível codificou nele sobre o protocolo de baixo nível. Isso torna mais fácil alterar a configuração do gráfico do protocolo em algum momento no futuro.`send(SWP, packet)``sendSWP``SwpState``send``sendSWP``sendSWP``sendSWP`

Agora, passamos para a implementação específica do protocolo SWP da `deliver`operação, apresentada no procedimento `deliverSWP`. Esta rotina, na verdade, lida com dois tipos diferentes de mensagens recebidas: ACKs para quadros enviados anteriormente deste nó e quadros de dados que chegam a este nó. De certa forma, a parte ACK desta rotina é a contrapartida da parte remetente do algoritmo apresentado em `sendSWP`. A decisão sobre se a mensagem recebida é um ACK ou um quadro de dados é tomada verificando o `Flags`campo no cabeçalho. Observe que esta implementação específica não suporta ACKs piggybacking em quadros de dados.

Quando o quadro recebido é um ACK, o comando `deliverSWP`simplesmente encontra o slot na fila de transmissão (`sendQ`) que corresponde ao ACK, cancela o evento de tempo limite e libera o quadro salvo naquele slot. Esse trabalho é, na verdade, feito em um loop, já que o ACK pode ser cumulativo. A única outra coisa a se notar neste caso é a chamada à subrotina `swpInWindow`. Esta subrotina, apresentada a seguir, garante que o número de sequência do quadro que está sendo confirmado esteja dentro do intervalo de ACKs que o remetente espera receber.

Quando o quadro de entrada contém dados, `deliverSWP`primeiro chama `msgStripHdr`e `load_swp_hdr`para extrair o cabeçalho do quadro. A rotina `load_swp_hdr`é a contrapartida da `store_swp_hdr` discutida anteriormente; ela traduz uma sequência de bytes para a estrutura de dados C que contém o cabeçalho SWP. `deliverSWP`Em seguida, chama `swpInWindow`para garantir que o número de sequência do quadro esteja dentro do intervalo de números de sequência esperado. Se estiver, a rotina percorre o conjunto de quadros consecutivos recebidos e os repassa para o protocolo de nível superior, invocando a `deliverHLP`rotina. Ela também envia um ACK cumulativo de volta ao remetente, mas o faz percorrendo a fila de recebimento (não utiliza a `SeqNumToAck`variável usada na descrição em prosa apresentada anteriormente nesta seção).

```
static int
deliverSWP(SwpState *state, Msg *frame)
{
    SwpHdr   hdr;
```

```
    char     *hbuf;

    hbuf = msgStripHdr(frame, HLEN);
    load_swp_hdr(&hdr, hbuf)
    if (hdr.Flags & FLAG_ACK_VALID)
    {
        /* received an acknowledgment—do SENDER side */
        if (swpInWindow(hdr.AckNum, state->LAR + 1, state->LFS))
        {
            do
            {
                struct sendQ_slot *slot;

                slot = &state->sendQ[++state->LAR % SWS];
                evCancel(slot->timeout);
                msgDestroy(&slot->msg);
                semSignal(&state->sendWindowNotFull);
            } while (state->LAR != hdr.AckNum);
        }
    }

    if (hdr.Flags & FLAG_HAS_DATA)
    {
        struct recvQ_slot *slot;

        /* received data packet—do RECEIVER side */
        slot = &state->recvQ[hdr.SeqNum % RWS];
        if (!swpInWindow(hdr.SeqNum, state->NFE, state->NFE + RWS - 1))
        {
            /* drop the message */
            return SUCCESS;
        }
        msgSaveCopy(&slot->msg, frame);
        slot->received = TRUE;
        if (hdr.SeqNum == state->NFE)
        {
            Msg m;

            while (slot->received)
            {
                deliver(HLP, &slot->msg);
                msgDestroy(&slot->msg);
                slot->received = FALSE;
                slot = &state->recvQ[++state->NFE % RWS];
            }
            /* send ACK: */
            prepare_ack(&m, state->NFE - 1);
            send(LINK, &m);
            msgDestroy(&m);
        }
    }
    return SUCCESS;
}
```

Por fim, `swpInWindow`é uma sub-rotina simples que verifica se um determinado número de sequência está entre algum número de sequência mínimo e máximo.

```
static bool
swpInWindow(SwpSeqno seqno, SwpSeqno min, SwpSeqno max)
{
    SwpSeqno pos, maxpos;

    pos    = seqno - min;       /* pos *should* be in range [0..MAX) */
    maxpos = max - min + 1;     /* maxpos is in range [0..MAX] */
    return pos < maxpos;
}
```

Ordem dos quadros e controle de fluxo

O protocolo de janela deslizante é talvez o algoritmo mais conhecido em redes de computadores. O que pode ser facilmente confundido sobre o algoritmo, no entanto, é que ele pode ser usado para desempenhar três funções diferentes. A primeira função é aquela em que nos concentramos nesta seção: entregar quadros de forma confiável através de um enlace não confiável. (Em geral, o algoritmo pode ser usado para entregar mensagens de forma confiável através de uma rede não confiável.) Esta é a função central do algoritmo.

A segunda função que o algoritmo de janela deslizante pode desempenhar é preservar a ordem em que os quadros são transmitidos. Isso é fácil de fazer no receptor — como cada quadro tem um número de sequência, o receptor apenas garante que não passará um quadro para o protocolo de nível imediatamente superior até que já tenha passado todos os quadros com um número de sequência menor. Ou seja, o receptor armazena em buffer (ou seja, não passa adiante) quadros fora de ordem. A versão do algoritmo de janela deslizante descrita nesta seção preserva a ordem dos quadros, embora possamos imaginar uma variação na qual o receptor passa os quadros para o próximo protocolo sem esperar que todos os quadros anteriores sejam entregues. Uma pergunta que devemos nos fazer é se realmente precisamos do protocolo de janela deslizante para manter os quadros em ordem no nível do enlace ou se, em vez disso, essa funcionalidade deve ser implementada por um protocolo mais alto na pilha.

A terceira função que o algoritmo de janela deslizante às vezes desempenha é oferecer suporte *ao controle de fluxo* — um mecanismo de feedback pelo qual o receptor é capaz de controlar o transmissor. Tal mecanismo é usado para impedir que o transmissor sobrecarregue o receptor — ou seja, transmita mais dados do que o receptor é capaz de processar. Isso geralmente é realizado por meio do aumento do protocolo de janela deslizante, de modo que o receptor não apenas reconheça os quadros recebidos, mas também informe ao transmissor quantos quadros ele tem espaço para receber. O número de quadros que o receptor é capaz de receber corresponde à quantidade de espaço livre no buffer. Como no caso da entrega ordenada, precisamos garantir que o controle de fluxo seja necessário no nível do enlace antes de incorporá-lo ao protocolo de janela deslizante.

Conclusão principal

Um conceito importante a ser retirado desta discussão é o princípio de design de sistemas que chamamos de *separação de responsabilidades* . Ou seja, você deve ter o cuidado de distinguir entre diferentes funções que às vezes são reunidas em um único mecanismo, e deve garantir que cada função seja necessária e esteja sendo suportada da maneira mais eficaz. Neste caso específico, entrega confiável, entrega ordenada e controle de fluxo às vezes são combinados em um único protocolo de janela deslizante, e devemos nos perguntar se isso é a coisa certa a fazer no nível do link. [Próximo]

2.5.3 Canais Lógicos Concorrentes

O protocolo de enlace de dados usado na ARPANET original oferece uma alternativa interessante ao protocolo de janela deslizante, pois consegue manter o canal cheio enquanto ainda utiliza o algoritmo simples de parar e esperar. Uma consequência importante dessa abordagem é que os quadros enviados por um determinado enlace não são mantidos em nenhuma ordem específica. O protocolo também não implica nada sobre controle de fluxo.

A ideia subjacente ao protocolo ARPANET, que chamamos de *canais lógicos concorrentes* , é multiplexar vários canais lógicos em um único enlace ponto a ponto e executar o algoritmo stop-and-wait em cada um desses canais lógicos. Não há relação entre os quadros enviados em nenhum dos canais lógicos, mas, como um quadro diferente pode estar pendente em cada um dos vários canais lógicos, o remetente consegue manter o enlace cheio.

Mais precisamente, o remetente mantém 3 bits de estado para cada canal: um booleano, que indica se o canal está ocupado; o número de sequência de 1 bit a ser usado na próxima vez que um quadro for enviado neste canal lógico; e o próximo número de sequência a ser esperado em um quadro que chega neste canal. Quando o nó tem um quadro para enviar, ele usa o canal com menor nível de inatividade; caso contrário, se comporta como um "parar e esperar".

Na prática, a ARPANET suportava 8 canais lógicos em cada enlace terrestre e 16 em cada enlace de satélite. No caso do enlace terrestre, o cabeçalho de cada quadro incluía um número de canal de 3 bits e um número de sequência de 1 bit, totalizando 4 bits. Este é exatamente o número de bits que o protocolo de janela deslizante requer para suportar até 8 quadros pendentes no enlace quando .`RWS = SWS`

2.6 Redes de Multiacesso

Desenvolvida em meados da década de 1970 por pesquisadores do Xerox Palo Alto Research Center (PARC), a Ethernet acabou se tornando a tecnologia dominante em redes locais, emergindo de um conjunto de tecnologias concorrentes. Hoje, ela compete principalmente com as redes sem fio 802.11, mas continua extremamente popular em redes de campus e data centers. O nome mais geral para a tecnologia por trás da Ethernet é Carrier Sense, Multiple Access with Collision Detect (CSMA/CD).

Como indicado pelo nome CSMA, a Ethernet é uma rede de acesso múltiplo, o que significa que um conjunto de nós envia e recebe quadros por meio de um link compartilhado. Portanto, podemos pensar em uma Ethernet como um barramento com várias estações conectadas a ele. O "sensor de portadora" em CSMA/CD significa que todos os nós conseguem distinguir entre um link ocioso e um ocupado, e a "detecção de colisão" significa que um nó escuta enquanto transmite e, portanto, pode detectar quando um quadro que está transmitindo interferiu (colidiu) com um quadro transmitido por outro nó.

A Ethernet tem suas raízes em uma antiga rede de rádio por pacotes, chamada Aloha, desenvolvida na Universidade do Havaí para suportar a comunicação entre computadores nas Ilhas Havaianas. Assim como a rede Aloha, o problema fundamental enfrentado pela Ethernet é como mediar o acesso a um meio compartilhado de forma justa e eficiente (na Aloha, o meio era a atmosfera, enquanto na Ethernet o meio era originalmente um cabo coaxial). A ideia central tanto da Aloha quanto da Ethernet é um algoritmo que controla quando cada nó pode transmitir.

Os enlaces Ethernet modernos são agora, em grande parte, ponto a ponto; ou seja, conectam um host a um *switch* Ethernet ou interconectam switches. Como consequência, o algoritmo de "acesso múltiplo" não é muito utilizado nas redes Ethernet cabeadas atuais, mas uma variante agora é utilizada em redes sem fio, como as redes 802.11 (também conhecidas como Wi-Fi). Devido à enorme influência da Ethernet, optamos por descrever seu algoritmo clássico aqui e, em seguida, explicar como ele foi adaptado ao Wi-Fi na próxima seção. Também discutiremos switches Ethernet em outro lugar. Por enquanto, vamos nos concentrar em como um único enlace Ethernet funciona.

A Digital Equipment Corporation e a Intel Corporation se uniram à Xerox para definir um padrão Ethernet de 10 Mbps em 1978. Esse padrão formou a base para o padrão IEEE

802.3, que também define uma coleção muito mais ampla de mídias físicas sobre as quais uma Ethernet pode operar, incluindo versões de 100 Mbps, 1 Gbps, 10 Gbps, 40 Gbps e 100 Gbps.

2.6.1 Propriedades Físicas

Os segmentos Ethernet foram implementados originalmente usando cabos coaxiais de até 500 m de comprimento. (Os Ethernets modernos usam pares de cobre trançados, geralmente um tipo específico conhecido como "Categoria 5" ou fibras ópticas, e em alguns casos podem ser bem mais longos que 500 m.) Esse cabo era semelhante ao tipo usado para TV a cabo. Os hosts se conectavam a um segmento Ethernet por meio de tapping. Um *transceptor* , um pequeno dispositivo conectado diretamente ao tap, detectava quando a linha estava ociosa e conduzia o sinal quando o host estava transmitindo. Ele também recebia sinais de entrada. O transceptor, por sua vez, se conectava a um adaptador Ethernet, que era conectado ao host. Essa configuração é mostrada na Figura 39 .

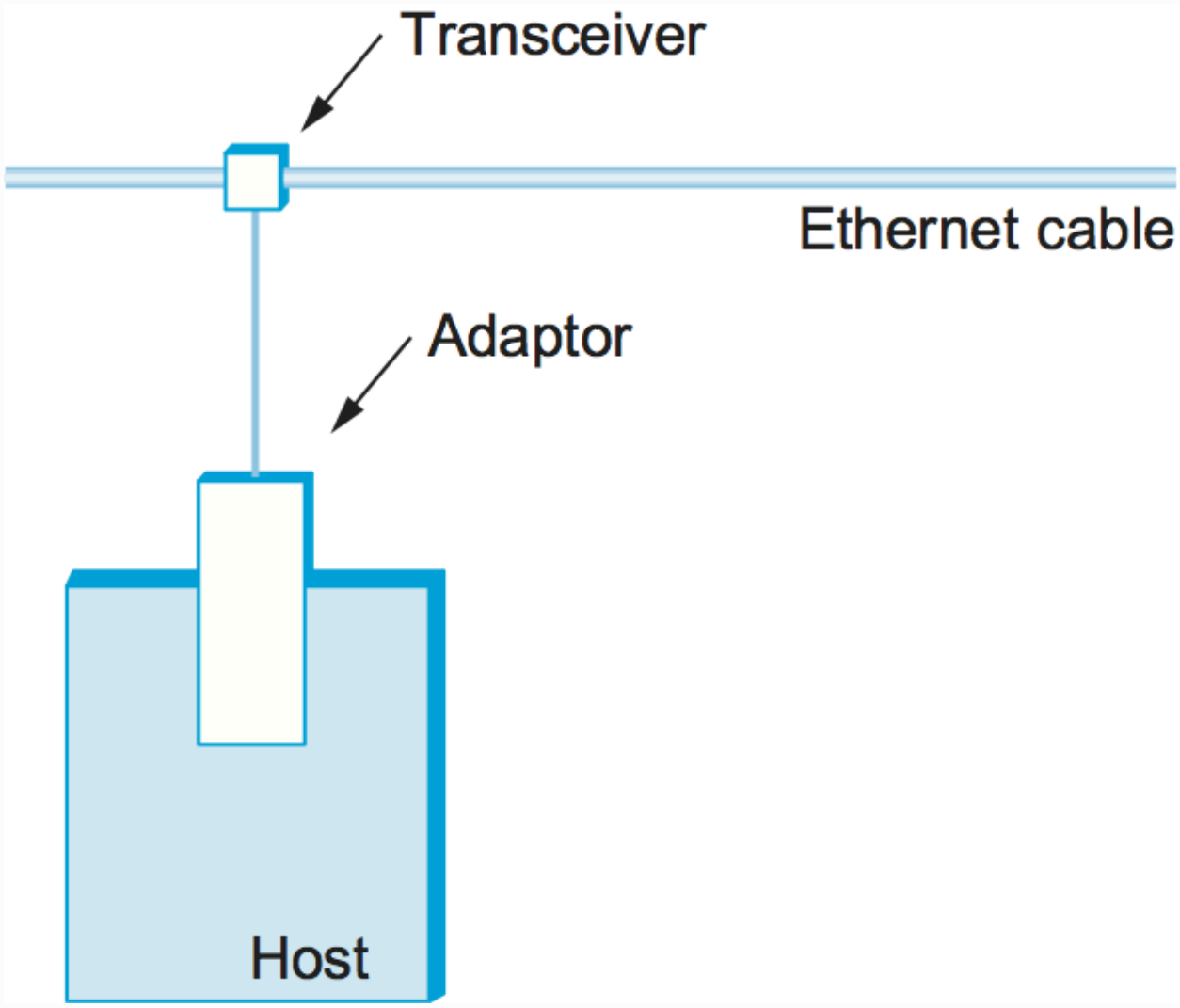

Figura 39. *Transceptor e adaptador Ethernet.*

Vários segmentos Ethernet podem ser unidos por *repetidores* (ou uma variante multiporta de um repetidor, chamada de *hub*). Um repetidor é um dispositivo que encaminha sinais digitais, assim como um amplificador encaminha sinais analógicos; repetidores não compreendem bits ou quadros. Não mais do que quatro repetidores podiam ser posicionados entre qualquer par de hosts, o que significa que uma Ethernet clássica tinha um alcance total de apenas 2.500 m. Por exemplo, usar apenas dois repetidores entre qualquer par de hosts suporta uma configuração semelhante à ilustrada na Figura 40 ; ou seja, um segmento que percorre a espinha dorsal de um edifício com um segmento em cada andar.

