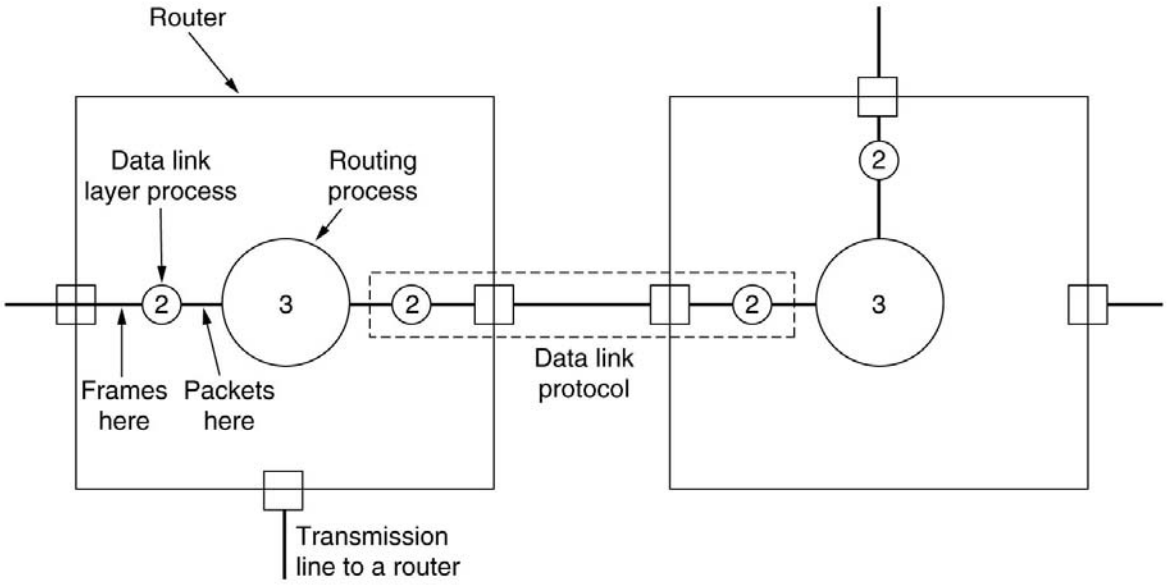
3. Nível de Enlace de Dados

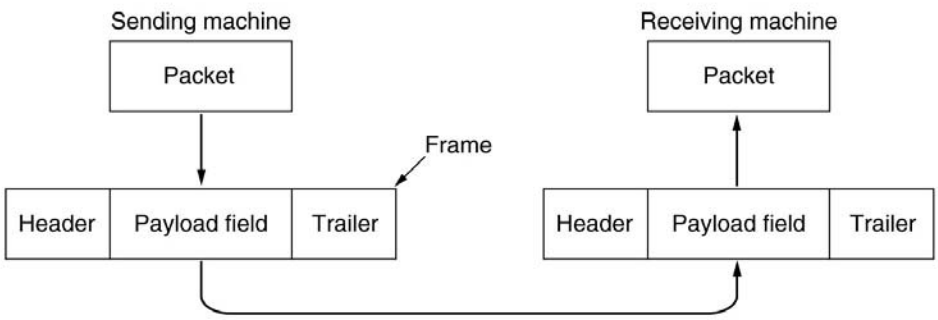
- ✓ É a segunda camada OSI;
- ✓ Fornece uma comunicação eficiente entre duas máquinas adjacentes (fisicamente conectadas).



3.1 Questões de projeto - Funções

1. Prover uma interface de serviços bem definidos para a camada de rede;
2. Lidar com erros de transmissão
3. Regular o fluxo de dados.

