**Capítulo 3**

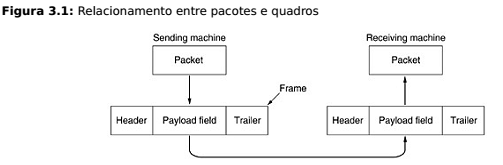
Estuda os princípios de projeto da segunda camada (enlace de dados). Este capítulo tratará de algoritmos que permitirão a comunicação eficiente e confiável entre dois computadores adjacentes (fisicamente conectadas por meio de um canal de comunicação) no nível da camada de enlace de dados. Como existem atrasos durante a propagação da informação através de um meio transmissor, os protocolos usados para comunicação foram desenvolvidos para considerar todos os fatores que influenciarão na transmissão da informação.

**3.1 – Questões de Projeto da Camada de Enlace**

Dentre as funções específicas desta camada, destacam-se as seguintes:

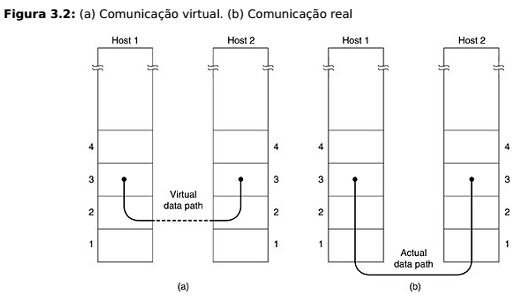
1. Fornecer uma interface de serviço bem definida à camada de rede;
2. Lidar com erros de transmissão;
3. Regular o fluxo de dados, de tal forma que receptores lentos não sejam atropelados por transmissões rápidas.

Para realizar estes serviços, a camada de enlace recebe pacotes de dados da camada de rede e os encapsula em **quadros de transmissão** (frame). Cada quadro contém um cabeçalho (header), um campo de carga útil (contém o pacote) e um final (trailer) de quadro. O gerenciamento de quadros é o núcleo das atividades desta camanda.



* + 1. **– Serviços Oferecidos à Camada de Rede**

A função da camada de enlace é oferecer serviços à camada de rede. O principal serviço é transferir dados da camada de rede da máquina de origem até a camada de rede da máquina de destino como mostra a figura 3.2(a). A transmissão segue o caminho mostrado na figura 3.2(b). No entanto é mais fácil entender os conceitos tendo como referência a imagem 3.2(a).



A camada de enlace pode ser projetada para oferecer diversos serviços, que podem variar de sistema para sistema, três possibilidades razoáveis oferecidas com frequência são:

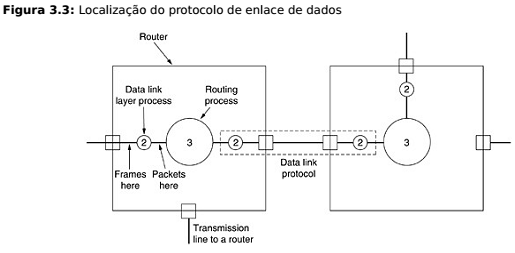
1. Serviço sem conexão e sem confirmação;
2. Serviço sem conexão e com confirmação;
3. Serviço orientado a conexão com confirmação;

**Serviço sem conexão e sem confirmação** – Consiste em fazer a máquina de origem enviar quadros independentes à máquina de destino, sem que a máquina de destino confirme o recebimento dos mesmos. Se um dado for perdido por alguma razão não haverá tentativa de detectar a perda ou de recuperá-la. Serviço apropriado quando a taxa de erro é baixa e a recuperação fica a cargo das camadas mais altas (transmissões em tempo real).

**Serviço sem conexão e com confirmação –** É o próximo passo em termo de confiabilidade. Quando este serviço é oferecido, ainda não há conexões lógicas sendo usadas, mas cada quadro enviado é individualmente confirmado. Desta forma o transmissor sabe se um quadro chegou corretamente ou não. Há o reenvio de quadros perdidos. Serviço útil em canais não confiáveis, como transmissões sem fio.

**Serviço orientado a conexão com confirmação** – É o serviço mais sofisticado que a camada de enlace fornece. Com ele, as máquinas de origem e destino estabelecem uma conexão antes dos dados serem transmitidos. Cada quadro enviado pela conexão é enumerado, e a camada de enlace de dados garante que cada quadro será de fato recebido. Além disso essa camada garante que todos os quadros serão recebidos uma única vez e na ordem correta. O serviços orientados a conexões fornecem aos processos da camada de rede o equivalente a um fluxo de bits confiáveis.

