# Redes de Computadores
# Professor Marco Antônio Chaves Câmara

## Capítulo 12 - Serviços da Camada de Enlace

## Apresentação

Até agora, estudamos características de equipamentos e materiais utilizados pela camada física. Além disto, em algumas oportunidades, destacamos características de tecnologias empregadas tanto na camada física quanto na camada de enlace, como as redes ethernet, por exemplo.

Neste capítulo, nos dedicamos a estudar especificamente os serviços oferecidos pela camada de enlace, ou melhor, as técnicas de enquadramento, de detecção e tratamento de erros, além dos protocolos elementares de controle de fluxo. Alguns destes recursos são inclusive utilizados por protocolos e aplicações de outras camadas, como poderemos ver mais adiante.

## 1. O que é a camada de enlace?

A principal função da camada de enlace é fazer com que as camadas superiores sejam atendidas por serviços de comunicação compatíveis com o perfil da aplicação utilizada.

Aplicações sensíveis a erros, por exemplo, podem exigir comunicações confiáveis e com sequenciamento. Uma comunicação confiável pressupõe que não existirão erros de comunicação para as camadas superiores. Portanto, os processos de correção de erros devem ser realizados na camada de enlace, liberando as camadas superiores do tratamento destes problemas.

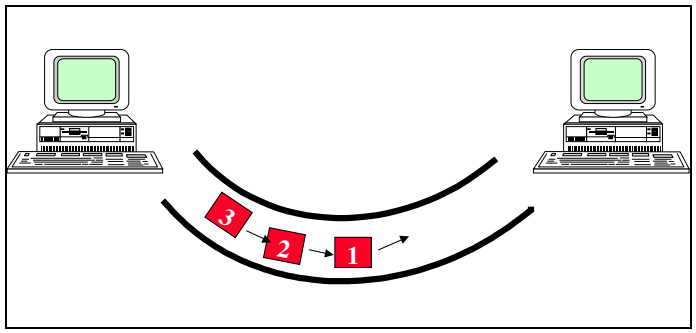



**Figura 1** - Operação do "cabo" virtual entre emissor e receptor

A correção de erros pressupõe alguma forma de realimentação do emissor pelo receptor, que precisa informar o recebimento dos dados transmitidos, além de verificar eventuais erros de transmissão. Esta situação exige a chamada "confirmação" (ou *ACKnowledge*, como é mais conhecida). Sistemas com confirmação (com ACK) conseguem garantir que as unidades de informação recebidas com problemas serão detectadas e provavelmente recuperadas.

Já uma comunicação com sequenciamento pressupõe que os dados colocados ordenadamente pelo transmissor chegarão na mesma ordem para o receptor. Sendo assim, podemos resumir o funcionamento da camada de enlace como um "cabo virtual" que interliga os equipamentos em comunicação. O conceito de "cabo virtual" significa dizer que, quando um computador **A** coloca os dados em sua camada de enlace, estes dados serão retirados imediatamente depois pelo computador **B**, na mesma ordem em que foram colocados.

Sem muita análise, poderíamos até dizer que o conceito de "cabo virtual" é óbvio. Isto porque, ao colocar os dados em uma determinada seqüência em um dos lados de uma comunicação, deveria ser possível recuperar estes dados do outro lado sem nenhum esforço! Para que então serviria a camada de enlace?

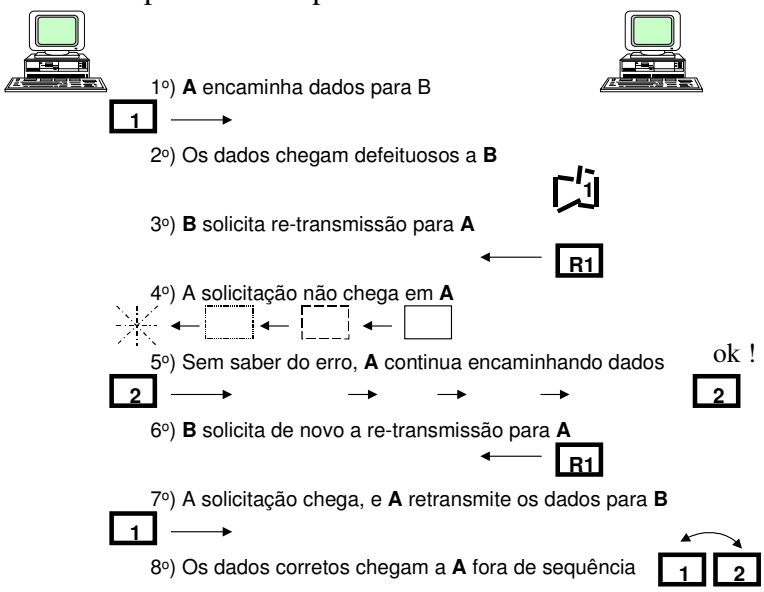



**Figura 2** - Exemplo de perda de sequenciamento

Na verdade, a existência da camada de enlace é provocada pelas falhas nos meios e dispositivos de comunicação utilizados na camada física. Estes meios e dispositivos podem, em determinadas situações, provocar erros de comunicação e atrasos de propagação. Os algoritmos utilizados para resolução destes problemas devem garantir integridade dos dados recebidos, além de seu sequenciamento. Esta atividade pode, no entanto, se tornar complicada.

Na Figura 2, demonstramos uma correção de erro que provocaria perda no sequenciamento da informação. Garantir que isto não vai ocorrer é uma das atividades da camada de enlace. Vemos também neste caso que a perda do sequenciamento ocorre em um ambiente com ACK. Em um ambiente sem ACK isto não ocorreria.

O sequenciamento pode ser garantido com a chamada "conexão", que pressupõe mais sofisticação no envio das mensagens, com a criação de contadores e rótulos para os quadros. Isto tudo sem abrir mão do ACK. Obviamente isto implica em redução da performance típica de transmissão em nome da alta confiabilidade. Alguns ambientes podem optar por um sistema mais simples, baseado apenas no mecanismo de ACK, evitando a conexão e ganhando performance. Isto pode ser feito, por exemplo, garantindo a espera pelo ACK antes do envio de um novo quadro.

Por outro lado, existem situações ainda mais críticas, onde o objetivo é não corrigir a transmissão, evitando a confirmação e, por conseqüência, esquecendo também a conexão. Em transmissões multimídia, por exemplo, muitas vezes é melhor receber um quadro com defeito do que tentar consertá-lo.

Como podemos ver, é a aplicação quem vai determinar o tipo de operação desejada para a camada de enlace. Embora a comunicação ainda não esteja ocorrendo em modo fim-a-fim (diretamente do emissor ao receptor), pode-se dizer que temos uma situação muito parecida ao controle de QoS, já que a aplicação determina o comportamento da camada.

Outras situações podem exigir o controle de fluxo, evitando que receptores lentos sejam inundados por transmissores muito rápidos. Esta é mais uma das atividades da camada de enlace. Além disto, também precisamos conhecer o processo de enquadramento, ou seja, a marcação do início e fim de cada quadro transmitido. Todas estas são funções típicas da camada de enlace. Vamos discutir separadamente cada um destes problemas neste capítulo.

## 2. Classes de serviços oferecidos à Camada de Rede

Lembrando o que estudamos no início do curso, sabemos que a comunicação entre dois equipamentos baseados no modelo OSI ocorre entre camadas adjacentes de forma virtual. Na prática, a única camada que possui interligação física entre os dois equipamentos é a camada física.

Para entendermos os serviços oferecidos pela camada de enlace, devemos considerar a comunicação virtual entre as duas camadas de rede dos equipamentos interligados. Desta forma, vamos considerar os serviços da camada de enlace como o meio de interligação entre as camadas de rede.

O modelo OSI permite que diferentes ambientes implementem diferentes serviços na camada de enlace. No entanto, poderíamos classificar os diferentes serviços oferecidos em três tipos:

### 2.1. Serviços com Conexão

A conexão é estabelecida por uma troca inicial de informações entre as duas camadas de enlace. Apenas após esta troca de informações é que os quadros são efetivamente transmitidos. Da mesma forma, ocorre nova troca de informações ao final da comunicação para garantir o fechamento da conexão.

A partir da abertura de uma conexão, a camada de enlace passa a garantir que todos os quadros encaminhados serão devidamente numerados. Isto implica na entrega dos quadros na mesma ordem em que foram encaminhados. Além disto, a numeração dos quadros impede o recebimento de múltiplos quadros iguais. Outra característica é que todos os quadros encaminhados são confirmados através do envio de quadros de

confirmação pelo receptor. Por este motivo não faz sentido imaginar um **serviço com conexão e sem confirmação**.

Desta forma a camada de enlace garante uma operação do tipo "cabo virtual" para a camada de rede. No entanto, em algumas aplicações e ambientes específicos, torna-se interessante eliminar esta garantia, que pode afetar a performance. Por este motivo, a camada de enlace pode oferecer serviços **sem conexão**, principalmente para aplicações que exijam alto desempenho.

## 2.2. Serviços sem conexão com confirmação

A confirmação garante que os quadros recebidos sem problemas pela camada de enlace serão confirmados através do envio de quadros de confirmação para o emissor. Se por acaso o emissor não receber uma destas confirmações, pode decidir por re-encaminhar o quadro correspondente, garantindo, portanto, a recepção íntegra de todos os quadros encaminhados.

Uma pergunta típica seria questionar a validade de um serviço baseado em conexões, se a confirmação já é suficiente para garantir a chegada de todos os quadros encaminhados.

Para responder isto, basta considerar um caso em que a confirmação de recebimento seja danificada ou por outro motivo qualquer não chegue ao emissor. Neste caso, o emissor poderia decidir pela re-transmissão da informação, que, no entanto, já estava disponível no receptor. A chegada deste novo quadro significaria a repetição de informações no receptor. Como os quadros não são numerados (exclusividade das conexões), o receptor ficaria impossibilitado de identificar tal repetição.