- ✓ Recebe os pacotes da camada de rede e os encapsula em quadros (*frames*).



relacionamento entre pacotes e quadros

3.1.1 Serviços oferecidos à camada de rede

a. Serviço não orientado à conexão sem confirmação

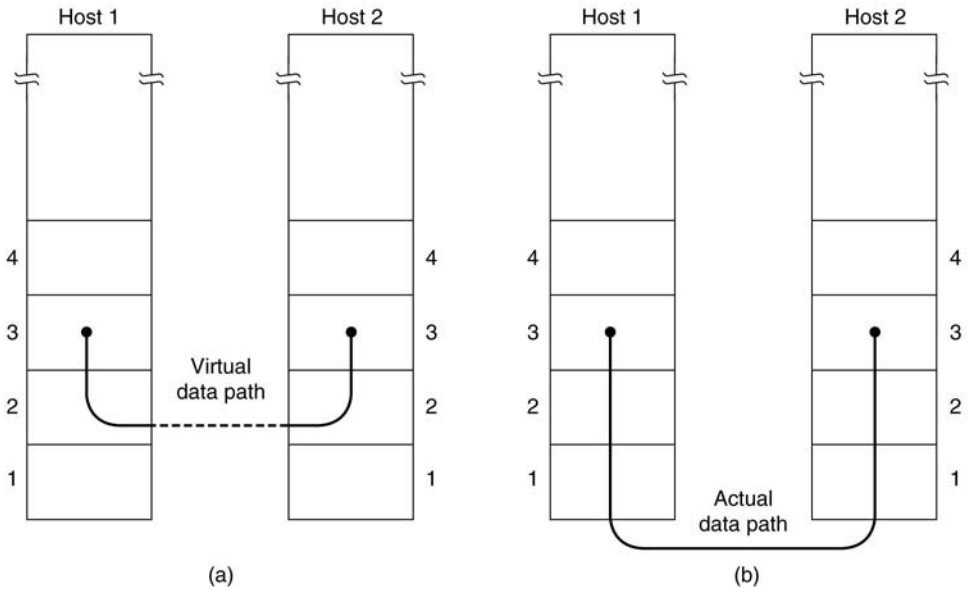
- ✓ Não há estabelecimento prévio de uma conexão antes da transmissão;
- ✓ Se um *frame* for perdido, nada é feito para recuperá-lo;
- ✓ Apropriado onde a taxa de erro é muito pequena, deixando a tarefa de recuperação de erros para as camadas superiores;
- ✓ Encontrado na maioria das LANs, em sistemas de tempo real e de voz.

b. Serviço não orientado à conexão com confirmação

- ✓ Não há estabelecimento prévio de uma conexão antes da transmissão;
- ✓ A confirmação de recebimento é feita para cada *frame* transmitido;
- ✓ A fonte sabe quando um *frame* foi bem recebido ou não;
- ✓ Útil quando o canal é não confiável, como em comunicação sem fio;
- ✓ A confirmação geralmente no transporte. ⇒canais pouco confiáveis ⇒enlace já que muitas retransmissões de mensagens ⇒ maior *overhead* que de *frames*.

c. Serviço orientado à conexão com confirmação

- ✓ Estabelecimento prévio de uma conexão antes da transmissão;
- ✓ Cada *frame* é numerado e entregue em ordem, sem duplicatas e garantido;



(a) comunicação virtual (b) Comunicação real

3.1.2 Enquadramento (*framing*)

✓ Meio físico⇒fluxo de bits sem garantias (perda, inserção e inversão);

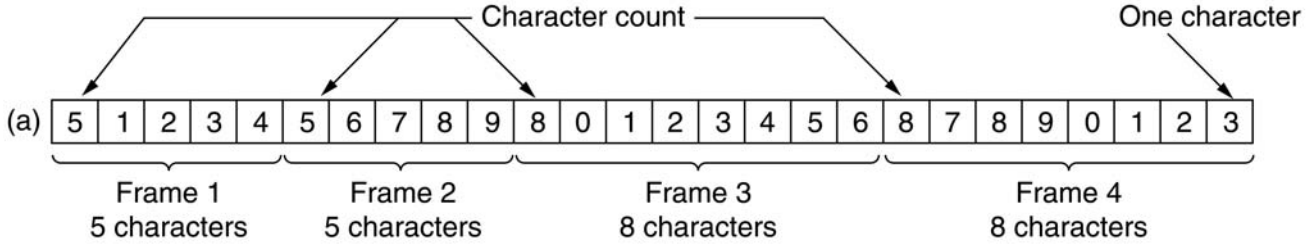
✓ Enlace⇒ dividir a cadeia de bits em quadros (*frames*) ⇒ *checksum*;