Neste serviço as transferências passam por três fases. Na primeira fase, a conexão é estabelecida. Na segunda fase os quadros são realmente transmitidos. Na terceira fase a conexão é desfeita, liberando as variáveis, os buffers e outros recursos usados na primeira fase, que tinham o objetivo de estabelecer e manter a conexão.



O protocolo de enlace de dados (data link protocol), na imagem acima, é o responsável pela confiabilidade das linhas de comunicação, tornando-as perfeitas ou, pelo menos, bastante razoáveis.

**3.1.2 – Enquadramento**

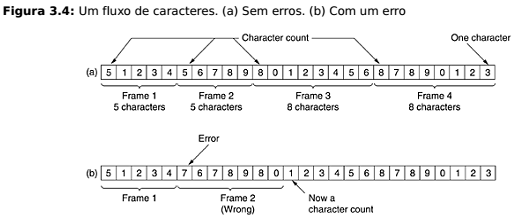
Para oferecer serviços à camada de rede, a camada de enlace deve usar serviços fornecidos pela camada física, que por sua vez tem a tarefa de aceitar um fluxo de bits brutos e tentar entrega-lo ao destino. A camada de enlace é responsável por detectar, e se possível, corrigir erros.

A estratégia é dividir o fluxo de bits em quadros (frames) e calcular o total de verificações (checksum) em relação a cada quadro. Quando um quadro chega a seu destino o total de verificação é recalculado. Caso o total recém calculado for diferente do que está contido no quadro, a camada de enlace saberá que houve um erro e tomará providências para lidar com ele.

O método de temporização pode ser utilizado para diferenciar um fluxo de bits(quadro) de outro, no entanto não é confiável. Então examinaremos quatro métodos de diferenciação de divisão do fluxo de bits em quadros.

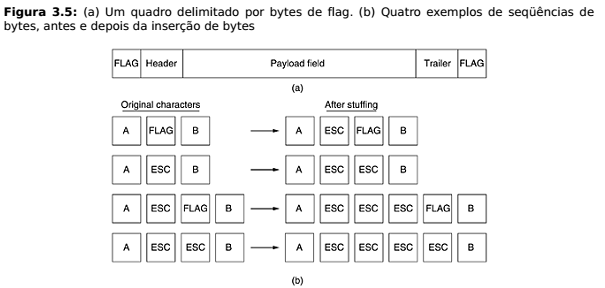
1. Contagem de caracteres;
2. Bytes de flags, com inserção de bytes;
3. Flags iniciais, com inserção de bits;
4. Violações de codificação da camada física.

O primeiro método de enquadramento utiliza um campo no cabeçalho para especificar o número de caracteres do quadro. Quando vê a contagem de caracteres, a camada de enlace de destino sabe quantos caracteres devem vir em seguida e, consequentemente, onde está o fim do quadro. Esta técnica é mostrada na figura 3.2(a) para os quadros de tamanho 5, 5, 8 e 8 caracteres respectivamente.



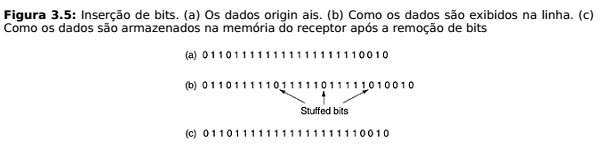
O problema com esse algoritmo é que a contagem pode ser adulterada por um erro de transmissão, tornando difícil a recuperação de quadros com erro. Logo este método não é mais usado.

O segundo método de enquadramento contorna o problema de ressincronização após um erro, fazendo cada quadro começar e terminar com bytes especiais, chamados **bytes de flag** com delimitador de início e de fim, como mostra a figura 3.5(a) na qual ele é representado por flag. Dois bytes de flag consecutivos indicam fim e início de quadro, respectivamente.