## 2.3. Serviços sem conexão nem confirmação

Este tipo de serviço oferece a menor confiabilidade possível dentre os serviços disponíveis na camada de enlace. No entanto, por outro lado, normalmente é o serviço que oferece maior performance.

Um dos pré-requisitos para escolha deste tipo de serviço é a qualidade da camada física. Se a taxa de erros observada no ambiente é bastante baixa, deixar a correção de eventuais erros para as camadas superiores pode se tornar interessante. Porém, se o meio possui taxa de erro elevada, o atraso provocado pelo tratamento dos erros em camadas superiores pode provocar problemas sérios de performance.

Resumindo, a performance é o principal fator que nos leva à escolha deste tipo de serviço na camada de enlace. Na verdade, o processo de encaminhamento de quadros de confirmação a cada quadro recebido pode provocar atrasos consideráveis na comunicação. Em aplicações que exigem

alta performance, como, por exemplo, a transmissão de voz ou vídeo, muitas vezes é mais interessante receber dados defeituosos do que atrasados.

# 3. Protocolos orientados a bits ou caracteres?

Os protocolos de enlace trabalham com base na análise de conjuntos de dados binários. Esses conjuntos podem ser analisados bit a bit, ou caractere a caractere. Essa característica diferencia o protocolo de enlace como sendo de um tipo ou de outro. Dos dois tipos, os protocolos orientados a caracteres são mais comuns.

Para maior simplicidade, no decorrer deste capítulo, sempre que for possível usar os dois tipos de protocolos, selecionaremos apenas um dos tipos no texto explicativo.

# 4. As funções da Camada de Enlace

Para oferecer as classes de serviço descritas no item anterior, a camada de enlace tem funções específicas. Essas funções têm o objetivo de garantir a transferência das informações dentro dos parâmetros definidos pela classe de serviços.

## 4.1. Enquadramento

Para implementar suas funções, a camada de enlace precisa identificar com muita clareza cada um dos conjuntos de bits (ou caracteres) que serão tratados como unidades de dados. Estes conjuntos, ou quadros, possuem campos compostos de bits com funções específicas, como endereços, códigos de controle etc.

O quadro funciona como a "unidade de informação" da camada de enlace. Para tanto, é necessário determinar suas fronteiras físicas, e garantir que emissor e receptor terão a habilidade de identificá-las na transmissão e recepção.

Antes de mais nada, portanto, é necessário garantir a identificação do início e do final de cada quadro. Para isto existem três estratégias diferentes.

### 4.1.1. Contagem de bits ou caracteres

Esta é a estratégia mais simples. O protocolo de enlace do receptor simplesmente conta o número de bits recebidos a partir do início do quadro, fechando o quadro quando o número total de bits do quadro for alcançado.

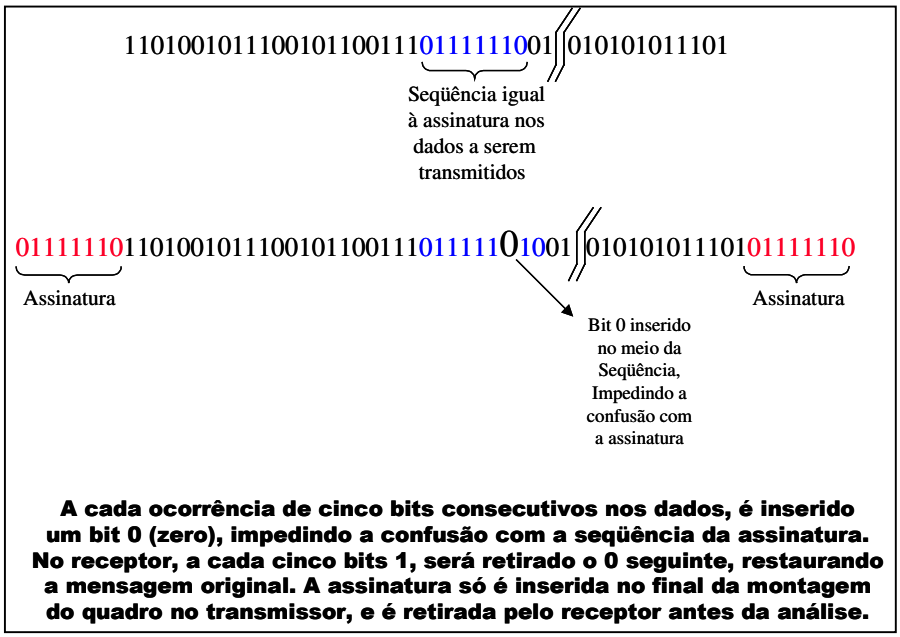
Embora seja bastante simples, o método pode falhar em caso de problemas de sincronismo, já que qualquer falha de contagem impedirá a recepção, não só do quadro onde ocorreu a falha, como também de todos os quadros posteriores.

Mesmo assim, o método de contagem é utilizado em alguns ambientes. Um exemplo notável é o protocolo *Ethernet*, que utiliza o método de contagem para identificar a fronteira final de cada quadro.

### 4.1.2. Assinatura

Este método funciona através da utilização seqüências especiais de bits ou Caracteres no início e final de cada quadro. Através da identificação destas seqüências pelo receptor, os quadros são delimitados.

Uma dúvida comum refere-se à possível confusão que ocorreria diante da presença dos caracteres de assinatura dentro dos campos normais do quadro. Este problema é resolvido através da análise, pelo protocolo do transmissor, da seqüência de dados do quadro que está sendo montado. A eventual ocorrência do(s) caractere(s) de assinatura provoca a inclusão de marcadores duplicados, que são retirados pelo receptor. Como os marcadores só ocorrem de forma duplicada nestes casos, a retirada dos mesmos pelo receptor acabará restaurando a seqüência original (ver abaixo).




O enquadramento por assinatura é bastante comum, sendo utilizado por diversos protocolos, entre eles o *frame-relay*, o HDLC, e até mesmo o *Ethernet*, que o utiliza para marcação do início de quadros.

### 4.1.3. Codificação inválida na camada física

Neste método, utiliza-se uma codificação física diferenciada para identificar os bits de início e final de cada quadro. No caso de uma codificação baseada em tensões elétricas, por exemplo, onde o bit 0 fosse normalmente identificado pela tensão de 0 volts, poderíamos utilizar uma tensão negativa (-5V, por exemplo) para identificar o bit zero, caso este estivesse localizado no início ou final de um

quadro. O mesmo processo poderia ser utilizado para diferenciar o bit 1, se este ocorresse no início ou final de um quadro.

Embora funcione adequadamente, a codificação inválida desobedece ao princípio de independência entre as camadas dos modelos de referência em camada. Isso porque o método permite que o protocolo de uma camada influencie em outra. Ou seja, a camada de enlace está, neste caso, modificando o método de codificação na camada física.

Raramente utilizado, o método de codificação inválida é utilizado, por exemplo, pelo *Token-Bus*, antigo protocolo para redes locais.

## 4.2. Controle de Erros

Tal como já vimos, os serviços da camada física envolvem a entrega de um conjunto de bits ao receptor, sem qualquer preocupação contra erros, nem mesmo quanto à possível ausência, no receptor, de alguns bits do conjunto originalmente entregue à camada física do transmissor.

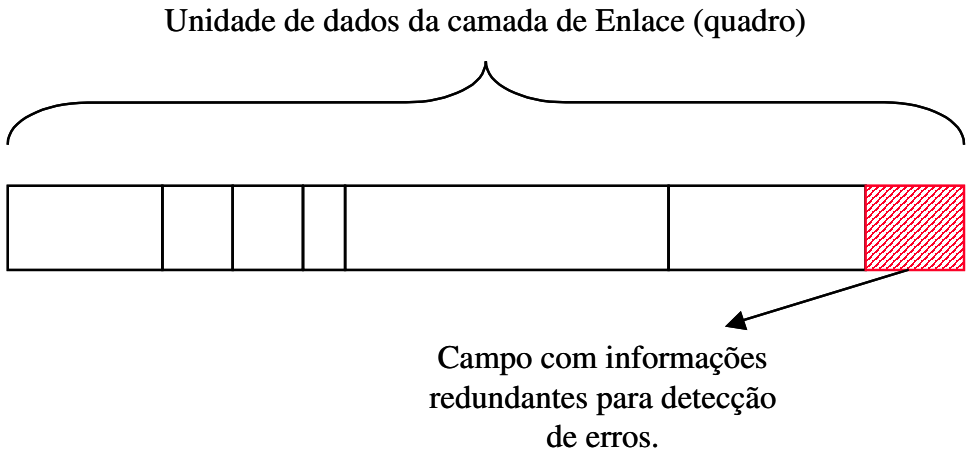



**Figura 3** - Campo com informações redundantes em um quadro típico

De fato, a camada física não possui nenhum mecanismo que permita tratar adequadamente eventuais erros de comunicação, por um motivo muito simples: não existe redundância nos dados encaminhados. Para identificar um erro em um sistema de comunicação, é fundamental um sistema de análise dos dados recebidos. Isso só pode ser implementado quando parte dos dados recebidos pode ser utilizada para conferir o conjunto completo dos dados. Na camada física, como a unidade de dados é o bit, e está a menor unidade de dados possível, não há como avaliar a correção de um determinado bit recebido.

Por este motivo, e para garantir performance, a atribuição de corrigir erros é da camada de enlace, a primeira camada de baixo para cima que pode contar com alguma redundância nos dados recebidos.

Outro aspecto importante é que a detecção e correção de erros são atividades diretamente relacionadas aos serviços com conexão ou confirmação. Nos serviços sem conexão nem confirmação, o controle de erros é desnecessário. A própria ausência do controle de erros é considerada, por alguns autores, como uma estratégia de tratamento de erros. Em outras palavras, neste caso, a estratégia de tratamento de erros é, simplesmente, não tratá-los! Isto é muito comum em ambientes com camada física de alta qualidade, e, por conseqüência, taxas de erro muito baixas.