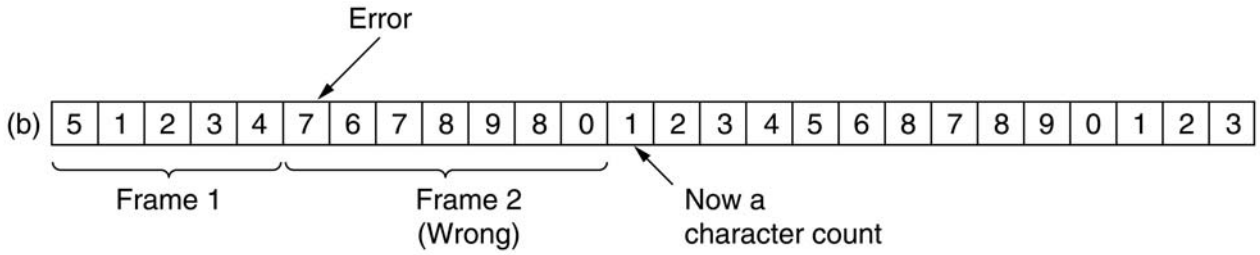
Métodos:

1. Contagem de caracteres:

✓ Uso de um campo no cabeçalho para especificar o tamanho do *frame*;

✓ Se houver erro no campo (ou mesmo no *checksum*), será impossível descobrir onde começa o próximo ⇒ perda de sincronismo.



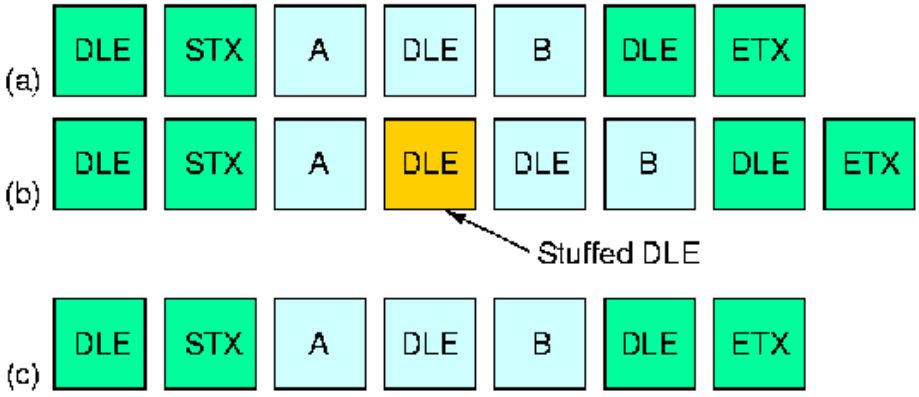


2. Bytes de flags com inserção de bytes (caracteres)

✓ Cada *frame* inicia com uma sequência ASCII DLE STX (*Data Link Scape* e *Start of TeXt*) e termina com uma sequência DLE ETX (*End TeXt*);