Existe um problema quando dados binários ou números de ponto flutuante são enviados, pois o padrão do byte de flag pode ocorrer nos dados, interferindo o enquadramento. Basta a camada de enlace do transmissor colocar um caractere de escape especial (ESC) imediatamente antes de cada byte de flag “acidental” nos dados. A camada de enlace do receptor remove o byte de escape antes de entregar os dados à camada de rede. Esta técnica é chamada de **inserção de bytes** ou **inserção de caracteres**. Exemplo na figura 3.5(b). Este esquema é uma simplificação do que é utilizado no protocolo PPP. Uma desvantagem deste método é que ele depende de utilização de caracteres de 8 bits, nem todos codigos de caracteres utilizam 8 bits.

Uma nova técnica surgiu para possibilitar uso de caracteres de tamanho arbitrário, ela permite que os quadros de dados contenham um número arbitrário de bits e possibilita a utilização de codigos de caracteres com um número arbitrário de bits por caracteres. De acordo c essa técnica, cada quadro começa e termina com um padrão de bits 01111110 (na verdade um byte de flag). Sempre que encontra cinco valores **1** consecutivos nos dados, a camada de enlace do transmissor insere um bit **0** no fluxo de bits que está sendo enviado. Ao ver cinco bits 1 consecutivos sendo recebido, seguido por um bit 0, o receptor remove automaticamente o bit 0. A inserção de bits , assim como a inserção de bytes, é completamente transparente para a camada de rede de ambos os computadores. A figura 3.5 mostra um exemplo de inserção de bits.



Com a inserção de bits, o limite entre dois quadros pode ser reconhecido sem qualquer tipo de ambiguidade pelo padrão de flags.

O último método de enquadramento só se aplica a redes nas quais a decodificação no meio físico contém algum tipo de redundância.Por exemplo, algumas LANs codificam 1 bit de dados utilizados utilizando 2 bits físicos. Normalmente, um bit 1 é um par alto-baixo, e um bit 0 é um par baixo-alto. O esquema significa que todo bit de dados tem uma transição intermediária, facilitando a lozalização dos limites de bits do receptor.

**3.1.3 - Controle de Erros**

Após resolver o problema de delimitação do início e do fim de cada quadro (frame), surge o problema dos erros. Como ter certeza de que todos os quadros serão entregues na camada de rede de destino, e na ordem apropriada?

A forma mais comum de garantir a entrega confiável de é dar ao transmissor um feedback sobre o que está acontecendo no outro extremo da linha. Normalmente o protocolo solicita que o receptor retorne quadros de controle especiais como confirmação positiva ou negativa sobre os quadros recebidos.

Este problema é tratado com a introdução de timers na camada de enlace. Quando o transmissor envia um quadro, em geral também inicializa um timer. O timer é ajustado para ser desativado após um intervalo suficientemente longo para o quadro chegar ao destino, ser processado e ter sua confirmação enviada de volta ao transmissor. Em geralo quadro será recebido e a confirmação voltará antes de se alacnçar o **timeout** (tempo limite) do timer e, nesse caso, o timer será cancelado.

Caso o quadro ou a confirmação for perdido o timer será desativado, alertando o transmissor. A soluçaõ deste problema seria retransmitir o quadro outra vez. Entretanto, existe a possibilidade de o receptor receber o mesmo quadro duas ou mais vezes e repassá-lo à camada de rede mais de uma vez. Para impedir que isso ocorra é necessário inserir números de sequencia aos quadros enviados, para que o receptor possa distinguir as retransmissões dos quadros originais.

**3.1.4 – Controle de Fluxo**

O controle de fluxo atua para administrar a quantidade de dados enviados e recebidos, de modo que um receptor, que lê os quadros mais devagar do que a velocidade que o emissor os envia,não perca nenhuma informação por causa de um “gargalo”. Existem duas abordagens. Na primeira, chamada de **controle de fluxo baseado em feedback**, o receptor envia de volta ao transmissor informações que permitem ao transmissor enviar mais dados, ou que pelo menos mostram a ele qual a situação real do receptor. Na segunda, chamada **controle de fluxo baseado em velocidade**, o protocolo tem um mecanismo interno que limita a velocidade com que os transmissores podem enviar os dados, sem usar o feedback, porque os esquemas baseados na velocidade nunca são utilizados na camada de enlace de dados.