Para os serviços que exigem controle de erros, existem duas estratégias: a re-transmissão e a correção de erros. Enquanto a primeira estratégia considera a re-transmissão dos dados pelo transmissor quando os erros forem detectados, a segunda envolve a transmissão de códigos especiais junto aos dados. Estes códigos permitem a correção dos dados eventualmente defeituosos sem a re-transmissão dos dados.

Ambas as estratégias devem lidar com problemas muitas vezes bastante complexos, como, por exemplo, a perda de conjuntos completos de dados. Nestes casos, não há como detectar a existência de erros, já que os dados desaparecem por completo no caminho entre transmissor e receptor. Cronometrar o tempo máximo para um a confirmação de recebimento pelo receptor permite descobrir casos como estes, já que o receptor não enviará confirmação alguma se por acaso não receber o conjunto de dados transmitidos. Outro problema está associado à ordenação dos quadros encaminhados, para garantir a ausência da repetição de quadros no receptor em caso de erros.

O estudo das técnicas utilizadas para detecção e correção de erros são objeto do item 5 deste capítulo.

## 4.3. Controle de Fluxo

O controle de fluxo é a habilidade do receptor de controlar a taxa de recepção de informações, de forma que seja possível o tratamento das informações recebidas, sem perdas. Isto é fundamental em ambientes heterogêneos, onde um transmissor mais rápido pode facilmente transmitir em uma taxa de transmissão superior à taxa admitida pelo receptor (computador mais rápido, máquina pouco carregada etc).

Poderíamos questionar a necessidade deste controle em ambientes homogêneos. No entanto, ambientes e aplicações heterogêneas existem em número muito maior, e além disto, o controle de fluxo permite o controle de outros problemas, como, por exemplo, as colisões em um ambiente ethernet (neste caso, o problema decorre da existência de múltiplos transmissores com dados a serem encaminhados). Por isto é extremamente importante implementar esta característica.

Normalmente o controle de fluxo já é estabelecido no momento do estabelecimento da conexão. Este define a quantidade de dados que pode ser transmitida, a necessidade do recebimento de uma confirmação antes da próxima transmissão, o que fazer caso a confirmação não chegue ou caso ocorram erros etc. Todos estes detalhes serão vistos durante o estudo dos protocolos de enlace, o que será feito no item 6 deste capítulo.

# 5. Algoritmos para tratamento de erros

Tal como vimos, estudaremos os dois principais métodos de tratamento de erros, a re-transmissão e a correção. Essa última é tipicamente interessante em canais simplex, ou em ambientes onde o atraso de propagação é muito elevado. A re-transmissão, no entanto, geralmente é mais interessante para canais *half* ou *full-duplex*, já que implica na transmissão de uma quantidade bem menor de dados.

Uma outra possível estratégia seria a simples detecção ou contagem dos erros. A partir daí, o próprio usuário poderia decidir, com base na aplicação do conjunto de dados recebidos, o que deveria ser feito. De qualquer sorte, mesmo que seja para solicitar a re-transmissão, os erros precisam ser detectados. É por isso que começamos a nossa análise com base nos protocolos de detecção de erros.

## 5.1. Detecção de Erros

Qualquer algoritmo de detecção de erros parte do princípio que nenhuma detecção de erro, por mais perfeita que seja, pode determinar, com 100% de certeza, que os dados recebidos estão corretos. No entanto, a eventual detecção de erro será sempre verdadeira, ou seja, não existe o falso-positivo, mas existe o falso-negativo.

De qualquer sorte, os algoritmos utilizados são bastante eficientes, o que, na prática, indica que a eventual ausência de erros detectados nos dá um grau de certeza bastante elevado. Erros não detectados normalmente são muito raros.

### 5.1.1. Paridade

Este é um método bastante simples. A cada conjunto de dados (tipicamente uma palavra com alguns bits de comprimento), é inserido um bit adicional chamado de bit de paridade. O método pode ser de paridade par ou ímpar, de acordo com a quantidade de bits 1 existente no conjunto de dados.

Sendo assim, o método visa garantir que a quantidade total de bits 1 (um) será par ou impar, conforme foi determinado na configuração do ambiente. Ao chegar ao receptor, basta contar a quantidade de bits 1 (um) no conjunto recebido. Se a quantidade de bits não corresponder ao sistema de paridade escolhido, certamente ocorreu algum erro de comunicação, que provocou a troca do valor de algum bit durante a transmissão.

Um dos problemas do método da paridade é a sua baixa eficiência de transmissão. Na figura abaixo podemos ver que, quando trabalhamos com um conjunto de sete bits, por exemplo, a perda de espaço de transmissão é de 12,5%, o que implica em uma redução muito grande de eficiência.

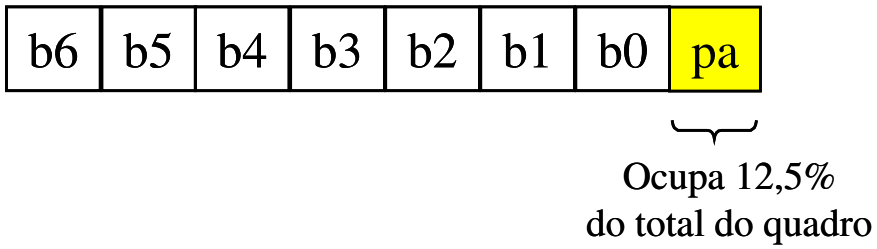



**Figura 4** - Ocupação de espaço de transmissão pelo bit de paridade

O outro problema do método é a incapacidade de detecção de falhas em um número par de bits, já que falhas deste tipo, por manterem a quantidade de bits 1, não podem ser detectadas. Por outro lado, para taxas de erros baixas, é muito improvável a ocorrência de erros em múltiplos bits.

Tal como acabamos de ver no método da paridade, um dos problemas na detecção de erros em conjuntos binários de bits está associado à quantidade de trocas de bit (erros) que ocorreram durante o processo de comunicação. Este número, também chamado de distância de *Hamming*, está diretamente associado à dificuldade na detecção de eventuais falhas.

*Richard Wesley Hamming* (1915-1998), referenciado mais tarde neste capítulo quando tratarmos dos métodos de correção de erros, desenvolveu um modelo que, comparando dois conjuntos de caracteres do mesmo tamanho, identifica o número de posições em que temos diferenças entre os caracteres dos dois conjuntos.

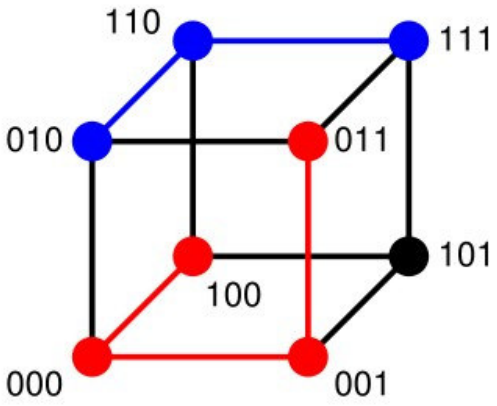



**Figura 5** - Distância de *Hamming* para conjuntos de 3 bits. Entre 010 e 111 (vermelho) temos uma distância de 2; entre 100 e 011 (azul) temos uma distância de 3.

Na prática, a distância de *Hamming* identifica a quantidade de erros de um caractere ocorridos durante a transmissão, ou mesmo o número de substituições necessárias na seqüência recebida para corrigi-la. A figura do cubo é clássica, e mostra graficamente como pode ser obtida a distância de *Hamming* entre duas seqüências de três bits.

### 5.1.2. *Check-Sum*

Uma forma de detectar erros que, de certa forma, possui uma certa independência em relação à distância de *Hamming* é o método de checagem de soma, ou, em inglês, *Check-Sum*.

Este método já é conhecido de profissionais da área financeira e contábil, que utilizam a checagem de soma para identificar erros no lançamento de dados numéricos precisos. Um contador, por exemplo, costuma somar os lançamentos de um lote de lançamentos contábeis. O resultado da soma é depois comparado com a soma informada pelo sistema de entrada de dados. Eventuais diferenças entre as duas somas identificam erros de lançamento.

No caso de dados binários, tipicamente armazenamos apenas os bits menos significativos da soma para a comparação, o que é suficientemente eficiente para reduzir a probabilidade de erros não detectados. Uma vantagem do método é a sua alta eficiência, com um pequeno consumo do espaço de transmissão com a informação redundante. Outra vantagem é a simplicidade, já que não são exigidos muitos recursos de processamento.

### 5.1.3. Código Polinomial ou CRC (*Cyclic Redundancy Code*)

O método de detecção de erro com base no CRC, ao contrário do método da checagem por soma, é baseado na divisão do conjunto de dados binários por um número binário fixo. O resto desta divisão é anexado ao final do conjunto de dados, e é denominado de CRC.

Desta forma obtemos mais eficiência na detecção de eventuais erros. Isso ocorre porque, embora o efeito da alteração de um determinado bit do conjunto de dados no quociente da divisão possa ser imperceptível, o efeito sobre o resto da divisão é sempre significativo. Sendo assim, qualquer alteração no conjunto de dados será mais facilmente detectada desta forma.

Para ser mais preciso, existe uma outra característica fundamental para entender os cálculos do CRC. Na verdade, os bits do conjunto de dados não são considerados simplesmente como bits componentes de um número binário, e sim como coeficientes de um polinômio fictício. Isto torna impossível a transposição de valores entre posições diferentes do mesmo número, criando uma aritmética diferenciada para os cálculos do CRC, que se encontra além dos objetivos deste curso.

## 5.2. Correção

Os métodos de correção permitem não só a detecção, como a identificação e correção dos erros. Estes métodos são particularmente interessantes em ambientes onde a retransmissão seria demasiadamente demorada, mas a ausência de erros é fundamental.

Entre os métodos de correção de erros, destacamos o método da paridade matricial, e o código de Hamming.