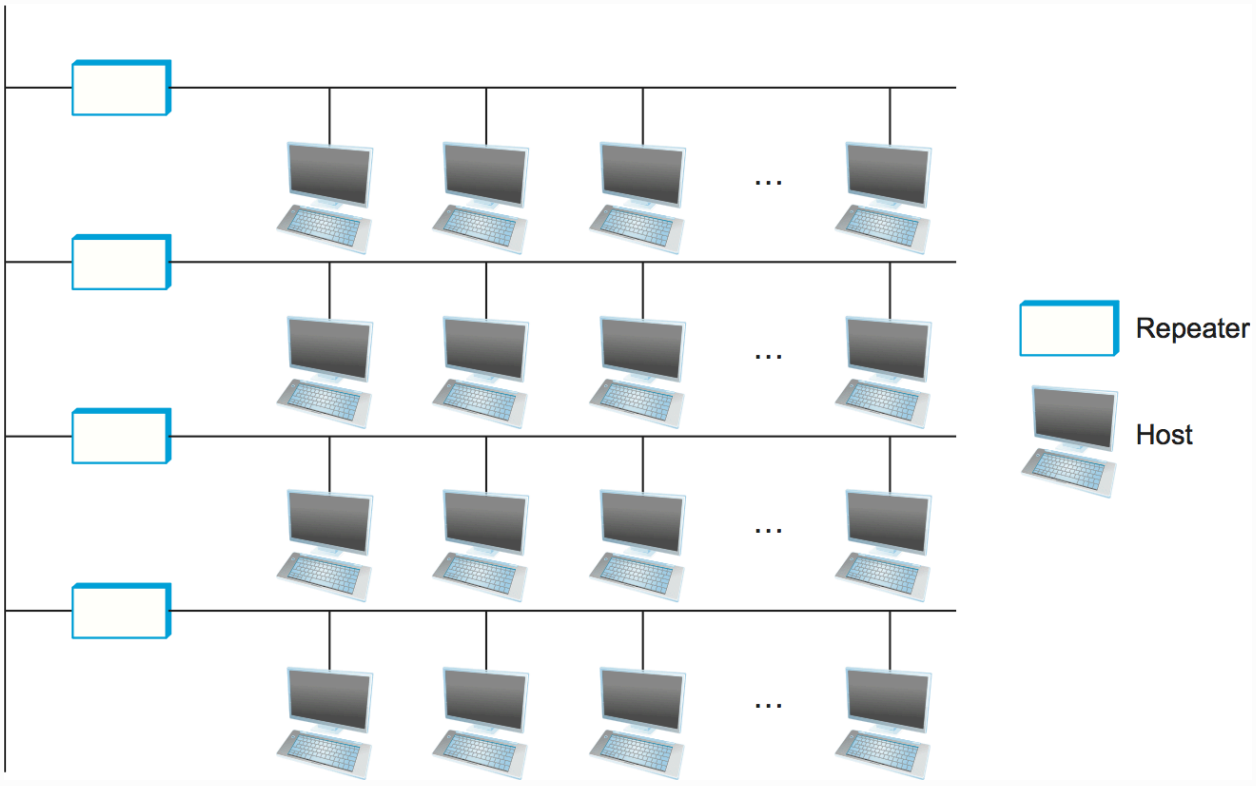

Figura 40. *Repetidor Ethernet, interconectando segmentos para formar um domínio de colisão maior.*

Qualquer sinal enviado por um host à Ethernet é transmitido por toda a rede; ou seja, o sinal é propagado em ambas as direções, e repetidores e hubs o encaminham em todos os segmentos de saída. Terminadores conectados à extremidade de cada segmento absorvem o sinal e impedem que ele retorne e interfira nos sinais de saída. As especificações Ethernet originais usavam o esquema de codificação Manchester descrito em uma seção anterior, enquanto a codificação 4B/5B (ou o esquema similar 8B/10B) é usada hoje em redes Ethernet de alta velocidade.

É importante entender que, independentemente de uma determinada Ethernet abranger um único segmento, uma sequência linear de segmentos conectados por repetidores ou múltiplos segmentos conectados em uma configuração em estrela, os dados transmitidos por qualquer host nessa Ethernet alcançam todos os outros hosts. Esta é a boa notícia. A má notícia é que todos esses hosts estão competindo pelo acesso ao mesmo link e, como consequência, dizemos que estão no mesmo *domínio*

de colisão . A parte de multiacesso da Ethernet trata de lidar com a competição pelo link que surge em um domínio de colisão.

2.6.2 Protocolo de acesso

Agora, voltaremos nossa atenção para o algoritmo que controla o acesso a um link Ethernet compartilhado. Esse algoritmo é comumente chamado de *controle de acesso à mídia* (MAC) da Ethernet. Ele normalmente é implementado em hardware no adaptador de rede. Não descreveremos o hardware *em si* , mas sim o algoritmo que ele implementa. Primeiro, porém, descreveremos o formato e os endereços dos quadros Ethernet.

Formato do quadro

Cada quadro Ethernet é definido pelo formato apresentado na Figura 41. O preâmbulo de 64 bits permite que o receptor sincronize com o sinal; é uma sequência de 0s e 1s alternados. Tanto o host de origem quanto o de destino são identificados com um endereço de 48 bits. O campo de tipo de pacote serve como chave de demultiplexação; ele identifica a qual dos muitos protocolos de nível superior este quadro deve ser entregue. Cada quadro contém até 1500 bytes de dados. No mínimo, um quadro deve conter pelo menos 46 bytes de dados, mesmo que isso signifique que o host tenha que preencher o quadro antes de transmiti-lo. A razão para esse tamanho mínimo de quadro é que o quadro deve ser longo o suficiente para detectar uma colisão; discutiremos isso mais adiante. Finalmente, cada quadro inclui um CRC de 32 bits. Assim como o protocolo HDLC descrito em uma seção anterior, o Ethernet é um protocolo de enquadramento orientado a bits. Observe que, da perspectiva do host, um quadro Ethernet tem um cabeçalho de 14 bytes: dois endereços de 6 bytes e um campo de tipo de 2 bytes. O adaptador de envio anexa o preâmbulo e o CRC antes da transmissão, e o adaptador de recebimento os remove.

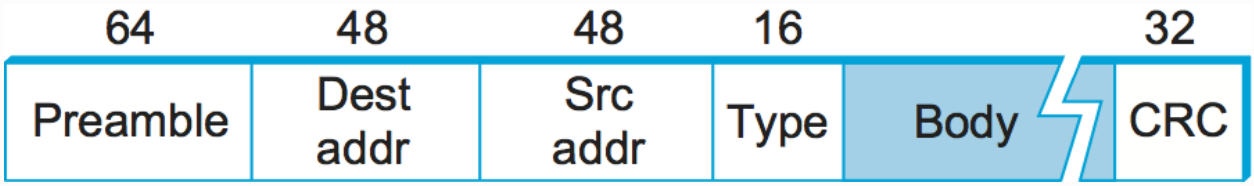

Figura 41. *Formato do quadro Ethernet.*

Endereços

Cada host em uma Ethernet — na verdade, cada host Ethernet no mundo — possui um endereço Ethernet exclusivo. Tecnicamente, o endereço pertence ao adaptador, não ao host; geralmente, ele é gravado na memória ROM. Os endereços Ethernet são normalmente impressos em um formato legível por humanos, como uma sequência de seis números separados por dois pontos. Cada número corresponde a 1 byte do endereço de 6 bytes e é representado por um par de dígitos hexadecimais, um para cada um dos nibbles de 4 bits do byte; os 0s iniciais são descartados. Por exemplo, `8:0:2b:e4:b1:2`é a representação legível por humanos do endereço Ethernet?

```
00001000  00000000  00101011  11100100  10110001  00000010
```

Para garantir que cada adaptador receba um endereço único, cada fabricante de dispositivos Ethernet recebe um prefixo diferente, que deve ser anexado ao endereço de cada adaptador que fabrica. Por exemplo, a Advanced Micro Devices recebeu o prefixo de 24 bits `080020`(ou `8:0:20`). Um determinado fabricante, então, garante que os sufixos de endereço que produz sejam únicos.

Cada quadro transmitido em uma Ethernet é recebido por todos os adaptadores conectados a essa Ethernet. Cada adaptador reconhece os quadros endereçados ao seu endereço e repassa apenas esses quadros ao host. (Um adaptador também pode ser programado para operar em modo *promíscuo* , no qual entrega todos os quadros recebidos ao host, mas este não é o modo normal.) Além desses endereços *unicast* , um endereço Ethernet composto apenas por 1s é tratado como um endereço *de broadcast* ; todos os adaptadores repassam quadros endereçados ao endereço de broadcast para o host. Da mesma forma, um endereço que tem o primeiro bit definido

como 1, mas não é o endereço de broadcast, é chamado de endereço *multicast* . Um determinado host pode programar seu adaptador para aceitar um conjunto de endereços multicast. Endereços multicast são usados para enviar mensagens a um subconjunto de hosts em uma Ethernet (por exemplo, todos os servidores de arquivos). Resumindo, um adaptador Ethernet recebe todos os quadros e aceita

- Quadros endereçados ao seu próprio endereço
- Quadros endereçados ao endereço de transmissão
- Quadros endereçados a um endereço multicast, se ele tiver sido instruído a escutar esse endereço
- Todos os quadros, se tiverem sido colocados em modo promíscuo

Ele passa para o host apenas os quadros que ele aceita.

Algoritmo do Transmissor

Como acabamos de ver, o lado receptor do protocolo Ethernet é simples; a inteligência real é implementada no lado remetente. O algoritmo do transmissor é definido da seguinte forma.

Quando o adaptador tem um quadro para enviar e a linha está ociosa, ele transmite o quadro imediatamente; não há negociação com os outros adaptadores. O limite superior de 1500 bytes na mensagem significa que o adaptador pode ocupar a linha apenas por um período de tempo fixo.

Quando um adaptador tem um quadro para enviar e a linha está ocupada, ele aguarda que a linha fique ociosa e então transmite imediatamente. (Para ser mais preciso, todos os adaptadores aguardam 9,6 μs após o término de um quadro antes de começar a transmitir o próximo. Isso vale tanto para o remetente do primeiro quadro quanto para os nós que aguardam a linha ficar ociosa.) Diz-se que a Ethernet é um protocolo *1-persistente* porque um adaptador com um quadro para enviar transmite com

probabilidade 1 sempre que uma linha ocupada fica ociosa. Em geral, um algoritmo *p-persistente* transmite com probabilidade

$$0 \leq p \leq 1$$

depois que uma linha se torna ociosa e adia com probabilidade $q = 1 - p$. O raciocínio por trás da escolha de um $p < 1$ é que pode haver vários adaptadores esperando que a linha ocupada fique ociosa, e não queremos que todos eles comecem a transmitir ao mesmo tempo. Se cada adaptador transmite imediatamente com uma probabilidade de, digamos, 33%, então até três adaptadores podem estar esperando para transmitir e as chances são de que apenas um começará a transmitir quando a linha ficar ociosa. Apesar desse raciocínio, um adaptador Ethernet sempre transmite imediatamente após perceber que a rede ficou ociosa e tem sido muito eficaz em fazê-lo.

Para completar a história sobre protocolos *p -persistentes para o caso em que* $p<1$, você pode se perguntar quanto tempo um remetente que perde o cara ou coroa (ou seja, decide adiar) tem que esperar antes de poder transmitir. A resposta para a rede Aloha, que originalmente desenvolveu esse estilo de protocolo, foi dividir o tempo em slots discretos, com cada slot correspondendo ao tempo que leva para transmitir um quadro completo. Sempre que um nó tem um quadro para enviar e detecta um slot vazio (ocioso), ele transmite com probabilidade p e adia até o próximo slot com probabilidade $q = 1 - p$. Se o próximo slot também estiver vazio, o nó novamente decide transmitir ou adiar, com probabilidades p e q , respectivamente. Se o próximo slot não estiver vazio — ou seja, alguma outra estação decidiu transmitir — então o nó simplesmente espera pelo próximo slot ocioso e o algoritmo se repete.

Voltando à nossa discussão sobre a Ethernet, como não há controle centralizado, é possível que dois (ou mais) adaptadores comecem a transmitir ao mesmo tempo, seja porque ambos encontraram a linha ociosa ou porque ambos estavam esperando que uma linha ocupada se tornasse ociosa. Quando isso acontece, diz-se que os dois (ou mais) quadros colidem *na* rede. Cada remetente, como a Ethernet suporta detecção de colisão, é capaz de determinar que uma colisão está em andamento. No momento em

que um adaptador detecta que seu quadro está colidindo com outro, ele primeiro se certifica de transmitir uma sequência de interferência de 32 bits e, em seguida, interrompe a transmissão. Assim, um transmissor enviará no mínimo 96 bits em caso de colisão: preâmbulo de 64 bits mais sequência de interferência de 32 bits.

Uma maneira de um adaptador enviar apenas 96 bits — o que às vezes é chamado de *quadro runt* — é se os dois hosts estiverem próximos um do outro. Se os dois hosts estivessem mais distantes, teriam que transmitir por mais tempo e, portanto, enviar mais bits antes de detectar a colisão. Na verdade, o pior cenário acontece quando os dois hosts estão em extremidades opostas da Ethernet. Para ter certeza de que o quadro que acabou de enviar não colidiu com outro quadro, o transmissor pode precisar enviar até 512 bits. Não por coincidência, cada quadro Ethernet deve ter pelo menos 512 bits (64 bytes) de comprimento: 14 bytes de cabeçalho mais 46 bytes de dados mais 4 bytes de CRC.

Por que 512 bits? A resposta está relacionada a outra pergunta que você pode fazer sobre uma Ethernet: por que seu comprimento é limitado a apenas 2.500 m? Por que não 10 ou 1.000 km? A resposta para ambas as perguntas tem a ver com o fato de que quanto mais distantes dois nós estiverem, mais tempo levará para um quadro enviado por um chegar ao outro, e a rede fica vulnerável a colisões durante esse período.