**3.2 Detecção e Correção de Erros**

Os erros de transmissão ainda estarão presentes por muitos anos, as redes via cabo tem menos probabilidade de ocorrência de erros do que as redes sem fio. Como resultado dos processos físicos que os geram, os erros em alguns meios tendem a ocorrer com mais frequência em grande volume (**rajadas**) do que **isoladamente**. Suponha que um bloco de 1000 bits com uma taxa de erros de 0,001 por bit seja enviado, a taxa de erro é baixa. Porém, se os erros surgirem em rajadas de 100, apenas um ou dois blocos em 100 será(ão) efetado(s) em média. A desvantagem dos erros em rajada é que eles são muito mais difíceis de corrigir do que os erros isolados.

**3.2.1 Códigos de Correção de Erros**

Os projetistas de redes desenvolvem duas estratégis básicas para tratar erros.Uma delas é incluir informações redundantes suficientes em cada bloco de dados enviado. Com isso o receptor consegue deduzir quais devem ter sido os dados transmitidos. A outra forma é incluir uma redundância suficiente apenas para permitir que o receptor deduza se houve um erro, mas sem identificá-lo, e solicitar uma retransmissão. A pimeira estratégia utiliza códigos de correção de erro, e a outra utiliza códigos de detecção de erros.

Entendendo um erro. Um quadro consiste em **m** bits de dados e re **r** bits redundantes ou de verificação. Seja o tamanho total **n** (isto é, n = m + r). Com frequência, uma unidade de n bits que contém bits de dados e bits de verificação é chamada **palavra de codigo** (codeword) de n bits.

Dadas duas palavras de codigo, 10001001 e 10110001, é possivel determinar quantos bits correspondentes apresentam diferença. Basta efetuar uma operação OR exclusivo entre as duas palavras e contar o numero de bits 1 no resultado.



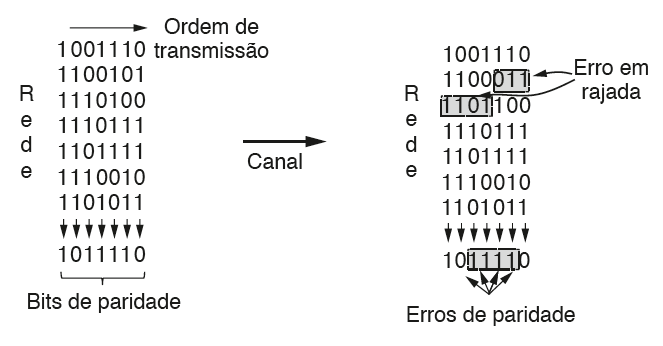
O número de posições de bits que duas palavras de codigo diferem entre si é chamado de **distância de Hamming.** Isso significa que, se duas palavras de código estiverem a uma distância de Hamming igual a **d** uma da outra, será necessário corrigir d erros de bits isolados para converter uma palavra na outra. Dado o algoritmo usado para calculos dos bits de verificação, é possivel criar uma lista com todos os códigos válidos. Assim é possível lucalizar as duas palavras de código cuja distancia de Hamming é mínima. Essa distância é a distância de Hamming do código completo.

As propriedades de detecção e correção de erros de um código dependem de sua distância de Hamming. Os códigos de Hamming so são capazes de corrigir erros simples, mas existe um artifício para torná-lo capaz de corrigir erros em rajada.

**3.2.2 – Códigos de Detecção de Erros**

Os códigos de correção de erros são extensamente utilizados em enlaces sem fio, conhecidos por serem ruidosos e propensos a erros em comparação com a fiação de cobre ou fibra óptica. É muito mais fácil lidar com erros em conexões cabeadas, pois eles ocorrem com menor frequência.

Se um único bit de paridade for inserido em um bloco e o blco for seriamente adulterado por um longo erro em rajada, a probabilidade de que o erro seja detectado é de apenas 0,5, o que não é muito aceitável. As disparidades poderão ser considerávelmente melhoradas se cada bloco for enviado com uma mstriz retangular com n bits de largura e k bits de altura. Um bit de paridade é calculado separadamente para cada coluna e afixado à matriz como sua última linha. Em seguida a matriz é transmitida uma linha de cada vez. Quando o bloco chega ao seu destino, o receptor verifica todos os bits de paridade. Se um deles estiver errado, será solicitada uma nova transmossão do bloco. Além disso, serão solicitadas retransmissões adicionais de acordo com a necessidade, até um bloco inteiro ser recebido sem quaisquer erros de paridade.



Esse método pe capaz de detectar uma única rajada de tamanho n, pois será alterado apenas 1 bit por coluna. No entanto, uma rajada de tamanho n + 1 não será detectada se o primeiro bit estiver invertido, o último bit estiver invertido e todos os outros estiverem corretos.