### 5.2.1. Paridade Matricial para erros em um bit isolado

O método da paridade matricial baseia-se na construção de uma matriz de dados a serem transmitidos. O conjunto de dados é distribuído, linha por linha, nas linhas da matriz, sem ocupar nem a última coluna, nem a última linha da mesma.

Na última coluna e na última linha desta matriz, são preenchidos os bits de paridade das respectivas linhas e colunas. Após isso, os dados são transmitidos coluna por coluna. No receptor, a matriz é reconstruída, e os bits de paridade são checados. Eventuais falhas, graças à característica bidimensional do modelo, poderão ser identificadas, o que leva a correção do problema.

O problema do método é que, a depender do tamanho da matriz montada, podemos ter um retardo significativo à transmissão dos dados, o que não é desejável nas redes modernas de hoje.

### 5.2.2. Código de *Hamming*

Curiosamente, o Código de *Hamming* surgiu da insatisfação de *Richard Hamming* com os erros no computador que o mesmo utilizava no seu trabalho nos Laboratórios Bell durante os anos 40. A entrada de dados era feita através de cartões perfurados, e eram muito comuns os erros na leitura dos cartões.

Durante o expediente normal, os erros eram corrigidos pela equipe de operadores do computador. Nos finais de semana, no entanto, como não existiam operadores, o computador simplesmente descartava o programa com erros, passando para a próxima tarefa. Como Richard trabalhava nos finais de semana, ele ficava irritado com a freqüência com que precisava recomeçar o seu trabalho, devido aos erros na entrada de dados. Sendo assim, decidiu trabalhar no problema da correção de erros, desenvolvendo um conjunto poderoso de algoritmos. Em 1950 ele publicou o que hoje é conhecido como Código de *Hamming*, que ainda hoje é largamente utilizado por ambientes que necessitam de correção de erros.

A teoria é simples. Se, em um determinado conjunto de bits, forem incluídos mais de um bit de checagem de erro, e esses bits forem organizados de forma a produzir diferentes identificações de erro a depender da localização do erro, deve existir uma forma de corrigir o conjunto de dados, identificando a localização do erro.

Por exemplo, em um conjunto de sete bits, existem 7 diferentes possibilidades de erro de um bit. Sendo assim, se forem inseridos 3 bits para identificação de erros dentro destes sete bits, certamente o erro poderá ser detectado e identificado, permitindo a correção da falha.

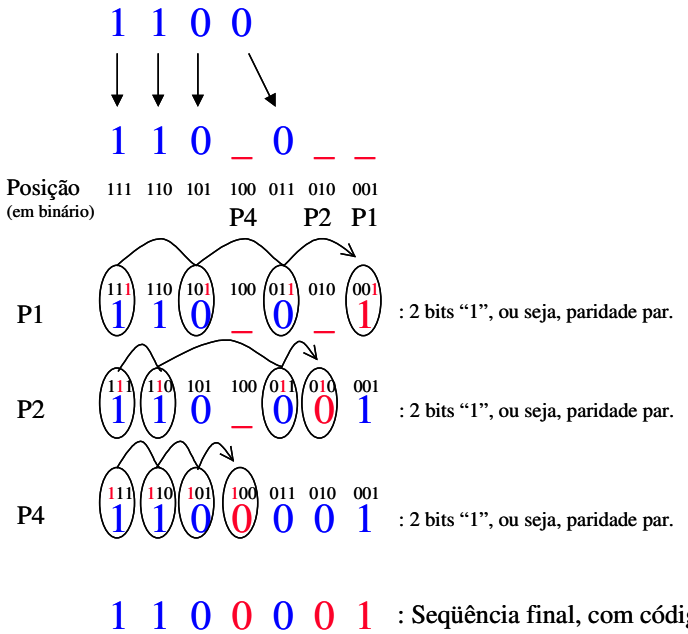
Na prática, o método permite detectar erros com distância de *Hamming* <= 2 neste conjunto de bits, além de corrigir erros que ocorram em apenas um bit.

Para demonstrar a operação do código de Hamming, vamos imaginar a seqüência de bits 1100, que precisa ser transmitida com a garantia da correção pelo código de Hamming.

O processo consiste pela criação de uma nova seqüência, onde os bits de dados (1100) serão intercalados a outros três bits de paridade. Teremos, então, sete bits, cujas posições serão numeradas de 1 a 7, em binário, da direita para a esquerda. Os três bits de paridade serão inseridos nas posições 1 ($001_2$), 2 ($010_2$)e 4 ($100_2$), que correspondem às três primeiras potências de dois.

Para obter os três bits de paridade, consideraremos diferentes conjuntos de bits para cada um deles. Identificando cada um dos bits pela sua posição binária, basta considerar:

- Para o primeiro bit de paridade (na posição $001_2$), devemos considerar os bits nas posições $001_2$, $011_2$, $101_2$ e $111_2$.
- Para o segundo bit de paridade (na posição 0102), devemos considerar os bits nas posições $010_2$, $011_2$, $110_2$ e $111_2$.
- Para o terceiro bit de paridade (na posição $100_2$), devemos considerar os bits nas posições $100_2$, $101_2$, $110_2$ e $111_2$.




Para checar a operação do código de Hamming, podemos simular a ocorrência de erro em um dos bits.

No primeiro exemplo, simulamos um erro no bit da posição 2 durante a transmissão. Vejam que o bit de paridade 1 está correto, o que garante que os bits 1, 3, 5 e 7 estão corretos (em amarelo). Da mesma forma, o bit de paridade 4 está correto, garantindo que os bits 4 e 6 também estão corretos (em azul). Vejam que apenas o bit 1 não ficou marcado, ou seja, é ele quem está errado!

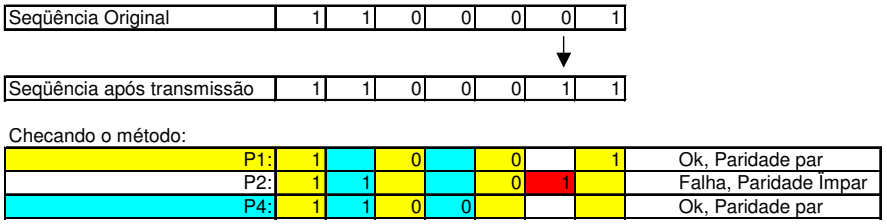



**Figura 6** - Tratamento de Erro no bit 2, usando o Código de Hamming

No segundo exemplo, simulamos um erro no bit da posição 5 durante a transmissão. Vejam que o bit de paridade 2 é o único correto. Com isso, o erro

poderia estar nos bits 1, 4 e 5. Como apenas o bit 5 está presente simultaneamente na análise das paridades 1 e 4, ele é o bit incorreto.

| Seqüência Original | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| ↓ | | | | | | | |
| Seqüência após transmissão | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

Checando o método:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| P1: | 1 | | 1 | | 0 | | 1 | Falha, Paridade Ímpar |
| P2: | 1 | 1 | | | 0 | 0 | | Ok, Paridade par |
| P4: | 1 | 1 | 1 | 0 | | | | Falha, Paridade Ímpar |

**Figura 7** - Tratamento de Erro no bit 5, usando o Código de Hamming

# 6. Controle de Fluxo na Camada de Enlace

Vamos começar o nosso estudo conhecendo alguns protocolos da camada de enlace, começando pelos mais simples, e evoluindo na busca da solução dos problemas encontrados.

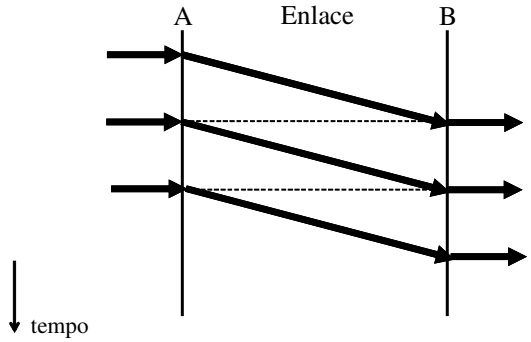
## 6.1. Protocolos Elementares

### 6.1.1. Protocolo Simplex sem restrições

Este é o protocolo mais simples. Ele considera um canal de comunicação perfeito, ou pelo menos próximo disto. Tal como vimos em capítulos anteriores, isto envolve taxas de erro de bit (BER) baixas, tais como aquelas encontradas nos ambientes de rede local típicos hoje em dia.

Como a taxa de erro é muito baixa, a possibilidade de erros é simplesmente desprezada pela camada de enlace, que considera que os eventuais erros serão corrigidos pelas camadas superiores. Isto implicará em perda de performance na correção de erros. No entanto, se os erros são incomuns, não há perda de performance, e sim ganho, pois uma eventual checagem de erros muito improváveis acabaria por implicar em perda de performance ainda mais significativa durante a transmissão.

Desprezar os erros significa abdicar da confirmação de recebimento pelo receptor, ou seja, temos transmissão de dados apenas em um sentido. Veja o diagrama abaixo :



Para aqueles que nunca viram um desenho similar, este é o diagrama típico para representação do controle de fluxo em protocolos. Duas linhas verticais identificam o emissor (A) e o receptor (B). À medida em que descemos no gráfico, temos o decurso do tempo (veja seta do lado esquerdo inferior do desenho). Quando uma unidade de informação (quadro) sai do emissor (A) para o receptor (B), temos um tempo de transmissão, que é representado pelo segmento de reta inclinado.

Entendido o diagrama, vemos que no Protocolo Simplex os dados são encaminhados quadro após quadro em seqüência, sem aguardar qualquer comunicação em sentido contrário. O emissor apenas aguarda o final da transmissão de um quadro para enviar o próximo imediatamente depois. Sendo assim, temos que considerar que o emissor possui uma infinidade de dados a transmitir, e o receptor está sempre pronto para receber e analisar os dados recebidos, sem nenhum retardo.

No entanto, no mundo real, tanto emissor quanto receptor podem ficar momentaneamente ocupados, não tendo a disponibilidade para liberar quadros novos para transmissão, ou para analisar imediatamente um novo quadro recebido. Nestes casos, tornam-se necessárias estruturas de armazenamento temporário chamadas de *buffers*, que garantem o armazenamento temporário de informações que serão transmitidas, ou que foram recebidas e não foram processadas. Isto permitiria a alimentação contínua dos processos de transmissão e recepção.