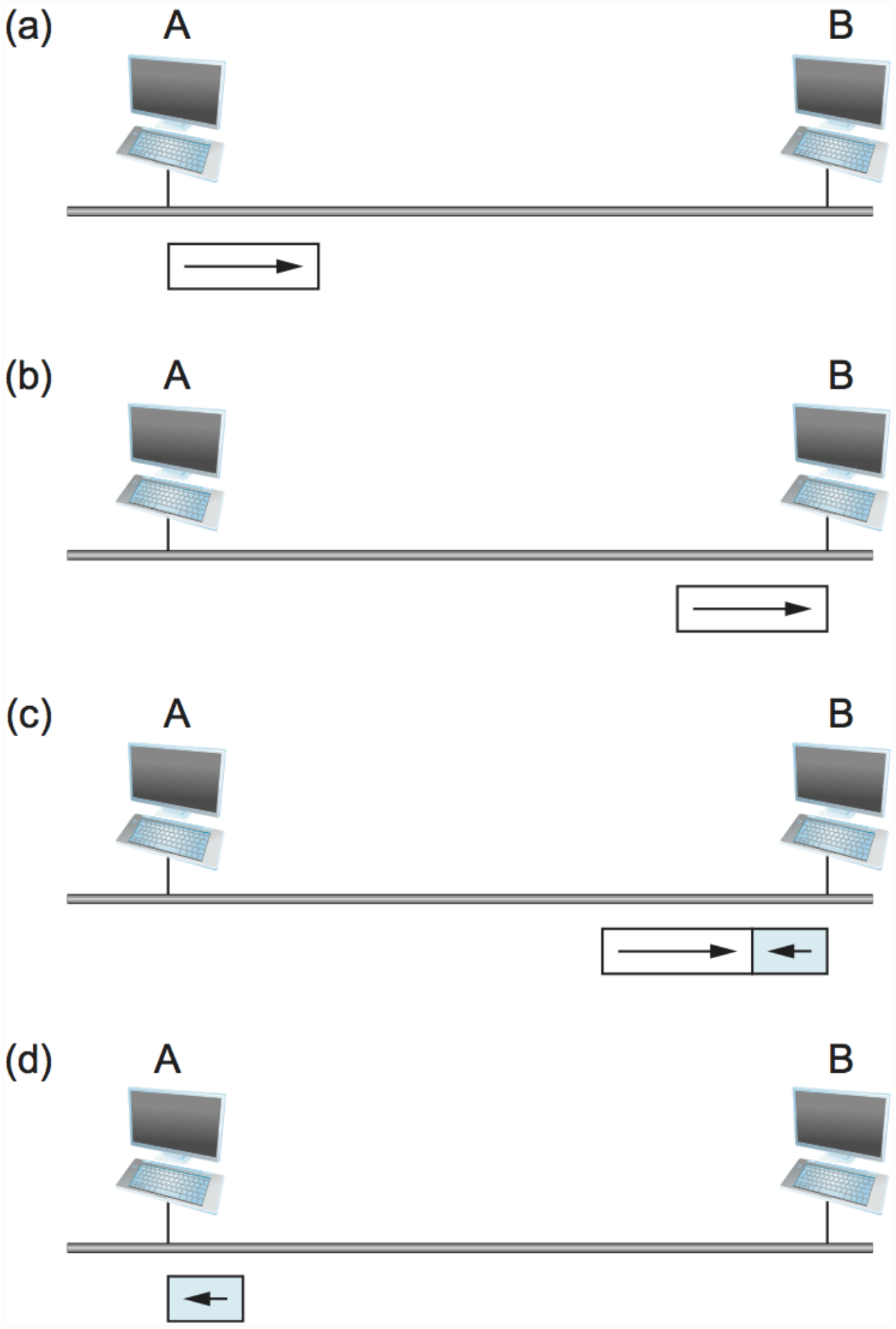
(a)
A
B
(b)
A
B
(c)
A
B
(d)
A
B

Figura 42. *Pior cenário: (a) A envia um quadro no instante t; (b) o quadro de A chega a B no instante t+d; (c) B começa a transmitir no instante t+d e colide com o quadro de A; (d) o quadro runt (32 bits) de B chega a A no instante t+2×d.*

A Figura 42 ilustra o pior cenário, onde os hosts A e B estão em extremidades opostas da rede. Suponha que o host A comece a transmitir um quadro no instante t, como mostrado em (a). Ele leva uma latência de enlace (vamos denotar a latência como d) para que o quadro chegue ao host B. Assim, o primeiro bit do quadro de A chega a B no instante t+d, como mostrado em (b). Suponha que um instante antes da chegada do quadro do host A (ou seja, B ainda vê uma linha livre), o host B começa a transmitir seu próprio quadro. O quadro de B colidirá imediatamente com o quadro de A, e essa colisão será detectada pelo host B (c). O host B enviará a sequência de interferência de 32 bits, conforme descrito acima. (O quadro de B será um runt.) Infelizmente, o host A não saberá que a colisão ocorreu até que o quadro de B o alcance, o que acontecerá uma latência de enlace depois, no instante $t+2\times d$, como mostrado em (d). O host A deve continuar transmitindo até esse momento para detectar a colisão. Em outras palavras, o host A deve transmitir por $2\times d$ para garantir que detecte todas as colisões possíveis. Considerando que uma Ethernet configurada ao máximo tem 2500 m de comprimento e que pode haver até quatro repetidores entre quaisquer dois hosts, o atraso de ida e volta foi determinado em 51,2 µs, o que em uma Ethernet de 10 Mbps corresponde a 512 bits. A outra maneira de analisar essa situação é que precisamos limitar a latência máxima da Ethernet a um valor relativamente pequeno (por exemplo, 51,2 µs) para que o algoritmo de acesso funcione; portanto, o comprimento máximo de uma Ethernet deve ser algo em torno de 2500 m.

Uma vez que um adaptador detecta uma colisão e interrompe sua transmissão, ele espera um certo tempo e tenta novamente. Cada vez que ele tenta transmitir, mas falha, o adaptador dobra o tempo que ele espera antes de tentar novamente. Essa estratégia de dobrar o intervalo de atraso entre cada tentativa de retransmissão é uma técnica geral conhecida como *backoff exponencial* . Mais precisamente, o adaptador primeiro atrasa 0 ou 51,2 µs, selecionados aleatoriamente. Se essa tentativa falhar, ele espera 0, 51,2, 102,4 ou 153,6 µs (selecionados aleatoriamente) antes de tentar

novamente; isso é k × 51,2 para k = 0..3. Após a terceira colisão, ele espera *k × 51,2* para k = 0..2 ^{3} - 1, novamente selecionado aleatoriamente. Em geral, o algoritmo seleciona aleatoriamente um *k* entre 0 e 2 ^{n} - 1 e aguarda k × 51,2 µs, onde *n* é o número de colisões ocorridas até o momento. O adaptador desiste após um determinado número de tentativas e relata um erro de transmissão ao host. Os adaptadores normalmente tentam novamente até 16 vezes, embora o algoritmo de backoff limite *n* na fórmula acima a 10.

2.6.3 Longevidade da Ethernet

A Ethernet tem sido a tecnologia dominante em redes locais por mais de 30 anos. Hoje, ela é tipicamente implantada ponto a ponto, em vez de se conectar a um cabo coaxial, frequentemente opera a velocidades de 1 ou 10 Gbps em vez de 10 Mbps e permite pacotes jumbo com até 9.000 bytes de dados em vez de 1.500 bytes. No entanto, ela permanece compatível com o padrão original. Vale a pena dizer algumas palavras sobre o sucesso da Ethernet, para que possamos entender as propriedades que devemos emular com qualquer tecnologia que tente substituí-la.

Primeiro, uma Ethernet é extremamente fácil de administrar e manter: não há tabelas de roteamento ou configuração para manter atualizadas, e é fácil adicionar um novo host à rede. É difícil imaginar uma rede mais simples de administrar. Segundo, é barata: cabo/fibra é relativamente barato, e o único outro custo é o adaptador de rede em cada host. A Ethernet tornou-se profundamente enraizada por essas razões, e qualquer abordagem baseada em switches que aspirasse a substituí-la exigia investimento adicional em infraestrutura (os switches), além do custo de cada adaptador. A variante baseada em switches da Ethernet eventualmente conseguiu substituir a Ethernet multiacesso, mas isso se deve principalmente ao fato de poder ser *implantada incrementalmente* — com alguns hosts conectados por links ponto a ponto a switches, enquanto outros permaneciam conectados por cabo coaxial e a repetidores ou hubs — mantendo a simplicidade da administração da rede.

2.7 Redes sem fio

As tecnologias sem fio diferem das conexões com fio em alguns aspectos importantes, embora compartilhem muitas propriedades comuns. Assim como nas conexões com fio, problemas de erros de bits são uma grande preocupação — geralmente ainda mais devido ao ambiente de ruído imprevisível da maioria das conexões sem fio. O enquadramento e a confiabilidade também precisam ser considerados. Ao contrário das conexões com fio, a energia é um grande problema para as conexões sem fio, especialmente porque elas são frequentemente usadas por pequenos dispositivos móveis (como telefones e sensores) que têm acesso limitado à energia (por exemplo, uma bateria pequena). Além disso, não se pode usar um transmissor de rádio com potência arbitrariamente alta — há preocupações com a interferência com outros dispositivos e, geralmente, regulamentações sobre quanta energia um dispositivo pode emitir em qualquer frequência.

Mídias sem fio também são inerentemente multiacesso; é difícil direcionar sua transmissão de rádio para apenas um receptor ou evitar receber sinais de rádio de qualquer transmissor com potência suficiente na sua vizinhança. Portanto, o controle de acesso à mídia é uma questão central para links sem fio. E, como é difícil controlar quem recebe seu sinal quando você transmite pelo ar, problemas de espionagem também podem precisar ser resolvidos.

Existe uma variedade desconcertante de diferentes tecnologias sem fio, cada uma das quais apresenta diferentes compensações em diversas dimensões. Uma maneira simples de categorizar as diferentes tecnologias é pelas taxas de dados que fornecem e pela distância entre os nós de comunicação. Outras diferenças importantes incluem a parte do espectro eletromagnético que utilizam (incluindo a necessidade de licença) e a quantidade de energia que consomem. Nesta seção, discutimos duas tecnologias sem fio proeminentes: Wi-Fi (mais formalmente conhecido como 802.11) e Bluetooth. A

próxima seção discute redes celulares no contexto dos serviços de acesso de ISP. A Tabela 4 apresenta uma visão geral dessas tecnologias e como elas se comparam.

	Bluetooth (802.15.1)	**Wi-Fi (802.11)**	**4G Celular**
Comprimento típico do link	10 metros	100 metros	Dezenas de quilômetros
Taxa de dados típica	2 Mbps (compartilhado)	150-450 Mbps	1-5 Mbps
Uso típico	Conectar um periférico a um computador	Conecte um computador a uma base com fio	Conecte o telefone celular a uma torre com fio
Analogia da tecnologia com fio	USB	Ethernet	PON

Você deve se lembrar que largura de banda às vezes significa a largura de uma banda de frequência em hertz e às vezes a taxa de dados de um link. Como ambos os conceitos aparecem em discussões sobre redes sem fio, usaremos largura *de banda* aqui em seu sentido mais estrito — largura de uma banda de frequência — e usaremos o termo *taxa de dados* para descrever o número de bits por segundo que podem ser enviados pelo link, como na Tabela 4 .

2.7.1 Questões básicas

Como todos os links sem fio compartilham o mesmo meio, o desafio é compartilhá-lo de forma eficiente, sem interferir indevidamente entre si. A maior parte desse compartilhamento é realizada dividindo-o ao longo das dimensões de frequência e espaço. O uso exclusivo de uma frequência específica em uma área geográfica específica pode ser alocado a uma entidade individual, como uma empresa. É possível limitar a área coberta por um sinal eletromagnético porque tais sinais enfraquecem, ou *atenuam* , com a distância de sua origem. Para reduzir a área coberta pelo seu sinal, reduza a potência do seu transmissor.

Essas alocações são normalmente determinadas por agências governamentais, como a Comissão Federal de Comunicações (FCC) nos Estados Unidos. Bandas específicas (faixas de frequência) são alocadas para determinados usos. Algumas bandas são reservadas para uso governamental. Outras bandas são reservadas para usos como rádio AM, rádio FM, televisão, comunicação via satélite e telefones celulares. Frequências específicas dentro dessas bandas são então licenciadas para organizações individuais para uso em determinadas áreas geográficas. Por fim, várias bandas de frequência são reservadas para uso isento de licença — bandas nas quais a licença não é necessária.

Dispositivos que utilizam frequências isentas de licença ainda estão sujeitos a certas restrições para que esse compartilhamento, que de outra forma seria irrestrito, funcione. A mais importante delas é o limite na potência de transmissão. Isso limita o alcance de um sinal, tornando-o menos propenso a interferir em outro sinal. Por exemplo, um telefone sem fio (um dispositivo comum sem licença) pode ter um alcance de cerca de 30 metros.

Uma ideia que aparece muito quando o espectro é compartilhado entre muitos dispositivos e aplicações é *o espectro espalhado* . A ideia por trás do espectro espalhado é espalhar o sinal por uma banda de frequência mais ampla, de modo a minimizar o impacto da interferência de outros dispositivos. (O espectro espalhado foi

originalmente projetado para uso militar, então esses "outros dispositivos" frequentemente tentavam bloquear o sinal.) Por exemplo, *o salto de frequência* é uma técnica de espectro espalhado que envolve a transmissão do sinal por uma sequência aleatória de frequências; isto é, primeiro transmitindo em uma frequência, depois uma segunda, depois uma terceira e assim por diante. A sequência de frequências não é verdadeiramente aleatória, mas é computada algoritmicamente por um gerador de números pseudoaleatórios. O receptor usa o mesmo algoritmo que o remetente e o inicializa com a mesma semente; portanto, ele é capaz de pular frequências em sincronia com o transmissor para receber corretamente o quadro. Este esquema reduz a interferência, tornando improvável que dois sinais estejam usando a mesma frequência por mais do que o bit isolado infrequente.

Uma segunda técnica de espectro espalhado, chamada *sequência direta* , adiciona redundância para maior tolerância à interferência. Cada bit de dados é representado por vários bits no sinal transmitido, de modo que, se alguns dos bits transmitidos forem danificados por interferência, geralmente há redundância suficiente para recuperar o bit original. Para cada bit que o remetente deseja transmitir, ele na verdade envia o OU exclusivo daquele bit e n bits aleatórios. Assim como no salto de frequência, a sequência de bits aleatórios é gerada por um gerador de números pseudoaleatórios conhecido tanto pelo remetente quanto pelo receptor. Os valores transmitidos, conhecidos como *código de chipping* de n bits , espalham o sinal por uma banda de frequência que é n vezes mais ampla do que o quadro exigiria de outra forma. A Figura 43 dá um exemplo de uma sequência de chipping de 4 bits.

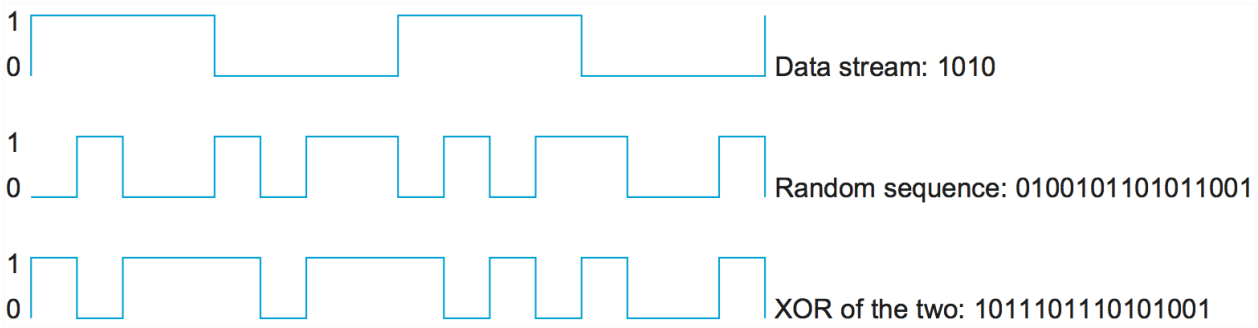

Figura 43. *Exemplo de sequência de chipping de 4 bits.*

Diferentes partes do espectro eletromagnético têm propriedades diferentes, tornando algumas mais adequadas à comunicação e outras menos. Por exemplo, algumas podem penetrar edifícios e outras não. Os governos regulam apenas a parte principal da comunicação: as faixas de rádio e micro-ondas. À medida que a demanda por espectro principal aumenta, há grande interesse no espectro que está se tornando disponível com a eliminação gradual da televisão analógica em favor da digital.

Em muitas redes sem fio atuais, observamos que existem duas classes diferentes de pontos de extremidade. Um ponto de extremidade, às vezes descrito como *estação base* , geralmente não tem mobilidade, mas possui uma conexão com fio (ou pelo menos de alta largura de banda) com a internet ou outras redes, como mostrado na Figura 44. O nó na outra extremidade do link — mostrado aqui como um nó cliente — geralmente é móvel e depende de seu link com a estação base para toda a sua comunicação com outros nós.