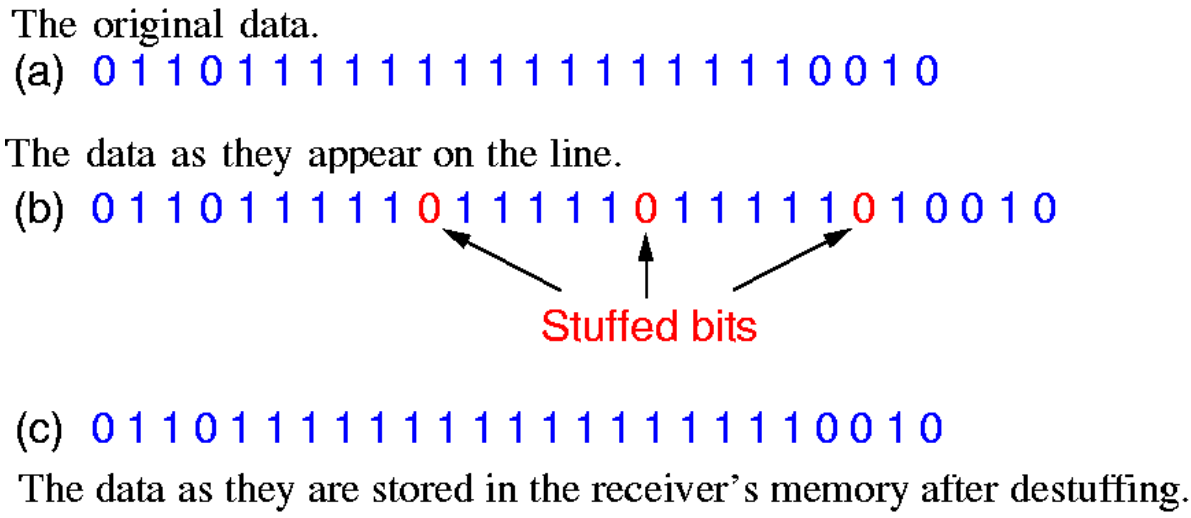
✓ Como pode ocorrer DLE STX ou DLE ETX na seq. de dados⇒ duplicam-se os DLEs dos dados para diferenciá-los dos marcadores;

✓ A desvantagem é que se amarra a transmissão ao esquema ASCII⇒ impossibilita transmissão de dados de tamanhos arbitrários.



3. Flags de início e fim com inserção de bits

- ✓ Permite que o *frame* contenha números arbitrários de bits e códigos de caracteres com quaisquer quantidades de bits por caracter;
- ✓ Cada *frame* começa e termina com um padrão especial 01111110, o flag;
- ✓ Quando a camada de enlace do fonte encontra 5 1's consecutivos, ela insere um 0 na cadeia a ser enviada;
- ✓ Quando o receptor encontra 5 1's seguidos de um 0, ele exclui este último. A inserção do bit 0 é totalmente transparente à camada de rede;
- ✓ Se os dados do usuário contém 01111110, este dado é transmitido como 011111010, mas depositado na memória do receptor como 01111110;
- ✓ Caso haja perda de sincronismo, basta procurar pelo *flag* que só ocorrerá nos limites do *frame*.



4. Violação da codificação da camada física

- ✓ Só se aplica em redes cuja codificação do nível físico possui alguma redundância;

ex. bit 1 é um par alto/baixo e bit 0 um par baixo/alto – LAN. Se ocorrer alto/alto ou baixo/baixo, houve erro de transmissão.

NOTA:
Pode-se utilizar uma combinação de um dos métodos com a contagem de caracteres, para maior segurança. O *frame* só será usado se houver um delimitador na posição indicada e o *checksum* estiver ok.

3.1.3 Controle de Erros

- ✓ Garantir a entrega ordenada de quadros (serviço orientado à conexão);
- ✓ Utiliza *frames* especiais de controle (enviados pelo receptor);
- ✓ Controle da perda total de *frames* (evitar *dead-locks*) ⇒ *timers*;
- ✓ Controle da recepção de quadros repetidos⇒ números de seqüência;

3.1.4 Controle de Fluxo

- ✓ Não permitir que a fonte envie *frames* numa taxa maior que a capacidade de recepção do destino;
- ✓ Estabelecer um protocolo em que o receptor habilite a fonte a enviar um número definido de mensagens e aguardar a próxima liberação de envio.

3.2 Detecção e correção de erros

- ✓ Erros de transmissão são freqüentes, devido à natureza física dos meios de transmissão (*loops* locais, rádio, cabos, etc);
- ✓ Erros tendem a ocorrer em rajadas (mais do que isoladamente);
  - Vantagem: fáceis de detectar (pois afetam grandes quantidades de bits);
  - Desvantagem: mais difíceis de se corrigir que erros isolados.
- ✓ Campo de redundância: metadados necessários à avaliação da consistência do campo de dados.