Para isto, no entanto, a capacidade de produzir quadros para transmissão deve ser maior do que a taxa de entrega à camada de enlace, ou não haveriam quadros disponíveis quando a interrupção ocorresse. No caso do receptor, a cada interrupção de processamento, seria necessário armazenar os dados em um buffer de recepção. Isto implica em *buffers* infinitos, o que é obviamente impossível. Na prática, teríamos de tempos em tempos a perda de dados por estouro da capacidade de memória, seja no emissor, seja no receptor.

Por outro lado, em termos de eficiência de transmissão, o Protocolo Simplex é muito interessante. Se observarmos o diagrama, vemos que o meio físico encontra-se permanentemente ocupado com a transmissão de dados. Ou seja, temos uma ocupação de 100% do meio físico, com o uso de toda a sua capacidade para a atividade fim, ou seja, transmitir dados.

### 6.1.2. Protocolo Simplex Pare-e-Espere (Stop-and-wait)

Os buffers não são infinitos

O tempo de processamento não é ignorado

O transmissor não envia outra mensagem até que a anterior tenha sido aceita como correta pelo receptor

Embora o tráfego de dados seja simplex, há fluxo de quadros em ambos os sentidos.

### 6.1.3. Protocolo Simplex para um canal com ruído

Os quadros são numerados seqüencialmente

O tx transmite um quadro

O rx envia uma quadro de reconhecimento se o quadro for recebido corretamente, caso contrário, há um descarte e é aguardada uma retransmissão

Quadros não reconhecidos são retransmitidos devido à temporização estabelecida por um relógio interno.

## 6.2. Estratégias para Aumento de Performance dos Protocolos de Enlace

Considerando-se a existência de tráfego de confirmação em sentido contrário, podemos utilizar algumas técnicas especiais para aumentar a eficiência dos protocolos. Tais técnicas podem ser utilizadas isoladamente ou em conjunto. Apresentamos abaixo duas das mais conhecidas, o *Piggybacking* e o *Pipeline*.

### 6.2.1. Piggybacking

Em todos os protocolos básicos analisados, os dados são transmitidos em apenas um sentido. O canal reverso é usado apenas para ACK ou NACK. No entanto, tipicamente, as comunicações exigem transferência nos dois sentidos. Para manter o esquema atual, seriam necessários dois canais duplos, onde o canal reverso de cada um dos dois seria muito mal aproveitado. Isto porque o tempo gasto para transmissão dos quadros de ACK ou NACK ocupa um percentual pouco significativo da capacidade do canal, e ainda implica em perdas de eficiência na transmissão, já que o emissor precisa ficar aguardando a confirmação antes de enviar o próximo quadro.

Uma primeira idéia seria reter o ACK ou NACK até o envio do próximo quadro no sentido reverso. Quando isto acontecesse, um dos campos do quadro seria utilizado para carregar o ACK ou NACK. É como se os quadros em sentido inverso oferecessem uma "carona" para o ACK ou NACK. O termo utilizado para este tipo de operação é *piggybacking*.

Além da vantagem da redução da ocupação do canal com as confirmações, seus cabeçalhos e rodapés, temos também uma redução no número de quadros encaminhados, e com isto, uma quantidade menor de interrupções para tratamento de quadros, e *buffers* de recepção menores.

O único problema que pode ocorrer é a inexistência temporária de quadros em sentido contrário. Neste caso, torna-se necessária a devolução forçada do ACK ou NACK em um quadro específico, tal como se não houvesse *piggybacking*. Isto é implementado através do uso de um temporizador que determine o tempo máximo de espera (*time-out*) por um quadro em sentido contrário.

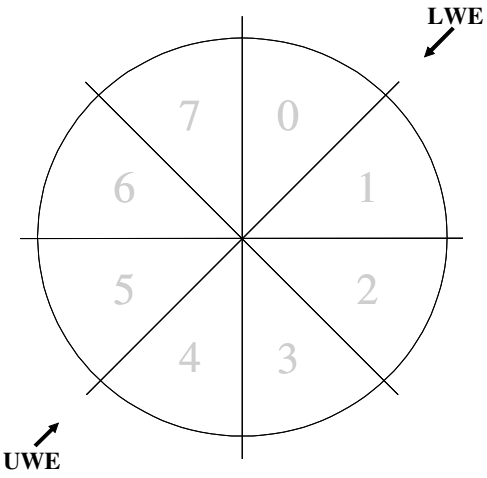
### 6.2.2. Pipeline

Uma outra opção que permite aumentar a eficiência do canal em caso de redução da taxa de erro é o envio de quadros maiores, ou o envio de múltiplos quadros em seqüência, com confirmações em bloco (ACK ou NACK múltiplo). Salvo no caso de erro de comunicação, esta estratégia costuma implicar em aumento significativo de performance, já que se perde menos tempo de transmissão com ACKs e NACKs.

A ocorrência de erros de comunicação em taxas mais elevadas pode invalidar o *pipeline*, já que torna-se necessária a re-envio de múltiplos quadros, mesmo que o erro tenha envolvido apenas um deles.

## 6.3. Protocolos de Janela Deslizante (*Sliding Windows*)

Continuando o trabalho de melhoria da eficiência e performance dos protocolos de enlace, precisamos conhecer os protocolos de "janela deslizante", ou *Sliding Windows*. Sua operação baseia-se no estabelecimento de "janelas de transmissão e recepção" com um determinado número de quadros a serem transmitidos no emissor e/ou recebidos no receptor. As janelas são numeradas, permitindo o acompanhamento dos ACK e NACKS com muito mais eficiência. Vejamos o desenho abaixo :




O desenho representa a área de *buffers* de transmissão ou recepção em um ambiente baseado no protocolo *Sliding Windows*.

Como podemos ver, a área é representada na forma circular, demonstrando a reutilização cíclica dos *buffers*. Neste desenho, estamos considerando uma janela de tamanho 8, o que permite a identificação de cada um dos *buffers* por um número de 3 bits binários. Tais números são marcados não só nos *buffers* como também em cada um dos quadros, ACKs e NACKs encaminhados. Isto simplifica a identificação dos mesmos na eventualidade de recuperação de falhas.

Cabe ao protocolo estabelecer o tamanho da janela, que normalmente tem relação com o RTT (*Round Trip Time*), ou tempo de ida-e-volta. Este é o mesmo conceito que vimos quando estudamos o protocolo de transmissão *Ethernet*, neste caso identificando o tempo máximo para transmissão de um quadro, recepção pelo receptor e retorno de um ACK ou NACK. Quanto maior o RTT, maior pode ser o tamanho da janela.

### 6.3.1. Operação do Protocolo

Utilizando um esquema de transmissão que lembra o esquema *Pipeline*, o emissor encaminha os quadros em seqüência, identificando-os com os números entre 0 e 7. Cada quadro, ao ser recebido corretamente pelo receptor, provoca o envio, pelo mesmo, de um quadro de ACK contendo o mesmo número. Assim como no *pipeline*, o emissor não aguarda o recebimento do ACK correspondente para o

envio do próximo quadro. Este continua encaminhando os próximos quadros, e prossegue nesta atividade até que a janela seja preenchida, mesmo que não ocorra nenhuma confirmação de recebimento até então.

Para controlar o número de quadros enviados sem confirmação, o emissor identifica duas posições na janela de transmissão utilizando os ponteiros LWE (*Lower Windows Edge*) e UWE (*Upper Windows Edge*). O LWE marca a fronteira inferior da janela (primeiro quadro transmitido ainda sem confirmação) e o UWE a fronteira superior (último quadro sem confirmação). A quantidade de quadros sem confirmação pode ser obtida analisando-se os dois valores. Ao receber os ACKs dos quadros transmitidos, o valor de LWE vai sendo incrementado, permitindo a transmissão de novos quadros, o que, por sua vez, incrementará também o valor de UWE. Na figura anterior, considerando que temos a janela do emissor, temos quatro quadros transmitidos e identificados de 1 a 4. Se o processo continuar até o preenchimento de todas as posições do *buffer*, o transmissor interrompe a transmissão e fica aguardando o recebimento do ACK do quadro identificado pelo ponteiro LWE.

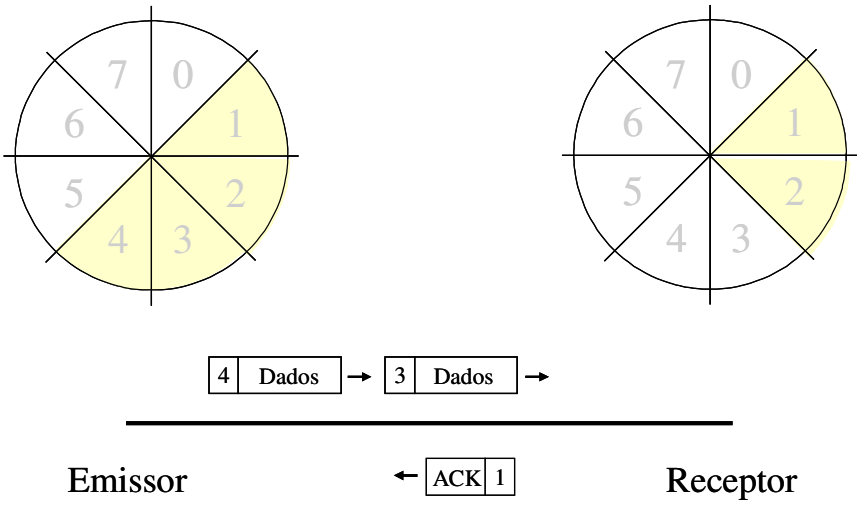
Já no receptor, o LWE e o UWE delimitam a faixa contendo os últimos quadros recebidos. A cada quadro recebido, o receptor encaminha um ACK ou NACK para o emissor confirmando a correção ou falha nos dados. Como vimos, o RTT foi ajustado para garantir a recepção, pelo emissor, do ACK ou NACK de um quadro antes da envio antecipado de toda a janela.