Observe que na Figura 44 utilizamos um par de linhas onduladas para representar a abstração de "enlace" sem fio fornecida entre dois dispositivos (por exemplo, entre uma estação base e um de seus nós clientes). Um dos aspectos interessantes da comunicação sem fio é que ela naturalmente suporta comunicação ponto-a-multiponto, pois as ondas de rádio enviadas por um dispositivo podem ser recebidas simultaneamente por vários dispositivos. No entanto, muitas vezes é útil criar uma abstração de enlace ponto-a-ponto para protocolos de camadas superiores, e veremos exemplos de como isso funciona mais adiante nesta seção.

Observe que, na Figura 44, a comunicação entre nós não-base (clientes) é roteada pela estação base. Isso ocorre apesar do fato de que as ondas de rádio emitidas por um nó cliente podem ser recebidas por outros nós clientes — o modelo comum de estação base não permite comunicação direta entre os nós clientes.

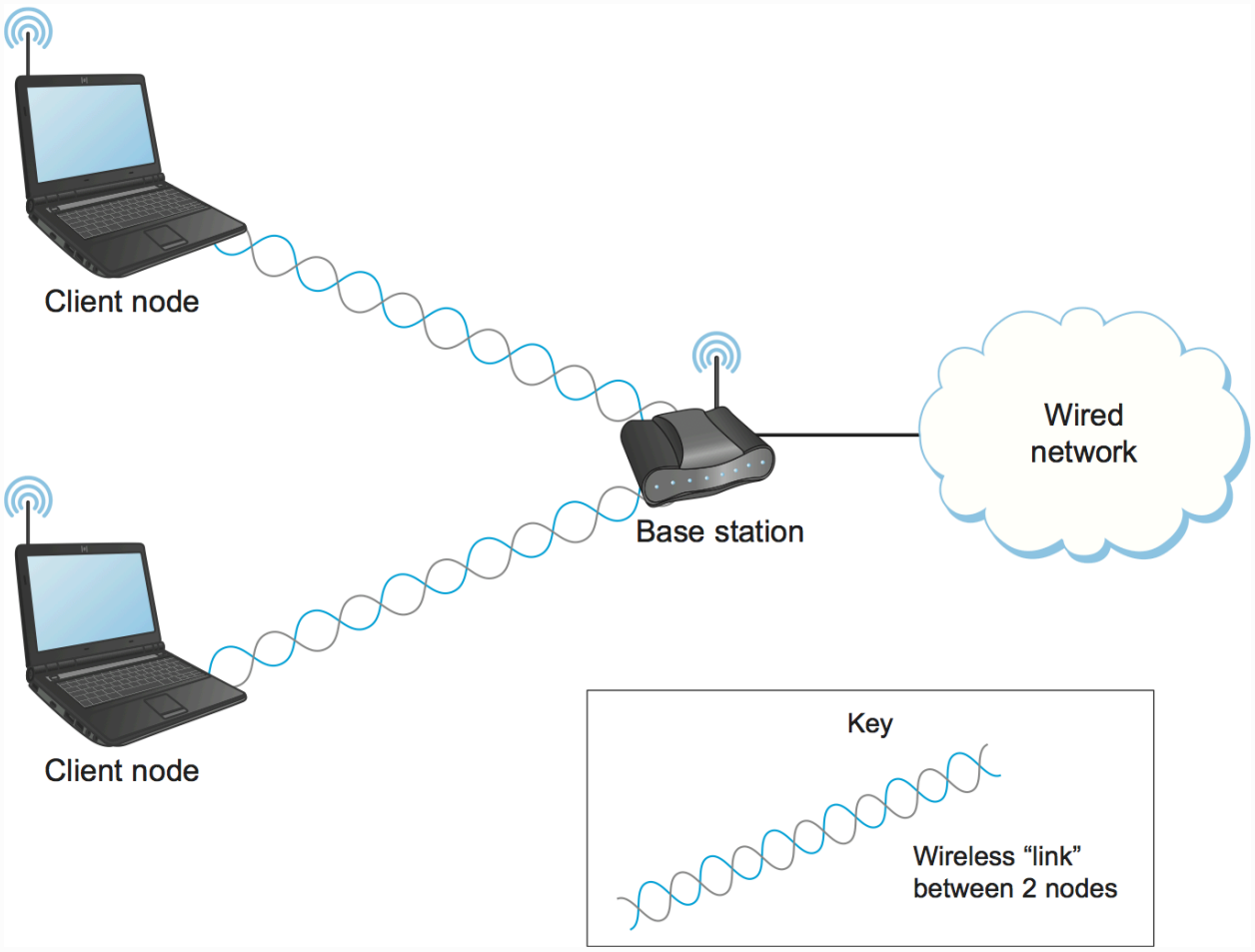

Figura 44. *Uma rede sem fio usando uma estação base.*

Essa topologia implica três níveis qualitativamente diferentes de mobilidade. O primeiro nível é a ausência de mobilidade, como quando um receptor precisa estar em um local fixo para receber uma transmissão direcional da estação base. O segundo nível é a mobilidade dentro do alcance de uma base, como é o caso do Bluetooth. O terceiro nível é a mobilidade entre bases, como é o caso de celulares e Wi-Fi.

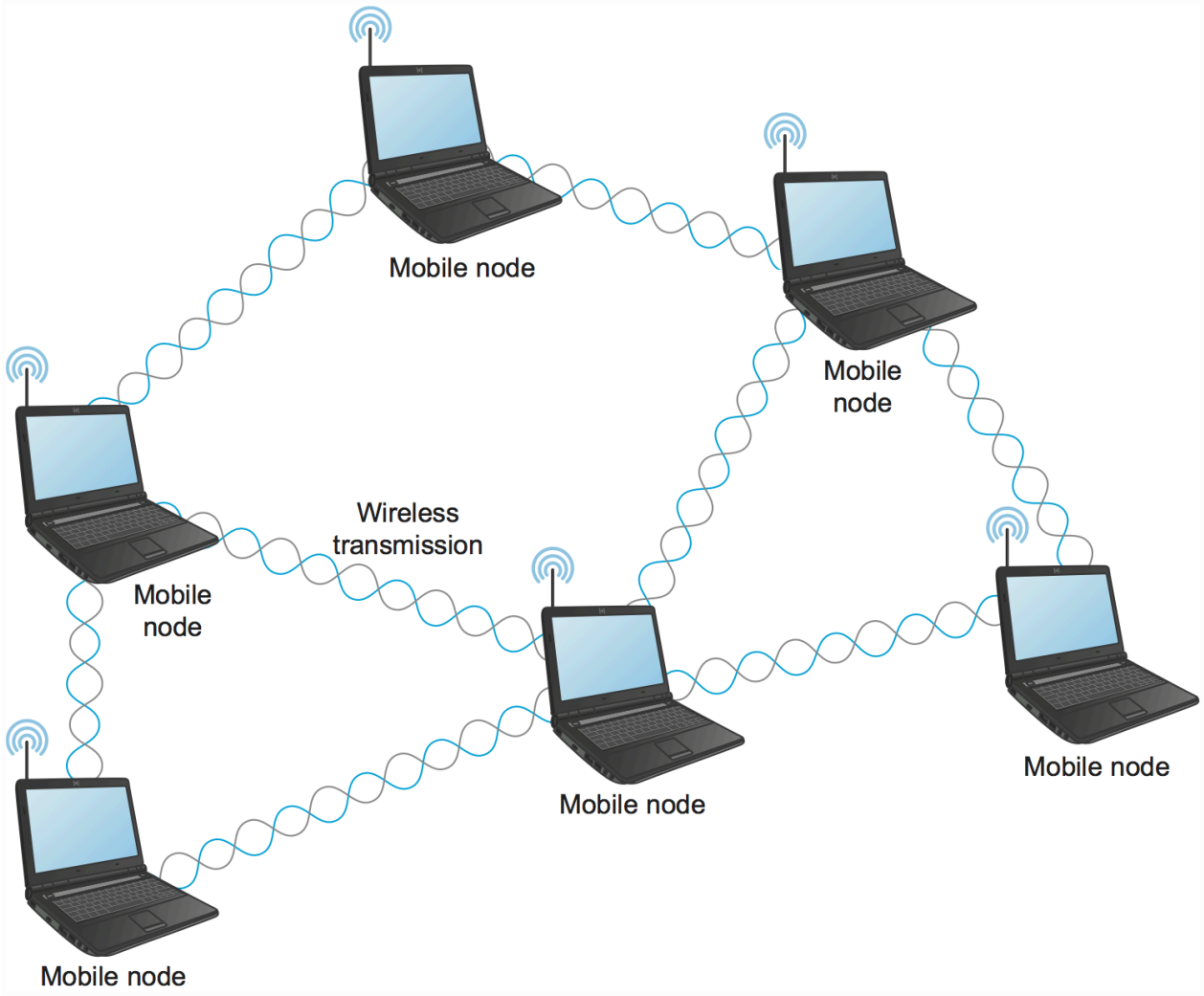

Figura 45. *Uma rede ad hoc ou mesh sem fio.*

Uma topologia alternativa que está despertando interesse crescente é a rede *mesh* ou *ad hoc* . Em uma malha sem fio, os nós são pares; ou seja, não há um nó de estação base específico. As mensagens podem ser encaminhadas por meio de uma cadeia de nós pares, desde que cada nó esteja dentro do alcance do nó anterior. Isso é ilustrado na Figura 45. Isso permite que a parte sem fio de uma rede se estenda além do alcance limitado de um único rádio. Do ponto de vista da competição entre tecnologias, isso permite que uma tecnologia de menor alcance estenda seu alcance e potencialmente concorra com uma tecnologia de maior alcance. As malhas também oferecem tolerância a falhas, fornecendo múltiplas rotas para uma mensagem ir do ponto A ao ponto B. Uma rede mesh pode ser estendida incrementalmente, com custos incrementais. Por outro lado, uma rede mesh exige que os nós não base tenham um

certo nível de sofisticação em seu hardware e software, aumentando potencialmente os custos por unidade e o consumo de energia, uma consideração crítica para dispositivos alimentados por bateria. As redes mesh sem fio são de considerável interesse em pesquisa, mas ainda estão em sua infância relativa em comparação com redes com estações base. Redes de sensores sem fio, outra tecnologia emergente, geralmente formam malhas sem fio.

Agora que abordamos alguns dos problemas comuns de redes sem fio, vamos dar uma olhada nos detalhes de duas tecnologias sem fio comuns.

2.7.2 Wi-Fi (802.11)

A maioria dos leitores já utilizou uma rede sem fio baseada nos padrões IEEE 802.11, frequentemente chamada de *Wi-Fi* . Tecnicamente, Wi-Fi é uma marca registrada, de propriedade de um grupo comercial chamado Wi-Fi Alliance, que certifica a conformidade dos produtos com o padrão 802.11. Assim como a Ethernet, o 802.11 foi projetado para uso em uma área geográfica limitada (residências, prédios comerciais, campi universitários) e seu principal desafio é mediar o acesso a um meio de comunicação compartilhado — neste caso, sinais que se propagam pelo espaço.

Propriedades físicas

802.11 define uma série de camadas físicas diferentes que operam em várias bandas de frequência e fornecem uma variedade de taxas de dados diferentes.

O padrão 802.11 original definiu dois padrões de camadas físicas baseados em rádio, um usando salto de frequência (mais de 79 larguras de banda de frequência de 1 MHz) e o outro usando espectro espalhado de sequência direta (com uma sequência de chipping de 11 bits). Ambos forneciam taxas de dados na faixa de 2 Mbps. Posteriormente, o padrão de camada física 802.11b foi adicionado e, usando uma variante de sequência direta, suportou até 11 Mbps. Todos esses três padrões operavam na faixa de frequência de 2,4 GHz isenta de licença do espectro

eletromagnético. Depois veio o 802.11a, que entregava até 54 Mbps usando uma variante de multiplexação por divisão de frequência chamada *multiplexação por divisão de frequência ortogonal (OFDM)* . O 802.11a roda na faixa de 5 GHz isenta de licença. O 802.11g veio em seguida, também usando OFDM, entregando até 54 Mbps. 802.11g é compatível com versões anteriores do 802.11b (e retorna à banda de 2,4 GHz).

No momento da redação deste texto, muitos dispositivos suportam 802.11n ou 802.11ac, que normalmente alcançam taxas de dados por dispositivo de 150 Mbps a 450 Mbps, respectivamente. Essa melhoria se deve, em parte, ao uso de múltiplas antenas e à possibilidade de maiores larguras de banda nos canais sem fio. O uso de múltiplas antenas é frequentemente chamado de *MIMO,* sigla em inglês para multiple-input, multiple-output (entradas e saídas múltiplas). O padrão emergente mais recente, 802.11ax, promete outra melhoria substancial na taxa de transferência, em parte pela adoção de muitas das técnicas de codificação e modulação usadas na rede celular 4G/5G, que descreveremos na próxima seção.

É comum que produtos comerciais suportem mais de uma variante do padrão 802.11; muitas estações base suportam todas as cinco variantes (a, b, g, n e ac). Isso não só garante a compatibilidade com qualquer dispositivo que suporte qualquer um dos padrões, como também permite que dois desses produtos escolham a opção de maior largura de banda para um ambiente específico.

Vale ressaltar que, embora todos os padrões 802.11 definam uma taxa de bits *máxima* que pode ser suportada, a maioria deles também suporta taxas de bits mais baixas (por exemplo, o 802.11a permite taxas de bits de 6, 9, 12, 18, 24, 36, 48 e 54 Mbps). Em taxas de bits mais baixas, é mais fácil decodificar os sinais transmitidos na presença de ruído. Diferentes esquemas de modulação são usados para atingir as diferentes taxas de bits. Além disso, a quantidade de informações redundantes na forma de códigos de correção de erros é variada. Mais informações redundantes significam maior resiliência a erros de bits, ao custo de reduzir a taxa de dados efetiva (já que mais bits transmitidos são redundantes).

Os sistemas tentam escolher uma taxa de bits ideal com base no ambiente de ruído em que se encontram; os algoritmos para seleção de taxa de bits podem ser bastante complexos. Curiosamente, os padrões 802.11 não especificam uma abordagem específica, mas deixam os algoritmos a cargo dos diversos fornecedores. A abordagem básica para escolher uma taxa de bits é estimar a taxa de erro de bits medindo diretamente a relação sinal-ruído (SNR) na camada física ou estimando a SNR medindo a frequência com que os pacotes são transmitidos e confirmados com sucesso. Em algumas abordagens, um remetente ocasionalmente testará uma taxa de bits mais alta enviando um ou mais pacotes nessa taxa para verificar se a taxa é bem-sucedida.

Prevenção de colisões

À primeira vista, pode parecer que um protocolo sem fio seguiria o mesmo algoritmo da Ethernet — esperar até que o link fique ocioso antes de transmitir e recuar caso ocorra uma colisão — e, em uma primeira aproximação, é isso que o 802.11 faz. A complicação adicional para a rede sem fio é que, enquanto um nó em uma Ethernet recebe as transmissões de todos os outros nós e pode transmitir e receber ao mesmo tempo, nenhuma dessas condições se aplica aos nós sem fio. Isso torna a detecção de colisões bastante mais complexa. A razão pela qual os nós sem fio geralmente não conseguem transmitir e receber ao mesmo tempo (na mesma frequência) é que a potência gerada pelo transmissor é muito maior do que qualquer recebido provavelmente será, sobrecarregando assim o circuito receptor. A razão pela qual um nó pode não receber transmissões de outro nó é porque esse nó pode estar muito distante ou bloqueado por um obstáculo. Essa situação é um pouco mais complexa do que parece à primeira vista, como a discussão a seguir ilustrará.