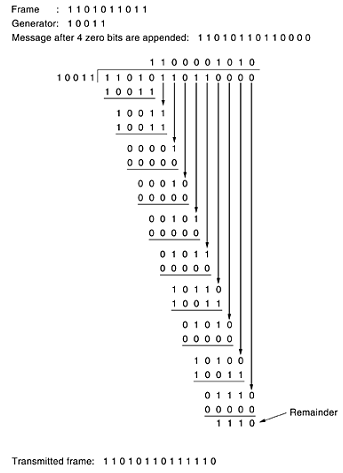
Embora este esquema se mostre adequado, na prática outro método teve seu uso mais difundido: o código polinomial, também conhecido como  **código de redundância cíclica CRC**. Esses códigos se baseiam no tratamento de strings de bits com representações de polinômios com coeficientes 0 e 1 apenas. Por exemplo, 110001 tem 6 bits, portanto represetnam um polinômio de seis termos com os coeficientes 1, 1, 0, 0, 0, 1: X5 + X4 + X0.

Quando o método de código polinomial é empregado, o transmissor e o receptor devem concordar em relação a um polinômio gerador, G(x), antecipadamente. Tanto o bit de mais alta órdem quando o de mais baixa órdem do polinômio gerador devem ser iguais a 1. Para calcular o total de verificações (checksum) de um quadro com m bits, que corresponde ao polinômio M(x), o queadro deve ter mais bits do que o polinômio gerador. A idéia é acrescentar um total de verificalaço ao final do quadro, de forma que o polinômio represetnado pelo quadro verificado pela soma seja divisível por G(x). Quando obtiver o quadro verificado, o receptor tentará dividir por G(x). A existência de um resto indica que houve um erro de transmissão.

A figura 3.8 ilustra o cálculo referente a um quadro 1101011011, usando o gerador G(x) = X4 + X + 1.

O polinômio T(x) deverá ser divisível (em 2 módulos), por G(x). Em qualquer problema de divisão, se você subtrair o resto do dividendo, o resultado será divisível pelo divisor.





**3.3 Protocolos Elementares de Enlace de Dados**

Para começar, suponha que, na camada física, na camada de enlace e na camada de rede existem processos independentes que se comunicam pelo envio de mensagens. Em muitos casos, os processis da camada física e da camada de enlace de dados estarão funcionando oem um processador dentri de um chip especial de E/S de rede, e o código da camada de rede estará na CPU principal.

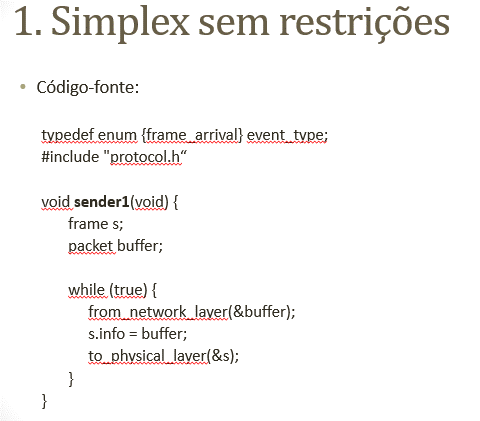
Ooutra suposição muito importante é a de que a máquina A deseja enviar um longo flluxo de dados à máquina B utilizando um serviço confiável, orientado à conexões. Supões-se que A tem um suprimento infinito de dados prontos para serem enviados, e nunca terá de esperar pela produção de dados. Quando a camada de dados de A solocita dados, a camada de rede sempre será caapaz de obedecer de imediato.

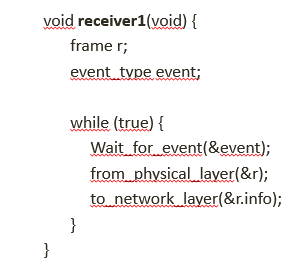
Também suponhamos que as máquinas não sofrerão panes.

**3.3.1 - Um Protocolo Simplex Sem Restrições**

Um protocolo muito simples, onde os dados são transmitidos apenas em um sentido. O protocolo consiste em dois procedimentos distintos, um que envia e outro que recebe informações. O procedimento transmissor é executado na camada de enlace da máquina de origem, e o receptor é executado na camada de enlace da máquina de destino. Não são usados números de sequência ou de confirmação. O único tipo de evento possível é o frame\_arrival (informando a chegada de um quadro).