Em determinadas situações, pode ocorrer a recepção de uma nova cópia de um quadro já recebido. Neste caso, pode-se concluir que ocorreu falha na recepção do ACK daquele quadro pelo emissor, que decidiu então pelo re-envio do quadro. Isso provocaria o descarte do mesmo e o envio de um novo ACK.

Para determinadas situações, nem mesmo eventuais quadros recebidos com defeito interrompem o processo. Após o envio do NACK numerado, o receptor prossegue na recepção de novos quadros enquanto aguarda o re-envio daquele quadro, que será armazenado, ao ser recebido, na posição correta da janela. Sistemas que adotam esta estratégia são chamados de *Selective Repeat* (ver item 6.3.3).

Os protocolos *Sliding Windows* podem operar inclusive como as versões mais simples. Como o tamanho da janela pode mudar, se o número for 1, temos um protocolo similar ao *stop-and-wait*. Isto, porém, é muito ineficaz em canais com canais com alto RTT.

A figura a seguir representa um processo típico de transmissão envolvendo emissor e receptor :




Neste caso, a janela do receptor ainda não está totalmente preenchida, o que é normal em um processo de transmissão típico. Observe que, na figura, estão em trânsito no meio físico dois quadros de transmissão (3 e 4) e um quadro de ACK (ACK 1). Ao chegar no receptor, os dois quadros vão provocar o preenchimento das posições 3 e 4 da janela do receptor. Já no transmissor, ao receber o quadro ACK 1, o LWE será incrementado, permitindo a transmissão do quadro 5.

Após entender a operação do protocolo, o problema que ainda precisa ser tratado é a correção de eventuais erros de transmissão. Isto porque teremos diversos quadros armazenados nas memórias do transmissor e do receptor, e estes precisam ser devidamente tratados. As duas estratégias mais conhecidas são a "*Go-Back-N*" e a "*Selective Repeat*".

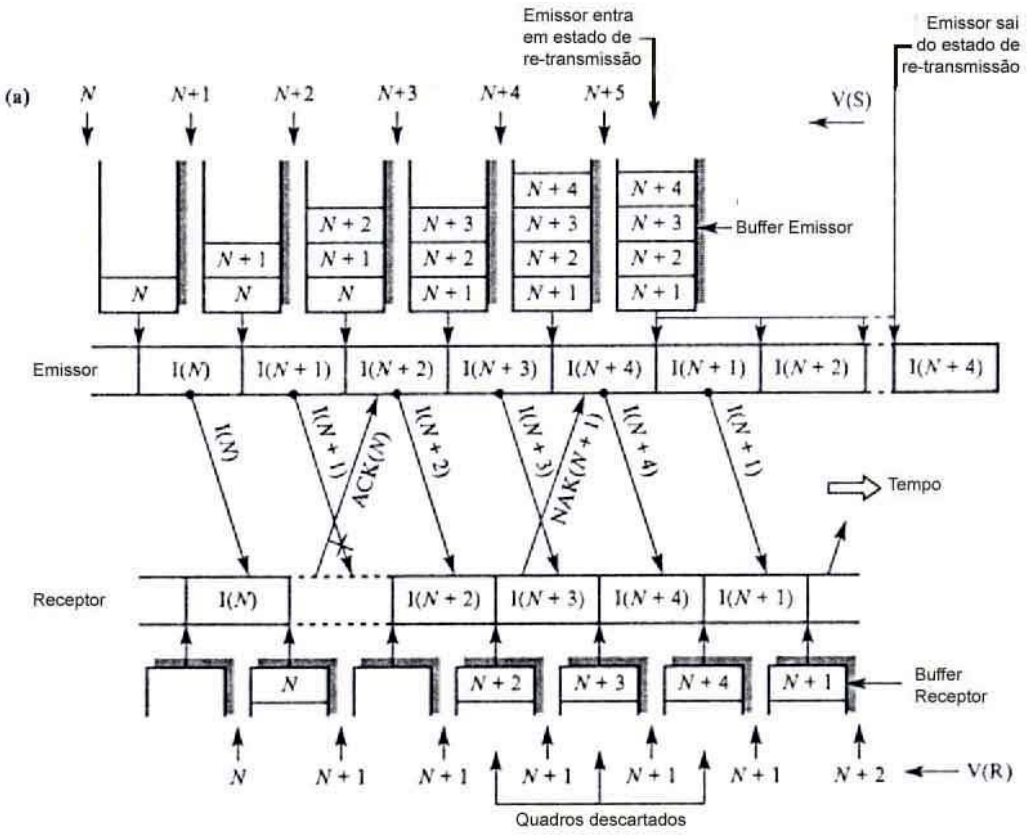
### 6.3.2. Estratégia *Go-Back-N*

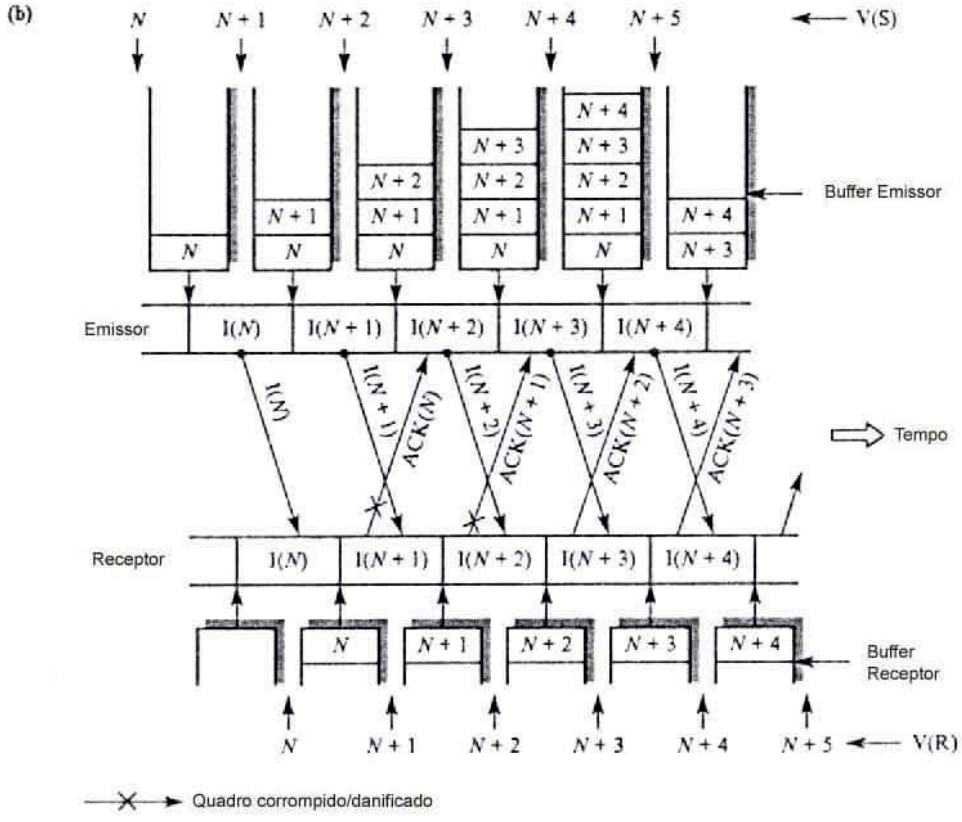
Nesta estratégia, todos os quadros presentes na memória do transmissor a partir do quadro defeituoso serão re-transmitidos. Daí o nome da estratégia, que indica o retorno ao quadro defeituoso para re-transmissão de todos os quadros.

Eventuais quadros posteriores, mesmo que já tenham sido confirmados, serão apagados.

Apesar da aparente perda de eficiência, este método pode ser bastante interessante em ambientes onde a simplicidade é característica crítica para a operação do ambiente. Afinal de contas, re-transmitir um grupo de quadros já armazenados no buffer de transmissão não é uma tarefa muito complexa. Por outro lado, devido à re-transmissão dos quadros, aumenta a utilização da banda de passagem.

O gráfico a seguir demonstra a operação do protocolo em duas diferentes situações : o erro na transmissão de um quadro e o erro na transmissão de um ACK.


(a)
Emissor entra em estado de re-transmissão
Emissor sai do estado de re-transmissão
V(S)
Buffer Emissor
Emissor
Tempo
Receptor
Buffer Receptor
Quadros descartados
V(R)


(b)
V(S)
Buffer Emissor
Emissor
Tempo
Receptor
Buffer Receptor
V(R)
Quadro corrompido/danificado