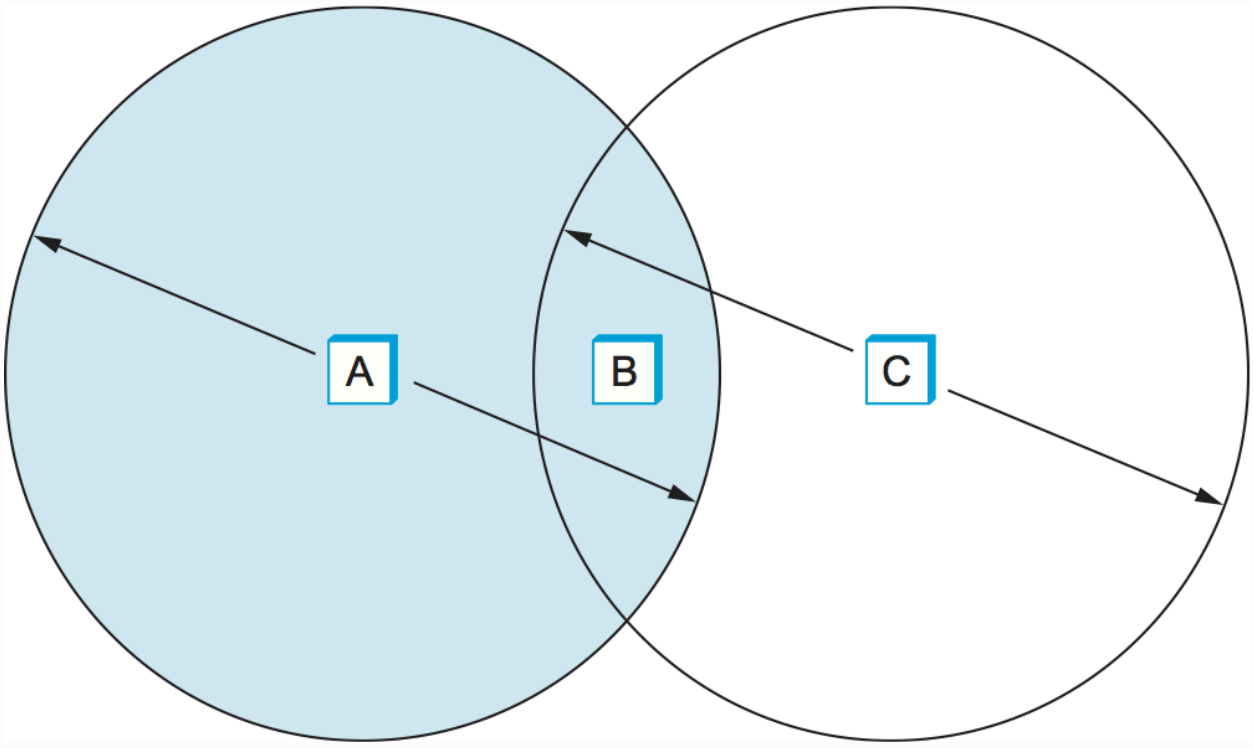

Figura 46. *O problema do nó oculto. Embora A e C estejam ocultos um do outro, seus sinais podem colidir em B. (O alcance de B não é mostrado.)*

Considere a situação representada na Figura 46 , onde A e C estão dentro do alcance de B, mas não um do outro. Suponha que A e C queiram se comunicar com B e, portanto, enviem um quadro cada. A e C não têm conhecimento um do outro, pois seus sinais não alcançam essa distância. Esses dois quadros colidem em B, mas, diferentemente de uma Ethernet, nem A nem C têm conhecimento dessa colisão. Diz-se que A e C são *nós ocultos* em relação um ao outro.

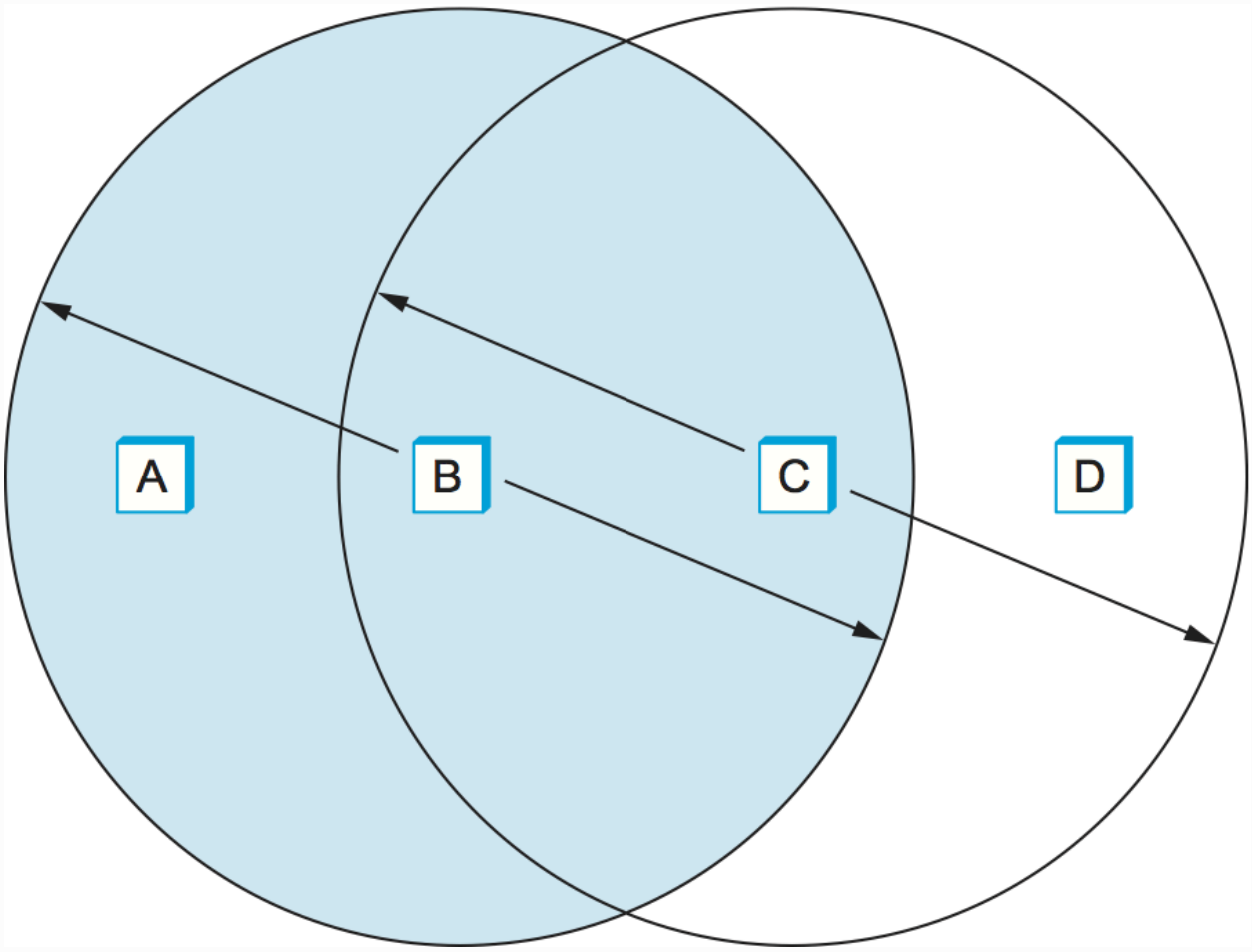

Figura 47. *O problema do nó exposto. Embora B e C estejam expostos aos sinais um do outro, não há interferência se B transmite para A enquanto C transmite para D. (Os alcances de A e D não são mostrados.)*

Um problema relacionado, chamado de *problema do nó exposto* , ocorre nas circunstâncias ilustradas na Figura 47 , onde cada um dos quatro nós é capaz de enviar e receber sinais que alcançam apenas os nós imediatamente à sua esquerda e direita. Por exemplo, B pode trocar quadros com A e C, mas não consegue alcançar D, enquanto C consegue alcançar B e D, mas não A. Suponha que B esteja enviando para A. O nó C está ciente dessa comunicação porque ouve a transmissão de B. Seria um erro, no entanto, para C concluir que não pode transmitir para ninguém apenas porque consegue ouvir a transmissão de B. Por exemplo, suponha que C queira transmitir para o nó D. Isso não é um problema, pois a transmissão de C para D não interferirá na capacidade de A de receber de B. (Isso interferiria no envio de A para B, mas B está transmitindo em nosso exemplo.)

O 802.11 resolve esses problemas usando CSMA/CA, onde CA significa *prevenção* de colisões , em contraste com a *detecção* de colisões do CSMA/CD usada em Ethernets. Há algumas etapas para fazer isso funcionar.

A parte do Carrier Sense parece bastante simples: antes de enviar um pacote, o transmissor verifica se consegue ouvir outras transmissões; caso contrário, ele envia. No entanto, devido ao problema do nó oculto, apenas esperar pela ausência de sinais de outros transmissores não garante que uma colisão não ocorrerá da perspectiva do receptor. Por esse motivo, uma parte do CSMA/CA é um ACK explícito do receptor para o remetente. Se o pacote foi decodificado com sucesso e passou seu CRC no receptor, o receptor envia um ACK de volta para o remetente.

Observe que, se ocorrer uma colisão, o pacote inteiro será inutilizado. Por esse motivo, o 802.11 adiciona um mecanismo opcional chamado RTS-CTS (Pronto para Enviar - Livre para Enviar). Isso contribui para resolver o problema do nó oculto. O remetente envia um RTS — um pacote curto — ao receptor pretendido e, se esse pacote for recebido com sucesso, o receptor responde com outro pacote curto, o CTS. Mesmo que o RTS não tenha sido ouvido por um nó oculto, o CTS provavelmente será. Isso efetivamente informa aos nós dentro do alcance do receptor que eles não devem enviar nada por um tempo — o tempo da transmissão pretendida está incluído nos pacotes RTS e CTS. Após esse tempo, mais um pequeno intervalo, pode-se presumir que a portadora está disponível novamente e outro nó está livre para tentar enviar.

É claro que dois nós podem detectar um link ocioso e tentar transmitir um quadro RTS ao mesmo tempo, causando a colisão de seus quadros RTS. Os remetentes percebem a colisão quando não recebem o quadro CTS após um período de tempo, e nesse caso cada um aguarda um tempo aleatório antes de tentar novamente. O tempo de atraso de um determinado nó é definido por um algoritmo de backoff exponencial muito semelhante ao usado na Ethernet.

Após uma troca RTS-CTS bem-sucedida, o remetente envia seu pacote de dados e, se tudo correr bem, recebe um ACK para esse pacote. Na ausência de um ACK oportuno,

o remetente tentará novamente solicitar o uso do canal, usando o mesmo processo descrito acima. Nesse momento, é claro, outros nós podem estar tentando acessar o canal novamente.

Sistema de Distribuição

Conforme descrito até agora, o padrão 802.11 seria adequado para uma rede com topologia mesh (*ad hoc*), e o desenvolvimento de um padrão 802.11 para redes mesh está quase concluído. Atualmente, no entanto, quase todas as redes 802.11 utilizam uma topologia orientada para estações base.

Em vez de todos os nós serem criados iguais, alguns nós podem circular livremente (por exemplo, seu laptop) e outros são conectados a uma infraestrutura de rede cabeada. O padrão 802.11 chama essas estações base *de pontos de acesso* (APs), e elas são conectadas entre si por um *sistema de distribuição* . A Figura 48 ilustra um sistema de distribuição que conecta três pontos de acesso, cada um dos quais atende os nós em alguma região. Cada ponto de acesso opera em algum canal na faixa de frequência apropriada, e cada AP normalmente estará em um canal diferente de seus vizinhos.

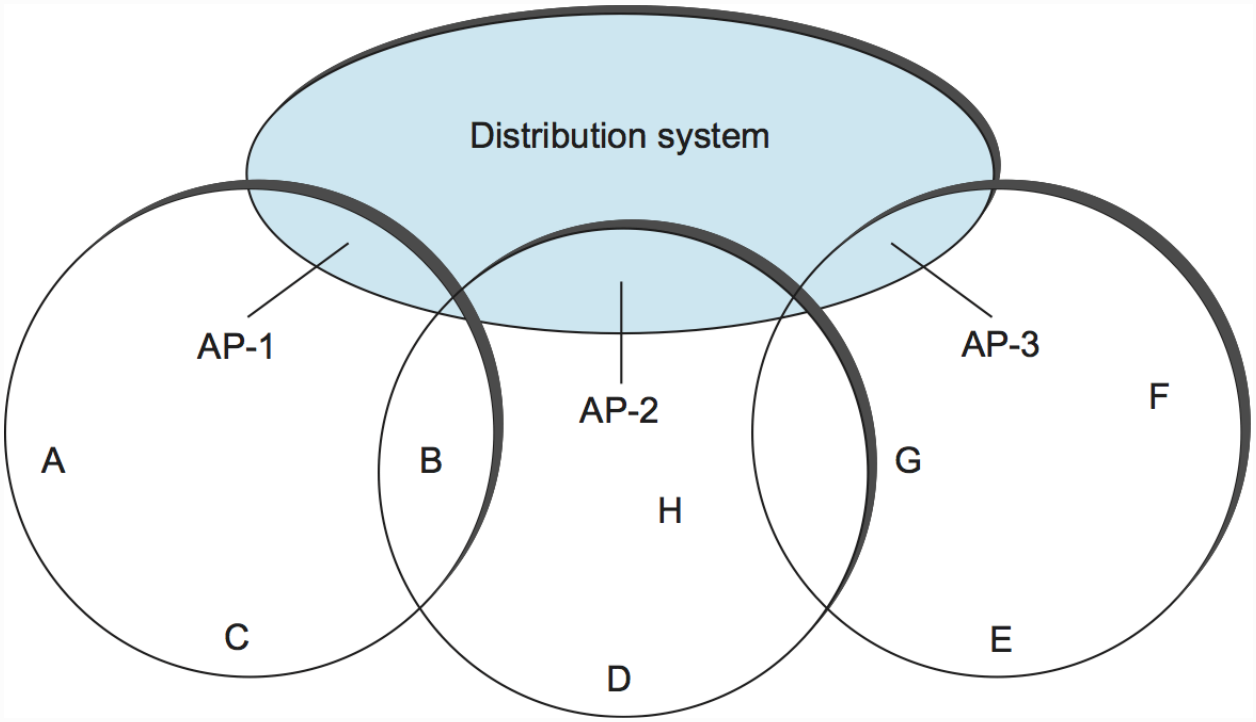

Figura 48. *Pontos de acesso conectados a um sistema de distribuição.*

Os detalhes do sistema de distribuição não são importantes para esta discussão — poderia ser uma Ethernet, por exemplo. O único ponto importante é que a rede de distribuição opera na camada de enlace, a mesma camada de protocolo dos enlaces sem fio. Em outras palavras, ela não depende de nenhum protocolo de nível superior (como a camada de rede).

Embora dois nós possam se comunicar diretamente se estiverem próximos um do outro, a ideia por trás dessa configuração é que cada nó se associe a um ponto de acesso. Para que o nó A se comunique com o nó E, por exemplo, A primeiro envia um quadro para seu ponto de acesso (AP-1), que o encaminha através do sistema de distribuição para o AP-3, que finalmente o transmite para E. Como o AP-1 sabia que deveria encaminhar a mensagem para o AP-3 está além do escopo do 802.11; ele pode ter usado um protocolo de ponte. O que o 802.11 especifica é como os nós selecionam seus pontos de acesso e, mais interessante, como esse algoritmo funciona considerando a movimentação dos nós de uma célula para outra.

A técnica para selecionar um AP é chamada *de digitalização* e envolve as quatro etapas a seguir:

1. O nó envia um `Probe`quadro.
2. Todos os APs ao alcance respondem com um quadro.`Probe Response`
3. O nó seleciona um dos pontos de acesso e envia um quadro a esse AP.`Association Request`
4. O AP responde com um quadro.`Association Response`

Um nó utiliza este protocolo sempre que se conecta à rede, bem como quando fica insatisfeito com seu AP atual. Isso pode acontecer, por exemplo, porque o sinal do AP atual enfraqueceu devido ao afastamento do nó. Sempre que um nó adquire um novo AP, o novo AP notifica o antigo sobre a mudança (isso acontece na etapa 4) por meio do sistema de distribuição.