O trasmissor é um loop While infinito que envia os dados o mais rápido possível. O corpo do while é formado por três ações: buscar um pacote, criar um quadro e transmitir o quadro ao destino. Não são usados números de sequência nem de confirmação, ou seja, não se trata o controle de fluxo e nem verifica erros.





Este protocolo oferece transmissões em um único sentido, do transmissor para o receptor. Pressupões-se que o canal de comunicação é livre de erros e que o receptor é capaz de processar toda a entrada de uma forma infinita e rápida. Consequentemente, o transmissor permanece em um loop enviando os dados com a maior rapidz possíve.

O receptor é igualmente simples. No início, ele espera que algo aconteça, e a única possibilidade é a chegada de um quadro não danificado. Eventualmente, o quadro chega e o procedimento wait\_for\_event retorna, com event definido como frame\_arrival. A chamada a from\_phisical\_layer remove o quadro recem-chegado do buffer de hardware e o coloca na variável **r**, onde o código receptor poderá buscá-lo quando necessário. Por fim, a parte referente aos dados é repassada à camada de rede, e a camada de enlace volta a esperar pelo próximo quadro, ficando efetivamente em suspenso até a chegada de outro quadro.

**3.3.2 – Um Protocolo Simplex Stop-And-Wait**

Como o protocolo anterior é pouco realista, continuamos supondo que o canal de comunicação não apresente erros e o tráfego de dados ainda é do tipo simplex. O principal problema neste caso é a forma de impedir que o transmissor inunde o receptor com dados, mais rapidamente que este é capaz de processá-lo. Além disso, se não há nenhuma atividade automática de bufferização e enfileiramento no hardware do receptor, o transmisor nunca terá de enviar um novo quadro enquanto o mais antigo não tiver sido buscado, a menos que o novo quadro substitua o antigo.

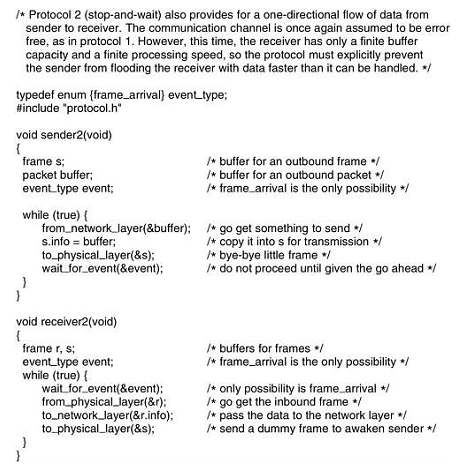
É comum que cada camada de enlace de dados tenha várias linhas para processar, e que o intervalo de tempo entre a chegada de um quadro e seu processamento varie de vorma considerável. Se puderem calcular o comportamento do receptor em uma situação totalmente desfavorável, será possível programar o transmissor para funcionar taão lentamente que, mesmo quando todos os quadros sofreram um retardo máximo, não haverá sobrecargas.

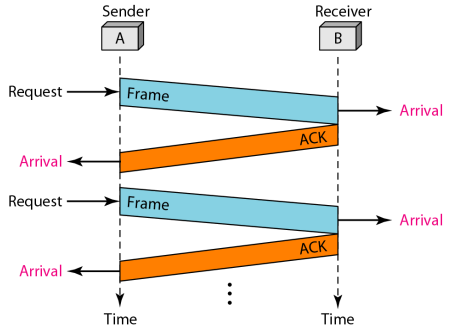
Mas esta estratégia é muito conservadora e nos leva a uma utilização de largura de banda muito abaixo do valor considerado ótimo, a menos que o comportamento do transmissor no melhor e no pior caso seja quase o mesmo.

Uma solução mais geral é fazer o receptor enviar um feedback ao transmissor. Depois de enviar um pacote à sua camada de rede, o receptor envia um pequeno quadro fictício (dummy) de volta ao transmissor, permitindo a transmissão do próximo quadro. Após o envio de um quadro, o protocolo exige que o transmissor espere a sua vez, até a chegada do pequeno quadro fictício. A utilização do feedback do receptor para informar ao transmisor quando ele pode enviar mais dados é um exemplo do controle de fluxo mencionado anteriormente.