### 6.3.3. Estratégia *Selective Repeat*

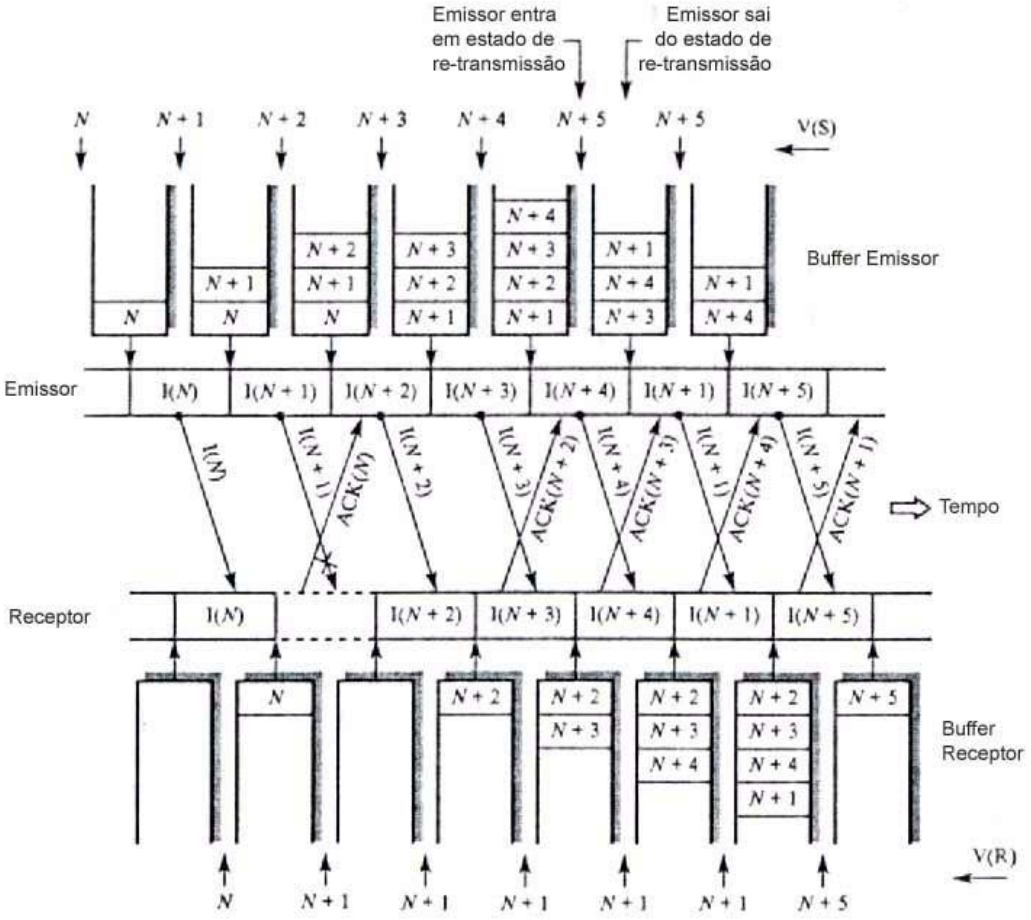
Nesta estratégia, apenas o quadro defeituoso é re-transmitido, permitindo o aproveitamento de eventuais quadros posteriores que já tenham sido confirmados.

Apesar do ganho de eficiência na utilização da banda de passagem disponível, esta estratégia traz consigo alguns problemas associados à sua complexidade. Os quadros muitas vezes precisam ser remanejados para garantir a ordenação. Além disto, aumenta a necessidade de manutenção dos *buffers* de recepção, já que os quadros precisam estar disponíveis até a efetiva entrega para as camadas superiores, na ordem correta.

Existem duas formas do emissor decidir pela re-transmissão do quadro defeituoso ou ausente. Na primeira, o emissor determina a ausência do quadro N pela não recepção do ACK-N. Esta é a chamada re-transmissão implícita. Na segunda, o receptor encaminha um NACK específico ao detectar a ausência de um quadro específico. Vamos analisar a re-transmissão implícita no gráfico a seguir.

Observe que o *buffer* do emissor, tal como vimos anteriormente, vai armazenando todos os quadros enviados, descartando o primeiro *buffer* (N) apenas ao receber o ACK correspondente. Ao perceber que o ACK do quadro (N+2) chegou, mas o ACK do quadro (N+1) não chegou, o emissor entra automaticamente em "modo de re-transmissão", interrompendo a transmissão de novos quadros, e re-enviando o quadro N+1. De qualquer forma, o emissor descarta o *buffer* N+2 (confirmado). Logo após o envio, o emissor sai do modo de re-transmissão, e descarta o *buffer* N+3.

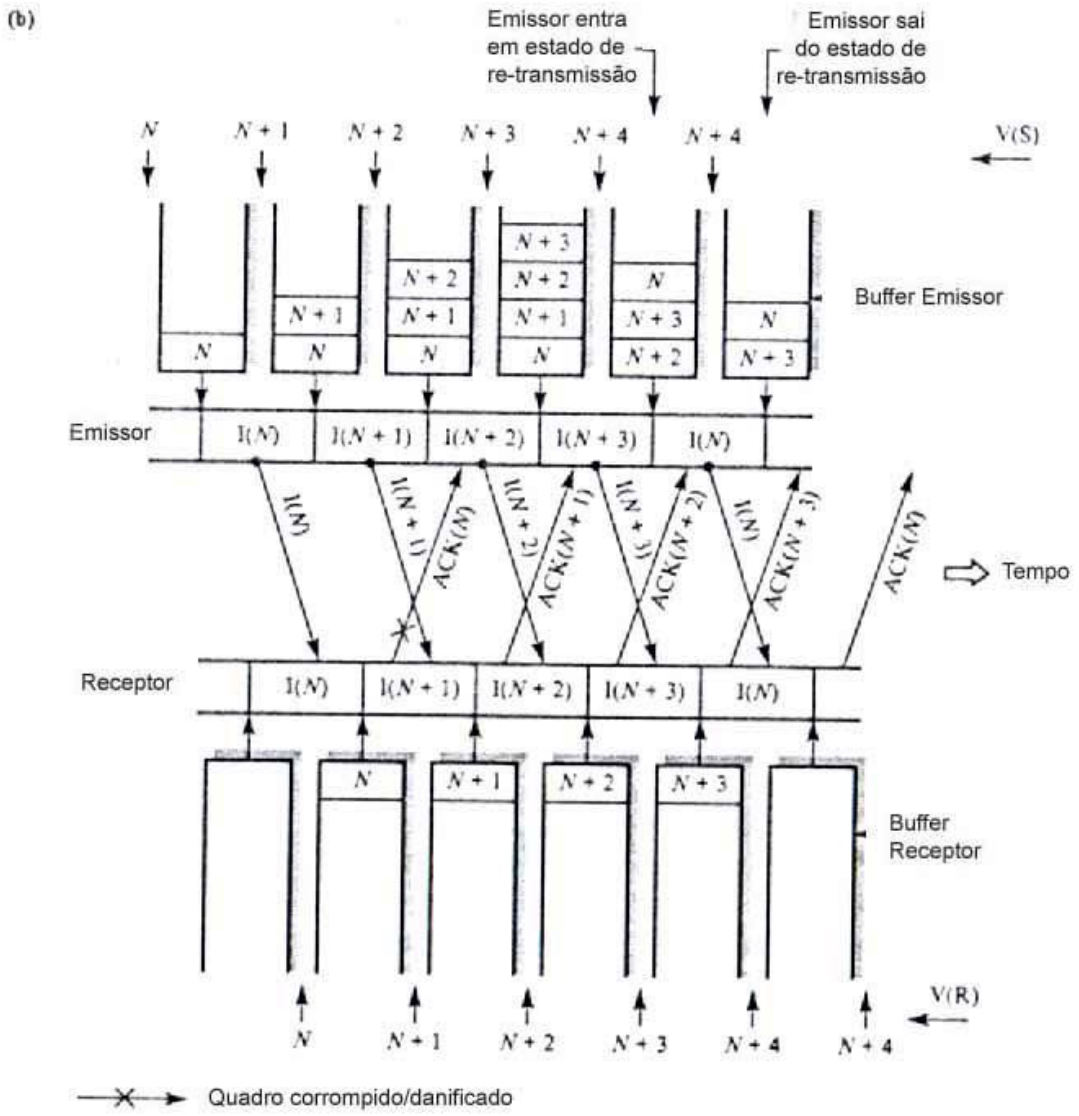
No receptor, podemos perceber que o *buffer* armazena os quadros recebidos após N+1 (que não foi recebido adequadamente) até receber a re-transmissão do mesmo, quando encaminha todos, na ordem correta, para a camada superior. Veja que no final, apenas o quadro N+5 fica armazenado no *buffer*.




Toda a nossa análise baseou-se em uma situação de erro na transmissão de um quadro. Veremos no gráfico a seguir o que acontece quando ocorre um erro no envio de um ACK.

Apesar de ter recebido corretamente o quadro N, houve falha no envio do ACK-N. Ao receber o ACK (N+1), o transmissor pressupõe a perda do quadro N, e o re-transmite assim que termina a transmissão do quadro N+3.

Ao receber uma segunda cópia do quadro N, o receptor descarta o mesmo, mas re-envia o ACK-N, garantindo a eliminação do mesmo no *buffer* do transmissor.




## 6.4. Protocolos do Mercado

O tipo de protocolo de enlace utilizado na prática tem normalmente relação direta com o tipo de interligação física existente entre emissor e receptor. Alguns aspectos importantes são a taxa de transferência nominal e também a distância entre os equipamentos, devido às questões relacionadas ao retardo.

Para sistemas com taxas de transferência reduzida, como aqueles utilizados em interligações com *modems*, tipicamente utiliza-se um protocolo simplex *stop-and-wait* orientado a caracteres, como o KERMIT ou o X-MODEM.

Sistemas com taxa de transferência mais elevada, especialmente aqueles que envolvem grandes separações físicas (e, portanto, retardos

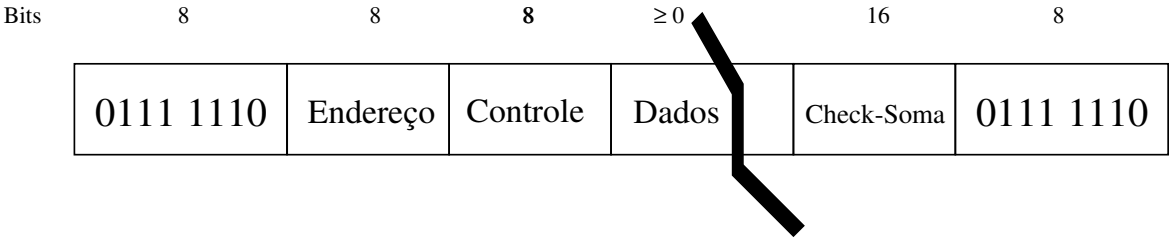
significativos), utilizam tipicamente protocolos mais complexos e eficientes, como o HDLC (*High Level Data Link Layer*) , por exemplo.