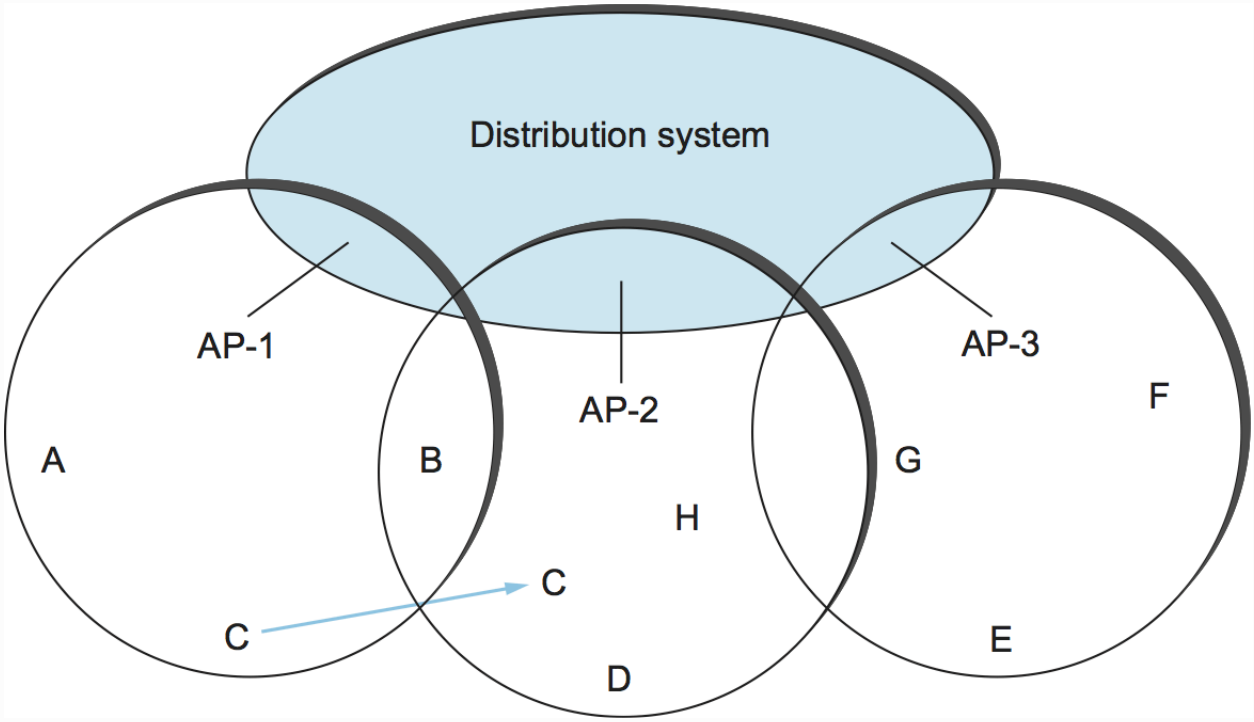

Figura 49. *Mobilidade do nó.*

Considere a situação mostrada na Figura 49 , onde o nó C se move da célula atendida pelo AP-1 para a célula atendida pelo AP-2. À medida que se move, ele envia

`Probe`quadros, que eventualmente resultam em quadros do AP-2. Em algum momento, C prefere o AP-2 ao AP-1 e, portanto, associa-se a esse ponto de acesso.`Probe Response`

O mecanismo descrito acima é chamado de *varredura ativa* , pois o nó está procurando ativamente por um ponto de acesso. Os APs também enviam periodicamente um `Beacon`quadro que anuncia as capacidades do ponto de acesso; estas incluem as taxas de transmissão suportadas pelo AP. Isso é chamado de *varredura passiva* , e um nó pode mudar para esse AP com base no `Beacon`quadro simplesmente enviando um quadro de volta ao ponto de acesso.`Association Request`

Formato do quadro

A maior parte do formato de quadro 802.11, representado na Figura 50 , é exatamente o que esperaríamos. O quadro contém os endereços dos nós de origem e destino, cada um com 48 bits de comprimento; até 2.312 bytes de dados; e um CRC de 32 bits. O `Control`campo contém três subcampos de interesse (não mostrados): um campo de 6 bits `Type`que indica se o quadro transporta dados, é um quadro RTS ou CTS, ou está sendo usado pelo algoritmo de varredura, e um par de campos de 1 bit — chamados `ToDS`e `FromDS`— que são descritos abaixo.

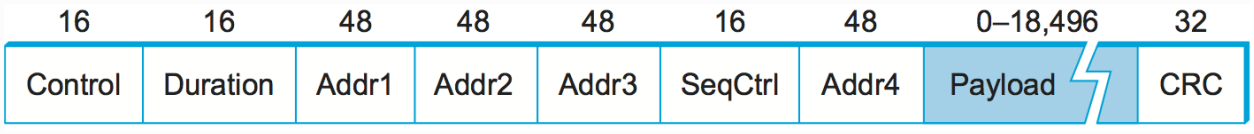

Figura 50. *Formato de quadro 802.11.*

O aspecto peculiar do formato de quadro 802.11 é que ele contém quatro endereços, em vez de dois. A interpretação desses endereços depende das configurações dos bits `ToDS`e `FromDS`no `Control`campo do quadro. Isso leva em conta a possibilidade de o quadro ter que ser encaminhado pelo sistema de distribuição, o que significaria que o remetente original não é necessariamente o mesmo que o nó transmissor mais recente. Raciocínio semelhante se aplica ao endereço de destino. No caso mais simples, quando um nó está enviando diretamente para outro, ambos os `DS`bits são 0,

`Addr1`identificando o nó de destino e `Addr2`o nó de origem. No caso mais complexo, ambos `DS`os bits são definidos como 1, indicando que a mensagem foi de um nó sem fio para o sistema de distribuição e, em seguida, do sistema de distribuição para outro nó sem fio. Com ambos os bits definidos, `Addr1`identifica o destino final, `Addr2` identifica o remetente imediato (aquele que encaminhou o quadro do sistema de distribuição para o destino final), `Addr3` identifica o destino intermediário (aquele que aceitou o quadro de um nó sem fio e o encaminhou pelo sistema de distribuição) e `Addr4`identifica a origem original. Em termos do exemplo dado na Figura 48 , `Addr1` corresponde a E, `Addr2`identifica AP-3, `Addr3`corresponde a AP-1 e `Addr4`identifica A.

Segurança de Links Sem Fio

Um dos problemas mais óbvios das conexões sem fio em comparação com fios ou fibras ópticas é que você não pode ter certeza de para onde seus dados foram. Você provavelmente consegue descobrir se eles foram recebidos pelo destinatário pretendido, mas não há como saber quantos outros destinatários também podem ter captado sua transmissão. Portanto, se você se preocupa com a privacidade dos seus dados, as redes sem fio representam um desafio.

Mesmo que você não se preocupe com a privacidade dos dados — ou talvez tenha cuidado disso de alguma outra forma —, você pode estar preocupado com a possibilidade de um usuário não autorizado injetar dados na sua rede. No mínimo, esse usuário pode consumir recursos que você preferiria consumir, como a largura de banda limitada entre sua casa e seu provedor de internet.

Por essas razões, as redes sem fio geralmente vêm com algum tipo de mecanismo para controlar o acesso tanto ao link em si quanto aos dados transmitidos. Esses mecanismos são frequentemente categorizados como *segurança sem fio.* O WPA2, amplamente adotado, é descrito no Capítulo 8.

2.7.3 Bluetooth (802.15.1)

O Bluetooth preenche o nicho de comunicação de curtíssimo alcance entre celulares, PDAs, notebooks e outros dispositivos pessoais ou periféricos. Por exemplo, o Bluetooth pode ser usado para conectar um celular a um fone de ouvido ou um notebook a um teclado. Em termos gerais, o Bluetooth é uma alternativa mais conveniente do que conectar dois dispositivos com um fio. Em tais aplicações, não é necessário fornecer muito alcance ou largura de banda. Isso significa que os rádios Bluetooth podem usar transmissão de energia bastante baixa, já que a potência de transmissão é um dos principais fatores que afetam a largura de banda e o alcance dos links sem fio. Isso corresponde às aplicações-alvo para dispositivos habilitados para Bluetooth — a maioria deles é alimentada por bateria (como o onipresente fone de ouvido de telefone) e, portanto, é importante que eles não consumam muita energia.

O Bluetooth opera na banda isenta de licença de 2,45 GHz. Os links Bluetooth têm larguras de banda típicas em torno de 1 a 3 Mbps e um alcance de cerca de 10 m. Por esse motivo, e como os dispositivos de comunicação geralmente pertencem a um indivíduo ou grupo, o Bluetooth às vezes é categorizado como uma Rede de Área Pessoal (PAN).

O Bluetooth é especificado por um consórcio da indústria chamado *Bluetooth Special Interest Group* . Ele especifica um conjunto completo de protocolos, indo além da camada de enlace para definir protocolos de aplicação, que ele chama *de perfis* , para uma variedade de aplicações. Por exemplo, há um perfil para sincronizar um PDA com um computador pessoal. Outro perfil fornece a um computador móvel acesso a uma LAN com fio, como no padrão 802.11, embora este não fosse o objetivo original do Bluetooth. O padrão IEEE 802.15.1 é baseado no Bluetooth, mas exclui os protocolos de aplicação.

A configuração básica de uma rede Bluetooth, chamada *piconet* , consiste em um dispositivo mestre e até sete dispositivos escravos, como mostrado na Figura 51. Toda a comunicação ocorre entre o mestre e um escravo; os escravos não se comunicam diretamente entre si. Como os escravos têm uma função mais simples, seu hardware e software Bluetooth podem ser mais simples e baratos.

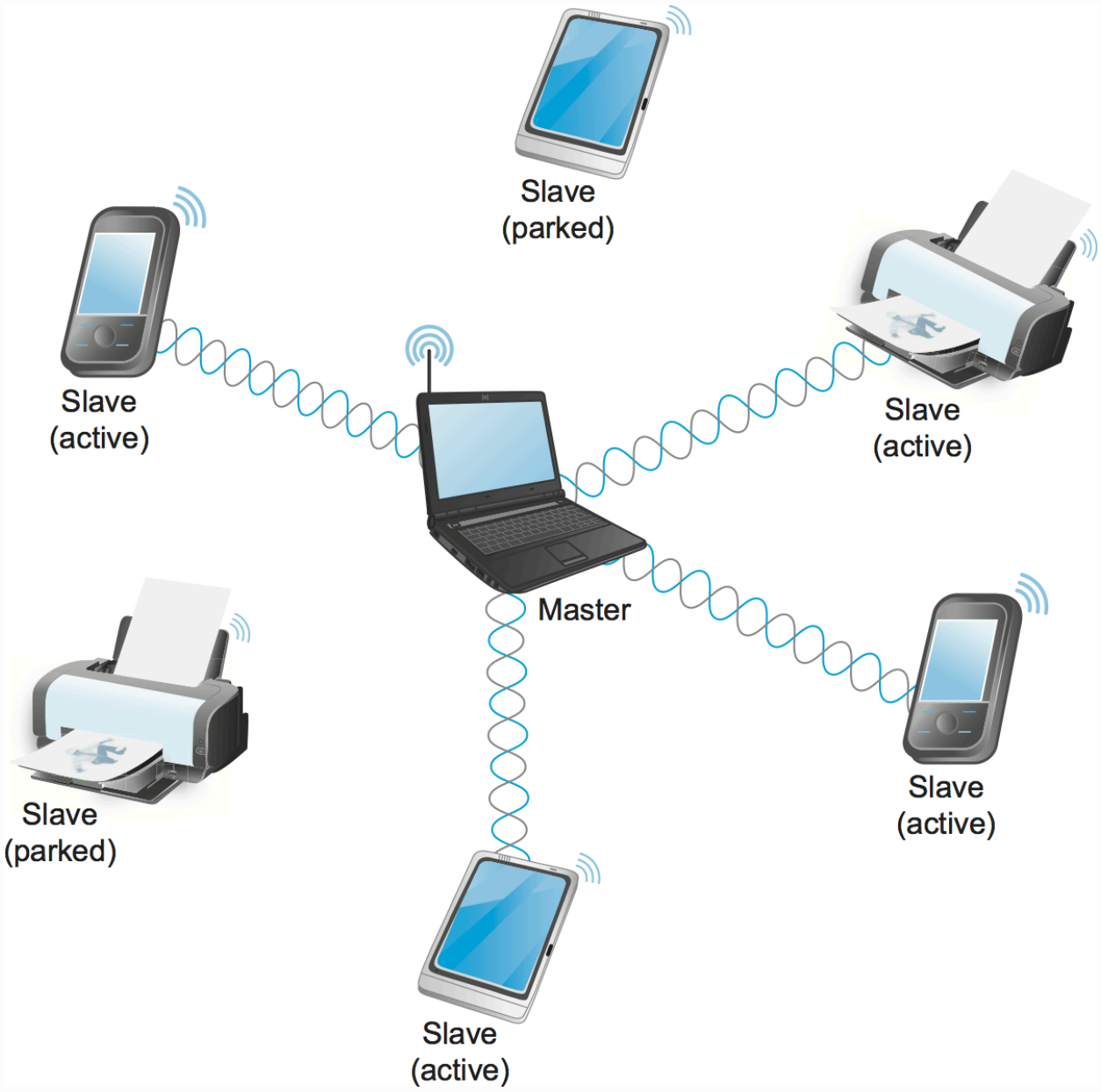

Figura 51. *Uma piconet Bluetooth.*

Como o Bluetooth opera em uma banda isenta de licença, é necessário usar uma técnica de espectro espalhado para lidar com possíveis interferências na banda. Ela utiliza salto de frequência com 79 *canais* (frequências), usando cada um por 625 μs por vez. Isso fornece um intervalo de tempo natural para o Bluetooth usar para multiplexação síncrona por divisão de tempo. Um quadro ocupa 1, 3 ou 5 intervalos de tempo consecutivos. Somente o mestre pode começar a transmitir em intervalos ímpares. Um escravo pode começar a transmitir em um intervalo par — mas somente

em resposta a uma solicitação do mestre durante o intervalo anterior, evitando assim qualquer contenção entre os dispositivos escravos.

Um dispositivo escravo pode ser *estacionado* , ou seja, configurado para um estado inativo e de baixo consumo de energia. Um dispositivo estacionado não pode se comunicar na piconet; ele só pode ser reativado pelo mestre. Uma piconet pode ter até 255 dispositivos estacionados, além dos dispositivos escravos ativos.

No âmbito da comunicação de baixíssimo consumo e curto alcance, existem algumas outras tecnologias além do Bluetooth. Uma delas é o ZigBee, desenvolvido pela ZigBee Alliance e padronizado como IEEE 802.15.4. Ele foi projetado para situações em que os requisitos de largura de banda são baixos e o consumo de energia deve ser muito baixo para proporcionar uma vida útil da bateria muito longa. Também se destina a ser mais simples e barato que o Bluetooth, tornando-o viável para incorporação em dispositivos mais baratos, como *sensores* . Os sensores estão se tornando uma classe cada vez mais importante de dispositivos em rede, à medida que a tecnologia avança a ponto de pequenos dispositivos muito baratos poderem ser implantados em grandes quantidades para monitorar aspectos como temperatura, umidade e consumo de energia em um edifício.

2.8 Redes de Acesso

Além das conexões Ethernet e Wi-Fi que normalmente usamos para nos conectar à internet em casa, no trabalho, na escola e em muitos espaços públicos, a maioria de nós se conecta à internet por meio de um serviço *de acesso* ou *banda larga* adquirido de um provedor de internet (ISP). Esta seção descreve duas dessas tecnologias: *Redes Ópticas Passivas* (PON), comumente chamadas de fibra óptica até a casa, e *Redes Celulares* , que conectam nossos dispositivos móveis. Em ambos os casos, as redes são multiacesso (como Ethernet e Wi-Fi), mas, como veremos, suas abordagens para mediar o acesso são bem diferentes.