Os protocolos nos quais o transmissor envia um quadro e em seguida espera por uma confirmação antes de continuar sua operação são chamados de stop\_and\_wait. A figura 3.11 mostra um exemplo de protocolo simplex stop\_and\_wait.







O protocolo 2 (stop\_and\_wait) também implementa um fluxo de dados unidirecional entre o transmissor e o receptor. Presume-se mais uma vez, que o canal de comunicação seja totalmente livre de erros, como no protocolo 1. O entanto, dessa vez, o receptor tem buffer finito e uma velocidade de processamento finita;portano, o protocolo deverá impedir explicitamente que o transmissor sobregarregue o receptor enviando dados mais rápidamente do que ele é capaz de processar.

**3.3.3 – Protocolo Simplex Para Canal Com Ruído**

Em um cenário em que podem ocorrer erros, os quadros podem ser danificados ou completamente perdidos. Se um quadro for danificado em trânsito o receptor detectará calculando o total de veerificação. Se o quadro for danificado a ponto de o total de ferificação nunca esteja correto o protocolo pode apresentar falhas(entregar um pacote incorreto ou duolicado na camada de rede).

A maneira mais fácil para corrigir estes problemas é fazer o transmissor inserir um número de sequência no cabeçálho de cada quaro enviado. Dessa forma, o receptor poderá verificar o número de sequência no cabeçálho de cada quadro recebido para confirmar se esse é um novo quadro ou se é uma duplicata a ser descartada.

Um número de sequência de 1 bit (0 ou 1) é suficiente para a correção do problema, uma vez que o que nos importa é saber se a entrega do quadro anterior foi realizada com sucesso.

A cada instante, o receptor espera o próximo número de sequência. Qualquer quadro recebido que contenha o número de sequência errado será rejeitado por ser considerado uma cópia. Quando um quadro com o número de sequência correto for recebido, ele será aceito e repassado à camada de rede. Em seguida, o número de sequência esperado é incrementado na base 2 (ou seja, 0 passa a ser 1, e 1 passa a ser 0). Este protocolo també transmite dados em apenas um sentido.

Ao enviar um quadro, o stransmissor ativa o timer, com um intervalo suficiente para enviar um quadro ao receptor e receber a confirmação do mesmo. Somente quando o tempo estiver se esgotado, poderemos supor com segurança que o quadro transmitido ou sua confirmação se perdeu, e que será neessário enviar uma cópia.

**3.4 – Protocolos de Janelas Deslizantes**

Nos protocolos anteriores, os quadros de dados eram transmitodos em apenas um sentido. Em situações mais práticas, há necessidade de transmitir dados em ambos os sentidos. Você pode obter uma trasmissão **full-duplex** definindo dois canais de comunicação distintos e usar cada um deles para um tráfego de dados simplex (em diferentes sentidos). Se isso for feito, haverá dois circuitos físicos separados, cada um com um canal “direto” (para dados) e um canal “inverso” (para confirmações). Em ambos os casos, a largura de banda do canal inverso é quase totalmente perdiida. Existem dois circuitos, mas apenas um deles tem sua capacidade explorada plenamente.

Uma idéia melhor é usar o mesmo circuito para dados em ambos os sentidos. Neste modelo, os quadros de dados enviados de A para B são misturados com os quadros de confirmação de A para B. Ao verificar o campo **kind** do cabeçalho de um quadro recebido, o receptor pode identificar se o quadro é de dados ou de confirmação.

O **entrelaçamento** de quadros de dados e de controle no mesmo circuito apresenta um avanço em relação ao isso de dois circuitos físicos separados, mas é possível introduzir um aperfeiçoamento. Assim que um quadro de dados chega a seu destino, em vez de enviar imediatamente um quadro de controle separado, o receptor se contém e espera até a camada de rede enviar o próximo quadro. A confirmação é acrescentada ao quadro de dados que está sendo enviado (por meio do campo **ACK** do cabeçalho do quadro). A técnica de retardar temporariamente as confirmações e enviá-las junto com o proximo quadro de dados é conhecida pelo nome de **piggingback** (superposição).