### 6.4.1. HDLC (*High level Data Link Control*)

Existe um grupo de protocolos derivados do SDLC (*Synchronous Data Link Control*), um dos protocolos da suíte SNA da IBM. Com a submissão, pela IBM, do SDLC para aprovação pela ANSI e pela ISO, surgiram o ADCCP (*Advanced Data Communication Control Procedure*) da ANSI e o HDLC da ISO.

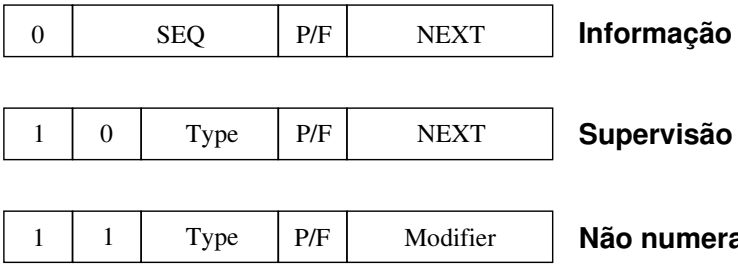
O HDLC foi adotado e modificado pelo CCITT, que o adotou com o nome LAP (*Link Access Procedure*), e este passou a fazer parte do protocolo X-25. O mesmo foi modificado posteriormente, sendo chamado de LAPB (*Link Access Procedure, balanced*), o que o tornou mais compatível com uma versão posterior do HDLC. Todos estes protocolos são muito semelhantes, embora possuam pequenas diferenças.

O HDLC é orientado a bits, utilizando o método de enquadramento por seqüência de bits visto no item 4.1.2 desta apostila. O formato do quadro é o seguinte:




O campo de endereço é importante em ambientes com múltiplos terminais, permitindo a identificação do terminal de destino. Em ambientes ponto-a-ponto, onde o mesmo seria aparentemente desnecessário, ele é utilizado algumas vezes para permitir a identificação entre comandos e respostas.

O campo de controle é usado para os números de seqüência e ACKs. No entanto, o mesmo tem aplicação específica em função do tipo de quadro HDLC com que estamos lidando. Antes de qualquer coisa, é importante saber que o HDLC utiliza um protocolo de janela deslizante com 3 bits para identificação da seqüência. Os quadros do tipo "Informação" carregam a identificação do número de seqüência do quadro (até sete quadros podem ficar "pendentes") no sub-campo SEQ (ver figura abaixo). O sub-campo NEXT é usado para o ACK feito através de *piggybacking*. Este ACK é feito através do envio do número de seqüência do próximo quadro ainda não recebido. O bit P/F é a sigla *Poll/Final*. Ele indica o interesse de um dos lados da comunicação em receber dados (*Poll*). Também em comunicações que envolvem diversos quadros consecutivos, todos os quadros, menos o último, são marcados como (*Poll*), identificando o final da seqüência. Em algumas variações do protocolo, o bit P/F é utilizado para forçar o envio de um quadro de "Supervisão", ao invés de esperar um pacote de retorno para fazer o *piggybacking*.




Os quadros de supervisão são classificados de acordo com o conteúdo do campo "Type". O tipo "0", por exemplo, é utilizado como ACK diante da inexistência de tráfego reverso para *piggybacking*. O quadro "1" é o NACK, com o campo NEXT indicando o quadro não recebido corretamente. A partir deste erro, é solicitado o envio de TODOS os quadros subseqüentes (estratégia "*go-back-N*"). O tipo "2" é utilizado para controle de fluxo, impedindo o envio de novos quadros (*Receiver Not Ready*).O tipo 3 implementa a estratégia "*Selective Repeat*", mas só implementado no HDLC e no ADCCP. O SDLC e o LAPB não implementam esta funcionalidade.

Os quadros do tipo "Não numerados" são usados para comunicações não orientadas à conexão. Neste tipo de quadro existem grandes diferenças entre as implementações do SDLC, ADCCP, HDLC e LAPB. Neste tipo de quadro, existem 5 bits para identificar o tipo de quadro, embora nem todas as 32 opções possíveis sejam utilizadas.

# 7. Como a camada de rede utiliza os serviços oferecidos ?

Pelo que vimos no item anterior, a camada de rede deve ser responsável pela seleção e pedido de serviços da camada de enlace. Como estamos falando de serviços padronizados pelo modelo OSI, devem existir mecanismos especiais para solicitar e receber serviços entre as camadas de rede e de enlace. Estes mecanismos são chamados de **primitivas**. Existem quatro primitivas básicas: de **pedido**, de **indicação**, de **resposta** e de **confirmação**.

Para entender melhor a operação e funcionamento das primitivas, vamos representá-las dentro de uma comunicação típica baseada no modelo OSI. Nesta representação, veremos dois equipamentos (emissor e receptor) comunicando-se através de um meio físico. Vamos representar as três camadas inferiores do modelo OSI (física, enlace e rede), para que possamos acompanhar o encaminhamento e recepção de cada uma das primitivas.

Observando também a figura, veremos que as primitivas de **pedido** e de **confirmação** são usadas no emissor, enquanto que as primitivas de **resposta** e de **indicação** são usadas no receptor. No entanto, apesar de acontecerem em pontos separados, estas primitivas estão diretamente relacionadas uma com as outras.

## 7.1. As primitivas de pedido e de indicação :

Durante um processo de transmissão, as solicitações costumam passar das camadas superiores para as camadas inferiores. Desta forma, durante a

transmissão, a camada de rede solicita serviços à camada de enlace, como a transmissão de um quadro ou mesmo a abertura de uma conexão.

Ao receber esta solicitação, a camada de enlace do lado receptor informa à camada de rede tal recepção, através de uma primitiva de indicação. Cabe então à camada de rede passar estas informações para as camadas superiores, se isto for necessário.

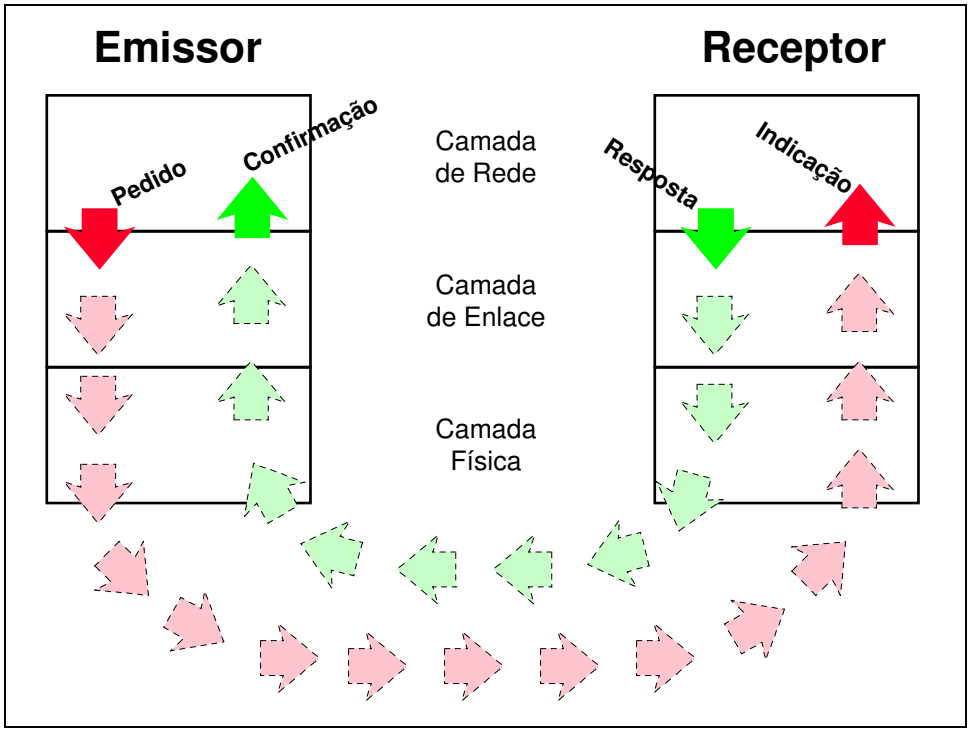



**Figura 7-A -** Primitivas de interligação rede/enlace

## 7.2. As primitivas de resposta e de confirmação :

A solicitação de serviço feita pela camada de rede do transmissor pode gerar resposta da camada de rede do receptor. Nestes casos, a camada de rede do receptor utiliza a primitiva de resposta para encaminhá-las.

Analogamente, a camada de rede do transmissor recebe tal resposta com uma primitiva de confirmação, que indica à mesma que o serviço solicitado foi devidamente realizado.

**Exercícios :**

1. Dentre os métodos de enquadramento, qual é aquele que fere o conceito de modelo em camadas? Por quê?

2. Qual a técnica utilizada para evitar a marcação indevida de início de quadro quando utilizamos um caractere especial para isso? Explique.

3. Por quê não faz sentido um serviço com conexão e sem confirmação?

4. O quê deve ser feito em um serviço com conexão para evitar quadros repetidos? Em que situação isso poderia ocorrer?

5. “O quadro não chegou”, “O quadro chegou com defeito”, “O quadro chegou perfeito”; em quais das situações:

a) O protocolo simplex pode ajudar?
b) O protocolo STOP-AND-WAIT pode ajudar?
c) Nenhum protocolo pode ajudar. Porquê?

6. Em que tipo de aplicação você utilizaria um serviço com conexão? Por quê? E o serviço simplex? Compare as duas.

7. Coloque em ordem de complexidade:

a) Sliding windows GO-BACK-N
b) STOP-AND-WAIT
c) Simplex
d) Sliding windows SELECTIVE REPEAT
e) STOP-AND-WAIT para canal com ruído.

Explique as principais modificações entre a terceira e a quarta tecnologia.

8. Quem determina o tamanho do buffer de recepção no protocolo sliding windows GO-BACK-N?. Por quê?

9. Em um protocolo PIGGYBACKING, o quê pode determinar um envio de ACK isolado?

10. Porque o PIGGYBACKING influencia o tamanho do buffer de transmissão em um protocolo SLIDING WINDOWS SELECTIVE REPEAT ?