Para contextualizar um pouco mais, os ISPs (por exemplo, empresas de telecomunicações ou de TV a cabo) frequentemente operam um backbone nacional, e centenas ou milhares de pontos de ponta estão conectados à periferia desse backbone, cada um atendendo a uma cidade ou bairro. Esses pontos de ponta são comumente chamados de *Escritórios Centrais* no mundo das telecomunicações e *Head-Ends* no mundo da TV a cabo, mas, apesar de seus nomes implicarem "centralizados" e "na raiz da hierarquia", esses pontos estão na extremidade da rede do ISP; o lado do ISP da última milha que se conecta diretamente aos clientes. Redes de acesso PON e celular são ancoradas nessas instalações. [1]

[1]

DSL é a versão legada, baseada em cobre, da PON. Os links DSL também são encerrados em Centrais Telefônicas, mas não descrevemos essa tecnologia, pois ela está sendo descontinuada.

2.8.1 Rede Óptica Passiva

PON é a tecnologia mais comumente usada para fornecer banda larga baseada em fibra para residências e empresas. A PON adota um design ponto-a-multiponto, o que significa que a rede é estruturada como uma árvore, com um único ponto começando na rede do ISP e então se espalhando para alcançar até 1.024 residências. A PON recebe esse nome porque os divisores são passivos: eles encaminham sinais ópticos downstream e upstream sem armazenar e encaminhar quadros ativamente. Dessa forma, eles são a variante óptica dos repetidores usados na Ethernet clássica. O enquadramento então ocorre na fonte, nas instalações do ISP, em um dispositivo chamado *Terminal de Linha Óptica* (OLT), e nos pontos finais em residências individuais, em um dispositivo chamado *Unidade de Rede Óptica* (ONU).

A Figura 52 mostra um exemplo de PON, simplificado para representar apenas uma ONU e uma OLT. Na prática, uma Central Telefônica incluiria várias OLTs

conectando-se a milhares de residências de clientes. Para completar, a Figura 52 também inclui dois outros detalhes sobre como a PON está conectada ao backbone do ISP (e, portanto, ao restante da Internet). O *Agg Switch* agrega o tráfego de um conjunto de OLTs, e o *BNG* (Broadband Network Gateway) é um equipamento de telecomunicações que, entre muitas outras coisas, mede o tráfego de Internet para fins de faturamento. Como o próprio nome indica, o BNG é efetivamente o gateway entre a rede de acesso (tudo à esquerda do BNG) e a Internet (tudo à direita do BNG).

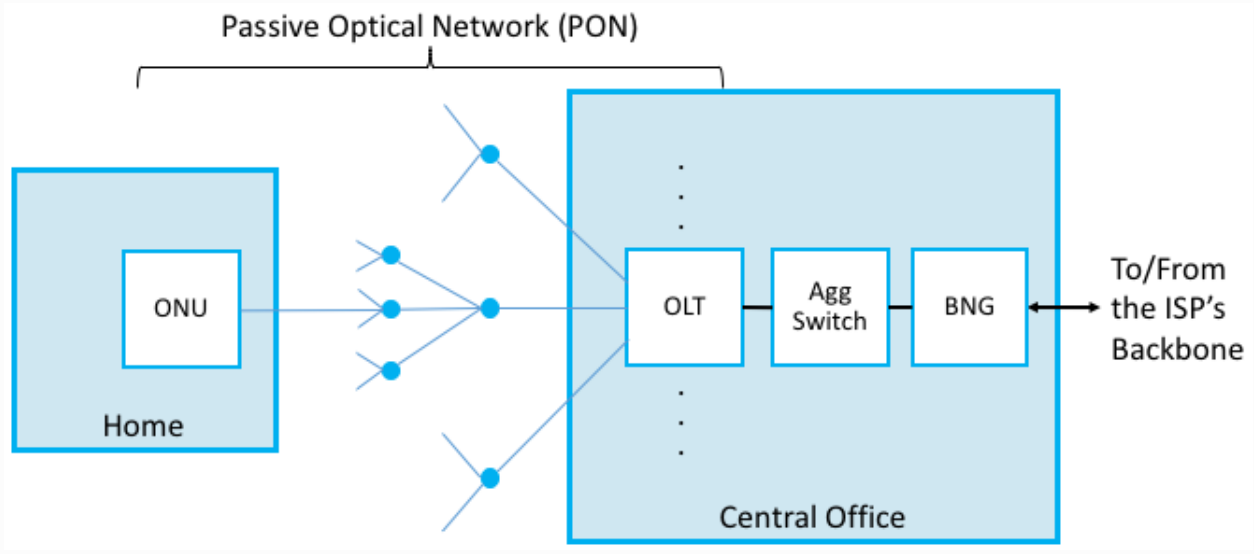

Figura 52. *Um exemplo de PON que conecta OLTs no escritório central a ONUs em residências e empresas.*

Como os divisores são passivos, a PON precisa implementar algum tipo de protocolo de multiacesso. A abordagem adotada pode ser resumida da seguinte forma. Primeiro, o tráfego upstream e downstream é transmitido em dois comprimentos de onda ópticos diferentes, sendo completamente independentes um do outro. O tráfego downstream começa na OLT e o sinal é propagado por todos os links da PON. Como consequência, cada quadro chega a cada ONU. Este dispositivo então analisa um identificador único nos quadros individuais enviados pelo comprimento de onda e mantém o quadro (se o identificador for para ele) ou o descarta (caso contrário). A criptografia é usada para impedir que as ONUs espionem o tráfego de seus vizinhos.

O tráfego upstream é então multiplexado por divisão de tempo no comprimento de onda upstream, com cada ONU recebendo periodicamente uma vez para transmitir. Como as ONUs estão distribuídas por uma área bastante ampla (medida em quilômetros) e a diferentes distâncias do OLT, não é prático que elas transmitam com base em relógios sincronizados, como no SONET. Em vez disso, o OLT transmite *concessões* para as ONUs individuais, dando a elas um intervalo de tempo durante o qual podem transmitir. Em outras palavras, o OLT único é responsável por implementar centralmente o compartilhamento round-robin da PON compartilhada. Isso inclui a possibilidade de o OLT conceder a cada ONU uma parcela diferente de tempo, implementando efetivamente diferentes níveis de serviço.

PON é semelhante à Ethernet no sentido de que define um algoritmo de compartilhamento que evoluiu ao longo do tempo para acomodar larguras de banda cada vez maiores. G-PON (Gigabit-PON) é o mais amplamente utilizado atualmente, suportando uma largura de banda de 2,25 Gbps. XGS-PON (10 Gigabit-PON) está apenas começando a ser implementado.

2.8.2 Rede Celular

Embora a tecnologia de telefonia celular tenha suas raízes na comunicação de voz analógica, os serviços de dados baseados em padrões celulares são agora a norma. Assim como o Wi-Fi, as redes celulares transmitem dados em determinadas larguras de banda no espectro de rádio. Ao contrário do Wi-Fi, que permite que qualquer pessoa use um canal em 2,4 ou 5 GHz (basta configurar uma estação rádio-base, como muitos de nós fazemos em casa), o uso exclusivo de várias faixas de frequência foi leiloado e licenciado para provedores de serviços, que por sua vez vendem o serviço de acesso móvel aos seus assinantes.

As faixas de frequência utilizadas pelas redes celulares variam ao redor do mundo e são complicadas pelo fato de os ISPs frequentemente oferecerem suporte simultâneo a tecnologias antigas/legadas e novas/de próxima geração, cada uma ocupando uma faixa de frequência diferente. Em resumo, as tecnologias celulares tradicionais variam

de 700 MHz a 2.400 MHz, com novas alocações de espectro médio ocorrendo agora em 6 GHz e alocações em ondas milimétricas (mmWave) abrindo acima de 24 GHz.

Serviço de Rádio de Banda Larga para Cidadãos (CBRS)

> Além das faixas licenciadas, há também uma faixa não licenciada de 3,5 GHz reservada na América do Norte, chamada *Citizens Broadband Radio Service* (CBRS), que qualquer pessoa com um rádio celular pode usar. Faixas não licenciadas semelhantes também estão sendo instaladas em outros países. Isso abre caminho para a instalação de redes celulares privadas, por exemplo, dentro de um campus universitário, uma empresa ou uma fábrica.

> Para ser mais preciso, a banda CBRS permite que três níveis de usuários compartilhem o espectro: o primeiro direito de uso vai para os proprietários originais desse espectro, radares navais e estações terrestres de satélite; seguido pelos usuários prioritários que recebem esse direito sobre as bandas de 10 MHz por três anos por meio de leilões regionais; e, finalmente, o restante da população, que pode acessar e utilizar uma parte dessa banda, desde que primeiro verifique com um banco de dados central de usuários registrados.

Assim como o 802.11, a tecnologia celular depende do uso de estações rádio-base conectadas a uma rede cabeada. No caso da rede celular, as estações rádio-base são frequentemente chamadas de *Unidades Rádio-Base de Banda Larga* (BBU), os dispositivos móveis que se conectam a elas são geralmente chamados de *Equipamentos de Usuário* (UE), e o conjunto de BBUs é ancorado em um *Núcleo de Pacotes Evoluídos* (EPC) hospedado em um Escritório Central. A rede sem fio atendida pelo EPC é frequentemente chamada de *Rede de Acesso por Rádio* (RAN).

As BBUs são oficialmente conhecidas por outro nome: Evolved NodeB, frequentemente abreviado como eNodeB ou eNB — onde NodeB é o nome dado à unidade de rádio em uma das primeiras versões das redes celulares (e que evoluiu desde então). Considerando que o mundo celular continua a evoluir em ritmo acelerado e que as

eNBs em breve serão atualizadas para gNBs, decidimos usar a BBU, mais genérica e menos enigmática.

A Figura 53 descreve uma configuração possível do cenário ponta a ponta, com alguns detalhes adicionais. O EPC possui vários subcomponentes, incluindo uma MME (Entidade de Gerenciamento de Mobilidade), um HSS (Servidor de Assinante Doméstico) e um par S/PGW (Gateway de Sessão/Pacote); o primeiro rastreia e gerencia a movimentação de UEs pela RAN, o segundo é um banco de dados que contém informações relacionadas ao assinante, e o par Gateway processa e encaminha pacotes entre a RAN e a Internet (forma o *plano de usuário* do EPC). Dizemos "uma configuração possível" porque os padrões celulares permitem ampla variabilidade na quantidade de S/PGWs pelos quais uma determinada MME é responsável, possibilitando que uma única MME gerencie a mobilidade em uma ampla área geográfica atendida por várias Centrais. Por fim, embora não explicitamente explicado na Figura 53 , às vezes a rede PON do ISP é usada para conectar as BBUs remotas de volta à Central.

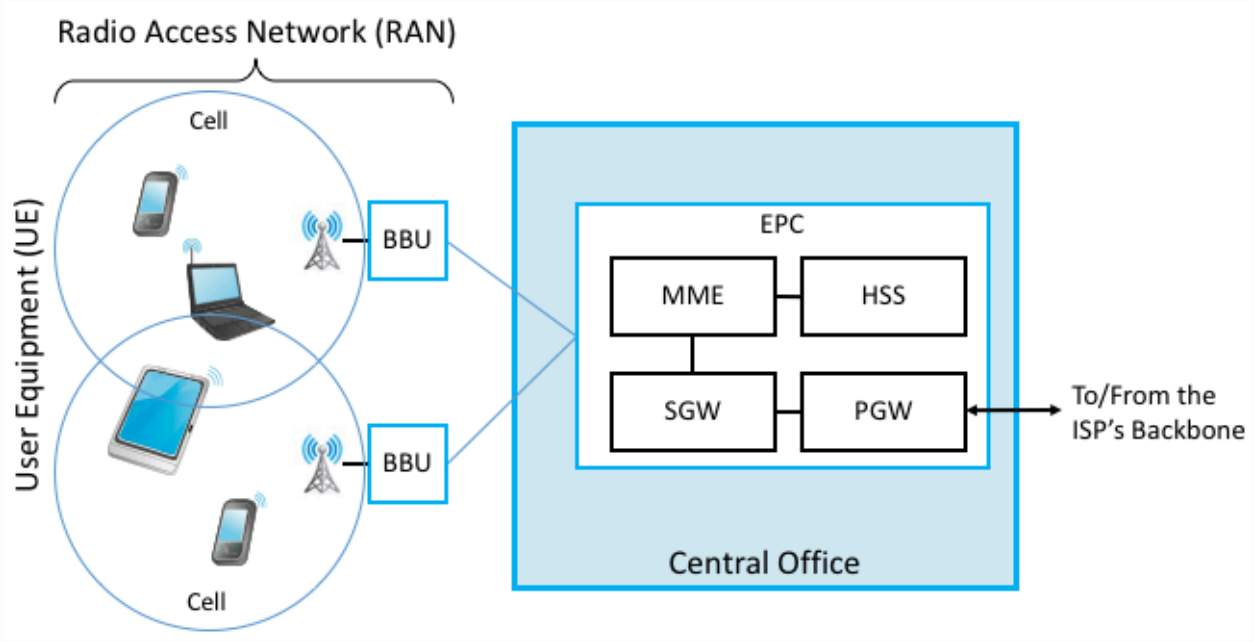

Figura 53. *Uma Rede de Acesso de Rádio (RAN) conectando um conjunto de dispositivos celulares (UEs) a um Núcleo de Pacote Evoluído (EPC) hospedado em um Escritório Central.*

A área geográfica atendida pela antena de uma BBU é chamada de *célula* . Uma BBU pode atender a uma única célula ou usar múltiplas antenas direcionais para atender a múltiplas células. As células não têm limites nítidos e se sobrepõem. Onde elas se sobrepõem, um UE pode potencialmente se comunicar com múltiplas BBUs. A qualquer momento, no entanto, o UE está em comunicação com, e sob o controle de, apenas uma BBU. Conforme o dispositivo começa a deixar uma célula, ele se move para uma área de sobreposição com uma ou mais outras células. A BBU atual detecta o sinal enfraquecido do telefone e dá o controle do dispositivo para qualquer estação base que esteja recebendo o sinal mais forte dele. Se o dispositivo estiver envolvido em uma chamada ou outra sessão de rede no momento, a sessão deve ser transferida para a nova estação base no que é chamado de *handoff* . O processo de tomada de decisão para handoffs está sob a alçada do MME, que historicamente tem sido um aspecto proprietário dos fornecedores de equipamentos celulares (embora implementações de MME de código aberto estejam começando a estar disponíveis).

Houve várias gerações de protocolos implementando a rede celular, coloquialmente conhecida como 1G, 2G, 3G e assim por diante. As duas primeiras gerações suportavam apenas voz, com o 3G definindo a transição para o acesso à banda larga, suportando taxas de dados medidas em centenas de quilobits por segundo. Hoje, o setor está no 4G (suportando taxas de dados tipicamente medidas em poucos megabits por segundo) e em processo de transição para o 5G (com a promessa de um aumento de dez vezes nas taxas de dados).

A partir do 3G, a designação geracional na verdade corresponde a um padrão definido pelo 3GPP (3rd Generation Partnership Project). Embora seu nome tenha "3G", o 3GPP continua a definir o padrão para 4G e 5G, cada um dos quais corresponde a uma versão do padrão. A versão 15, que agora é publicada, é considerada o ponto de demarcação entre 4G e 5G. Por outro nome, essa sequência de lançamentos e gerações é chamada de LTE, que significa *Long-Term Evolution* . A principal conclusão é que, embora os padrões sejam publicados como uma sequência de lançamentos discretos, a indústria como um todo tem seguido um caminho evolutivo bastante bem

definido conhecido como LTE. Esta seção usa a terminologia LTE, mas destaca as mudanças que vêm com o 5G quando apropriado.

A principal inovação da interface aérea do LTE é a forma como ele aloca o espectro de rádio disponível aos UEs. Ao contrário do Wi-Fi, que é baseado em contenção, o LTE utiliza uma estratégia baseada em reserva. Essa diferença está enraizada na premissa fundamental de cada sistema sobre a utilização: o Wi-Fi pressupõe uma rede com carga leve (e, portanto, transmite de forma otimista quando o link sem fio está ocioso e recua se a contenção for detectada), enquanto as redes celulares pressupõem (e buscam) alta utilização (e, portanto, atribuem explicitamente diferentes usuários a diferentes "partes" do espectro de rádio disponível).