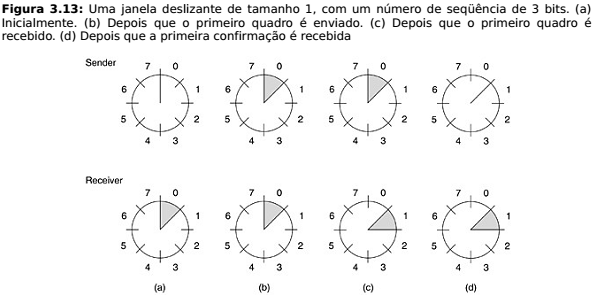
A principal vantagem do piggingback em relaçao ao envio de quadros de confirmação distintos é a melhor utilização da largura de banda disponível para o canal, pois o campo ACK do cabeçalho precisa apenas de alguns bits, e um quadro de confirmação precisaria de um cabeçalho e de um total de verificação. Também, um número menor de quadros enviados significa um número menor de interrupções. No próximo protocolo estudado, o campo de piggingback necessita apenas de um bit no cabeçalho do quadro. Em geral, ele raramente precisa de mais que alguns bits no cabeçalho.

Se ocorrer o atraso na confirmação de um pacote anteriormente enviado, a camada de enlace deve esperar um número fixo de milissegundos. Se um novo pacote chegar logo, a confirmação será acrescentada a ele; caso contrário, se nenhum pacote tiver chegado até o final desse intervalo de tempo, a camada de ençace simplesmente enviará um quadro de confirmação separado.

O protocolo de **janela deslizante stop\_and\_wait** utiliza 1 bit, restringindo os números de sequencia a 0 e 1; no entanto, versões mais sofisticadas pode, usar um valor arbitrário **n**.

A essência de todos os protocolos de janela deslizante é o fato de que, em qualquer instante, o transmissor mantém um conjunto de números de sequência correspondentes a quadros que ele pode enviar. Dizemos que estes quadros estão reunidos na janela de transmissão. Da mesma forma, o receptor mantém uma janela de recepção correspondente ao conjunto de quadros que está apto a aceitar. As janelas de do transmissor e do receptor não precisam ter o mesmo tamanho.

Os números de sequência contidos na janela do transmissor representam quadros que foram enviados ou que podem ser enviados, mas ainda não confirmados. Sempre que chega um novo pacote da camada de rede, ele recebe o próximo número de sequencia mais alto, e a borda superior da janela é incrementada em uma unidade. Quando uma confirmação é recebida, a borda inferior é incremetada em uma unidade. Dessa forma, a janela mantém continuamente uma lista de quadros não confirmados. O tamanho da janela da camada de enlace receptora corresponde aos quadros que ela é capaz de aceitar. Qualquer quadro que ficar fora da janela será descartado. A figura 3.13 mostra um exemplo.



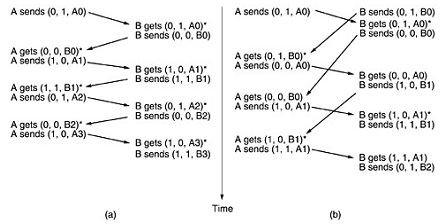
**3.4.1 – Um Protocolo de Janela Deslizante de Um Bit**

Este tipo de protocolo utiliza o stop\_and\_wait, pois o transmissor envia um quadro e espera uma confirmação antes de enviar o quadro seguinte.

Normalmente, uma das duas camadas de elnace parte primeiro e transmite o primeiro quadro. A máquina que inicia, busca um pacote na sua camada de rede, constrói um quadro a partir dele e o envia. Quando o quadro chega ao destino, a camada de enlace receptora verifica se ele é uma cópia, se o quadro for o esperado, ele será repassado à camada de rede e a janela receptora será deslocada para cima. O campo de confirmação contém o número do último quadro recebido sem erro.

Surgirá uma situação peculiar se os dois lados enviarem um pacote inicial simultaneamente. Conforme ilustrado na figura 3.15. Na parte (a), é exibida a operação normal do protocolo e na parte (b) é possível observar a peculiaridade. Se as máquinas A e B enviarem quadros simultaneamente, seus primeiros quadros se cruzarão gerando cópias de quadros , embora não haja erro de transmissão.





**3.4.2 – Um Protocolo Que Utiliza Go Back N**

Até agora estava sendo considerado que o tempo de transmissão necessário para a chegada de um quadro até o receptor somado ao tempo de transmissão para o retorno da confirmação era insignificante.

Em um cenário mais realista, o longo tempo de viagem de ida e volta pode impactar a eficienciada utilização da largura de banda.

Pag. 176