O mecanismo de acesso à mídia de última geração para LTE é chamado de *Acesso Múltiplo por Divisão de Frequência Ortogonal (OFDMA)* . A ideia é multiplexar dados em um conjunto de 12 frequências de subportadoras ortogonais, cada uma modulada independentemente. O "Acesso Múltiplo" em OFDMA implica que os dados podem ser enviados simultaneamente em nome de vários usuários, cada um em uma frequência de subportadora diferente e por um período de tempo diferente. As subbandas são estreitas (por exemplo, 15 kHz), mas a codificação dos dados do usuário em símbolos OFDMA é projetada para minimizar o risco de perda de dados devido à interferência entre bandas adjacentes.

O uso de OFDMA naturalmente leva à conceituação do espectro de rádio como um recurso bidimensional, como mostrado na Figura 54. A unidade mínima programável, chamada de *Elemento de Recurso (ER)* , corresponde a uma banda de 15 kHz em torno de uma frequência de subportadora e ao tempo necessário para transmitir um símbolo OFDMA. O número de bits que podem ser codificados em cada símbolo depende da taxa de modulação; portanto, por exemplo, usando Modulação de Amplitude em Quadratura (QAM), 16-QAM produz 4 bits por símbolo e 64-QAM produz 6 bits por símbolo.

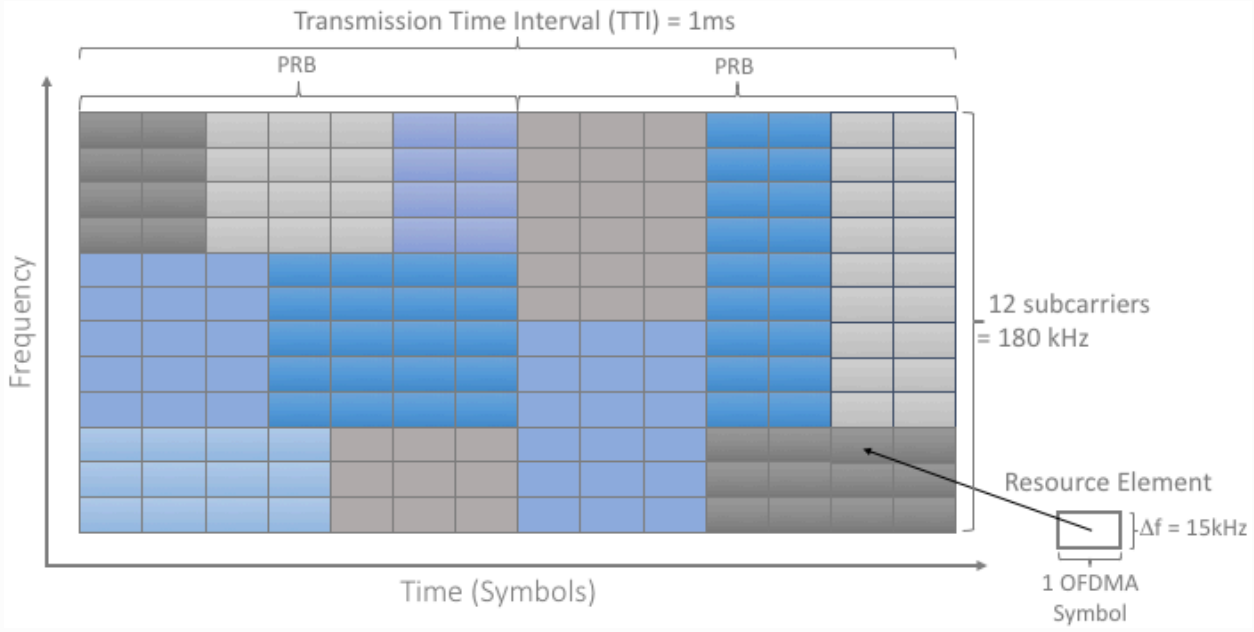

Figura 54. *O espectro de rádio disponível representado abstratamente por uma grade 2-D de Elementos de Recursos programáveis.*

Um escalonador toma decisões de alocação na granularidade de blocos de 7x12=84 elementos de recurso, denominado *Bloco de Recursos Físicos (PRB)* . A Figura 54 mostra dois PRBs consecutivos, onde os UEs são representados por blocos de cores diferentes. É claro que o tempo continua fluindo ao longo de um eixo e, dependendo do tamanho da banda de frequência licenciada, pode haver muito mais slots de subportadora (e, portanto, PRBs) disponíveis ao longo do outro eixo, de modo que o escalonador está essencialmente escalonando uma sequência de PRBs para transmissão.

O *Intervalo de Tempo de Transmissão (TTI)* de 1 ms mostrado na Figura 54 corresponde ao período em que a BBU recebe feedback dos UEs sobre a qualidade do sinal que estão recebendo. Esse feedback, chamado de *Indicador de Qualidade do Canal (CQI)* , basicamente relata a relação sinal-ruído observada, que impacta a capacidade do UE de recuperar os bits de dados. A estação base então usa essas informações para adaptar a forma como aloca o espectro de rádio disponível aos UEs que atende.

Até este ponto, a descrição de como agendamos o espectro de rádio é específica para o 4G. A transição do 4G para o 5G introduz graus de liberdade adicionais na forma como o espectro de rádio é agendado, possibilitando a adaptação da rede celular a um conjunto mais diversificado de dispositivos e domínios de aplicação.

Fundamentalmente, o 5G define uma família de formas de onda — diferentemente do 4G, que especificava apenas uma forma de onda — cada uma otimizada para uma banda diferente no espectro de rádio. [2] As bandas com frequências portadoras abaixo de 1 GHz são projetadas para fornecer banda larga móvel e serviços massivos de IoT, com foco principal no alcance. As frequências portadoras entre 1 GHz e 6 GHz são projetadas para oferecer larguras de banda mais amplas, com foco em banda larga móvel e aplicações de missão crítica. As frequências portadoras acima de 24 GHz (mmWaves) são projetadas para fornecer larguras de banda superlargas em coberturas curtas de linha de visão.

2

Uma forma de onda é a propriedade (formato) independente de frequência, amplitude e deslocamento de fase de um sinal. Uma onda senoidal é um exemplo de forma de onda.

Essas diferentes formas de onda afetam o agendamento e os intervalos das subportadoras (ou seja, o “tamanho” dos Elementos de Recurso descritos anteriormente).

- Para bandas abaixo de 1 GHz, o 5G permite larguras de banda máximas de 50 MHz. Nesse caso, há duas formas de onda: uma com espaçamento de subportadora de 15 kHz e outra de 30 kHz. (Usamos 15 kHz no exemplo mostrado na Figura 54. Os intervalos de escalonamento correspondentes são 0,5 ms e 0,25 ms, respectivamente. (Usamos 0,5 ms no exemplo mostrado na Figura 54.)

- Para as faixas de 1 GHz a 6 GHz, as larguras de banda máximas chegam a 100 MHz. Correspondentemente, existem três formas de onda com espaçamentos de subportadora de 15 kHz, 30 kHz e 60 kHz, correspondendo a intervalos de escalonamento de 0,5 ms, 0,25 ms e 0,125 ms, respectivamente.
- Para bandas milimétricas, as larguras de banda podem chegar a 400 MHz. Existem duas formas de onda, com espaçamentos de subportadora de 60 kHz e 120 kHz. Ambas têm intervalos de escalonamento de 0,125 ms.

Essa gama de opções é importante porque adiciona outro grau de liberdade ao agendador. Além de alocar blocos de recursos aos usuários, ele tem a capacidade de ajustar dinamicamente o tamanho dos blocos de recursos, alterando a forma de onda usada na banda pela qual é responsável pelo agendamento.

Seja 4G ou 5G, o algoritmo de escalonamento é um problema de otimização desafiador, com o objetivo de simultaneamente (a) maximizar a utilização da banda de frequência disponível e (b) garantir que cada UE receba o nível de serviço necessário. Este algoritmo não é especificado pelo 3GPP, mas sim propriedade intelectual exclusiva dos fornecedores de BBU.

Perspectiva: Corrida até o Limite

À medida que começamos a explorar como a softwarização está transformando a rede, devemos reconhecer que é a rede de acesso que conecta residências, empresas e usuários móveis à internet que está passando pela mudança mais radical. As redes de fibra óptica e celulares descritas na Seção 2.8 são atualmente construídas a partir de dispositivos de hardware complexos (por exemplo, OLTs, BNGs, BBUs, EPCs). Esses dispositivos não apenas eram historicamente fechados e proprietários, como também os fornecedores que os comercializam normalmente agregam um conjunto amplo e diversificado de funcionalidades em cada um. Como consequência, tornaram-se caros de construir, complicados de operar e lentos para mudar.

Em resposta, as operadoras de rede estão migrando ativamente desses dispositivos específicos para softwares abertos executados em servidores, switches e dispositivos de acesso comuns. Essa iniciativa costuma ser chamada de *CORD* , sigla para **Central Office Re** -architected as a **Datacenter** (Escritório Central Re -arquitetado como um Datacenter). Como o nome sugere, a ideia é construir a Central de Telecomunicações (ou a Head End do Cabo, resultando na sigla *HERD*) usando exatamente as mesmas tecnologias dos grandes datacenters que compõem a nuvem.

A motivação das operadoras para isso é, em parte, o benefício da economia de custos resultante da substituição de dispositivos específicos por hardware comum, mas também é impulsionada principalmente pela necessidade de acelerar o ritmo da inovação. Seu objetivo é habilitar novas classes de serviços de ponta — por exemplo, Segurança Pública, Veículos Autônomos, Fábricas Automatizadas, Internet das Coisas (IoT), Interfaces de Usuário Imersivas — que se beneficiam da conectividade de baixa latência para os usuários finais e, mais importante, para o número crescente de dispositivos com os quais esses usuários se cercam. Isso resulta em uma nuvem multicamadas semelhante à mostrada na Figura 55 .

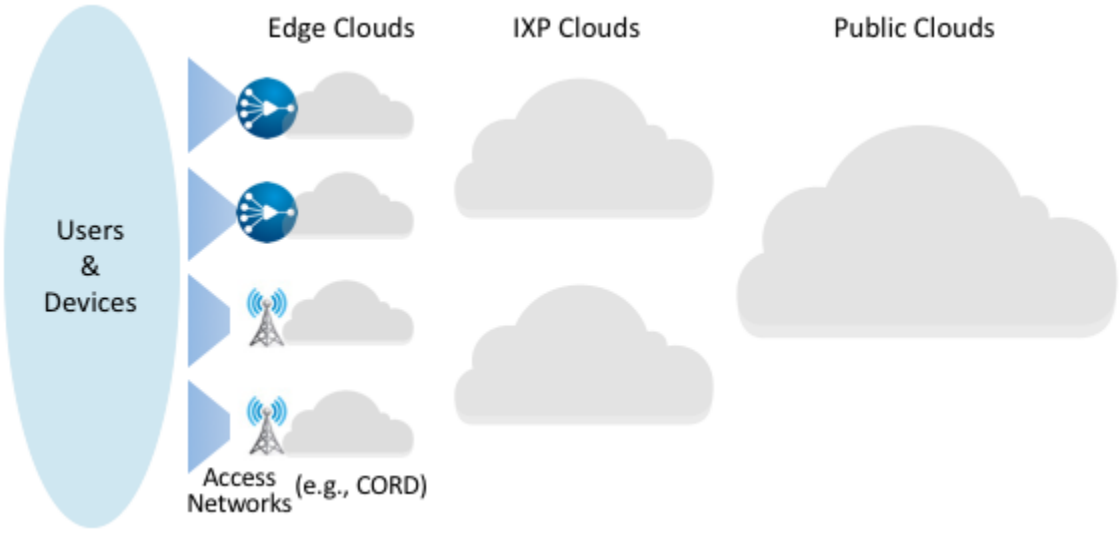

Figura 55. *A nuvem multicamadas emergente inclui nuvens públicas baseadas em datacenters, nuvens distribuídas hospedadas em IXP e nuvens de borda baseadas em acesso, como CORD. Embora existam cerca de 150 nuvens hospedadas em IXP em*

todo o mundo, podemos esperar que existam milhares ou até dezenas de milhares de nuvens de borda.

Tudo isso faz parte da tendência crescente de mover a funcionalidade para fora do data center e para mais perto da borda da rede, uma tendência que coloca provedores de nuvem e operadoras de rede em rota de colisão. Os provedores de nuvem, em busca de aplicações de baixa latência/alta largura de banda, estão migrando do data center para a borda, ao mesmo tempo em que as operadoras de rede estão adotando as melhores práticas e tecnologias da nuvem para a borda já existente e implementando a rede de acesso. É impossível prever como tudo isso se desenrolará ao longo do tempo; ambos os setores têm suas vantagens específicas.

Por um lado, os provedores de nuvem acreditam que, ao saturar áreas metropolitanas com clusters de borda e abstrair a rede de acesso, podem construir uma presença na borda com latência baixa o suficiente e largura de banda alta o suficiente para atender à próxima geração de aplicações de borda. Nesse cenário, a rede de acesso continua sendo um bitpipe burro, permitindo que os provedores de nuvem se destaquem no que fazem de melhor: executar serviços de nuvem escaláveis em hardware comum.

Por outro lado, as operadoras de rede acreditam que, ao construir a rede de acesso de próxima geração usando tecnologia de nuvem, poderão colocar aplicações de ponta na rede de acesso. Esse cenário traz vantagens inerentes: uma presença física existente e amplamente distribuída, suporte operacional existente e suporte nativo para mobilidade e serviço garantido.

Embora reconhecendo ambas as possibilidades, há um terceiro resultado que não só vale a pena considerar, mas também trabalhar para alcançá-lo: a *democratização da borda da rede.* A ideia é tornar a nuvem de borda de acesso acessível a qualquer pessoa, e não estritamente domínio de provedores de nuvem ou operadoras de rede tradicionais. Há três motivos para estar otimista quanto a essa possibilidade:

1. Hardware e software para a rede de acesso estão se tornando comoditizados e abertos. Este é um facilitador fundamental sobre o qual estávamos falando. Se ajuda as empresas de telecomunicações e de TV a cabo a serem ágeis, pode agregar o mesmo valor a qualquer pessoa.
2. Há demanda. Empresas dos setores automotivo, fabril e de armazéns desejam cada vez mais implantar redes 5G privadas para uma variedade de casos de uso de automação física (por exemplo, uma garagem onde um manobrista remoto estaciona seu carro ou uma fábrica que utiliza robôs de automação).
3. O espectro está se tornando disponível. O 5G está se abrindo para uso em um modelo sem licença ou com licenças limitadas nos EUA e na Alemanha, como dois exemplos principais, com outros países seguindo em breve. Isso significa que o 5G deve ter cerca de 100 a 200 MHz de espectro disponível para uso privado.

Em suma, a rede de acesso historicamente tem sido responsabilidade das empresas de telecomunicações, operadoras de TV a cabo e dos fornecedores que lhes vendem caixas proprietárias, mas a softwarização e a virtualização da rede de acesso abrem as portas para que qualquer pessoa (de cidades inteligentes a áreas rurais carentes, de condomínios a fábricas) estabeleça uma nuvem de ponta de acesso e a conecte à internet pública. Esperamos que isso se torne tão fácil quanto é hoje implantar um roteador Wi-Fi. Isso não apenas leva a ponta de acesso a ambientes novos (mais avançados), mas também tem o potencial de abrir a rede de acesso a desenvolvedores que instintivamente vão onde há oportunidades de inovação.

Perspectiva mais ampla

Para continuar lendo sobre a cloudificação da Internet, consulte Perspectiva: Redes virtuais até o fim .

Para saber mais sobre a transformação que está ocorrendo nas redes de acesso, recomendamos: CORD: Central Office Re-architected as a Datacenter, IEEE Communications, outubro de 2016 e Democratizing the Network Edge SIGCOMM CCR, abril de 2019 .