FLAG	Cabeçalho	Dados	Redundância	Final	FLAG

Um quadro de enlace

Estratégias:

a. Códigos de correção de erros:

- ✓ Grande redundância, suficiente para que o destino corrija erros;
- ✓ Usados em canais não confiáveis, que geram muitos erros de transmissão (como enlaces sem fio);
- ✓ Se houver erro, a origem deve retransmitir o quadro.

b. Códigos de detecção de erros:

- ✓ Redundância pequena, apenas para detectar os erros, mas não corrigi-los;
- ✓ Usados em canais confiáveis, com baixas taxas de erro (cabos de boa qualidade e fibra ótica);
- ✓ Se houver erro, o destino deve corrigir o quadro, uma vez que a retransmissão poderá gerar novos erros durante o reenvio.

☞ Canais *simplex* ⇒ impossibilidade de retransmissão⇒ correção de erros;
☞ Demais canais ⇒ possibilidade de retransmissão⇒ detecção de erros;

3.2.1 Códigos de correção de erro

Frame: n=m + r (m – bits de dados, r – bits de redundância - *checkbits*)
Palavra codificada em código n-bit.

Distância de Hamming: número de posições binárias em que dois códigos se diferem. Dada uma distância de *d* bits, será necessário corrigir *d* erros de bits isolados para se converter uma palavra em outra.

10001001 XOR
10110001
00111000 ⇒ distância = 3

Para se corrigir d erros, é necessário um código 2d+1 pois desta forma as palavras-código são tão diferentes que mesmo com d mudanças elas ainda se mantém mais parecidas com o código original.

Exemplo de correção de erro:

0 – 00000 1 - 11111
Distância de Hamming = 5⇒ 2d+1 = 5 ⇒ d=2 pode corrigir dupla inversão

Palavras de código válidas (Código com 4 palavras-código):
00000.00000 00000.11111 11111.00000 11111.11111

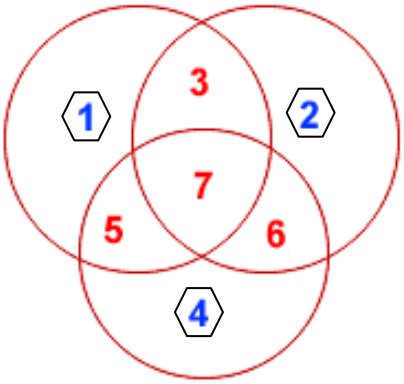
Original	Erro	Correção	estado
00000.11111	00000.00111	00000.11111	Ok
00000.00000	00000.00111	00000.11111	falha

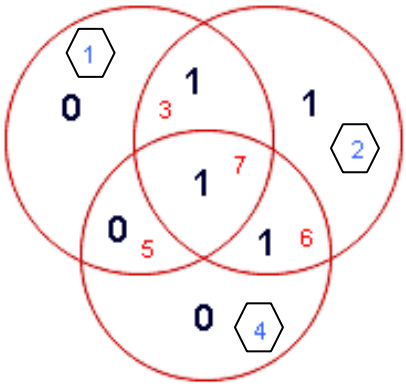
Método de Hamming para correção de erros simples

- ✓ Criação de palavras-código utilizando os bits que são potências de 2 para checagem (1, 2, 4, 8,...) e dos demais bits para dados (3, 5, 7, ..);
- ✓ Cada bit de checagem força a paridade de um grupo de bits (que o inclui);
- ✓ Um bit pode ser computado várias vezes;
- ✓ Para saber em quais bits de verificação o bit de dados k está incluso:
  7ª posição: 7=4+2+1 bits 1, 2 e 4
  5ª posição: 5=4+1 bits 1 e 4

Posições relativas: [D]ados e [P]aridade							
7	6	5	4	3	2	1	Posição
D	D	D	P	D	P	P	7 bit codeword
D	-	D	-	D	-	P	Paridade par
D	D	-	-	D	P	-	Paridade par
D	D	D	P	-	-	-	Paridade par

Exemplo: 1101							
7	6	5	4	3	2	1	Pos.
1	1	0	?	1	?	?	-
1		0		1		0	Bit 1
1	1			1	1		Bit 2
1	1	0	0				Bit 4
1	1	0	0	1	1	0	saída



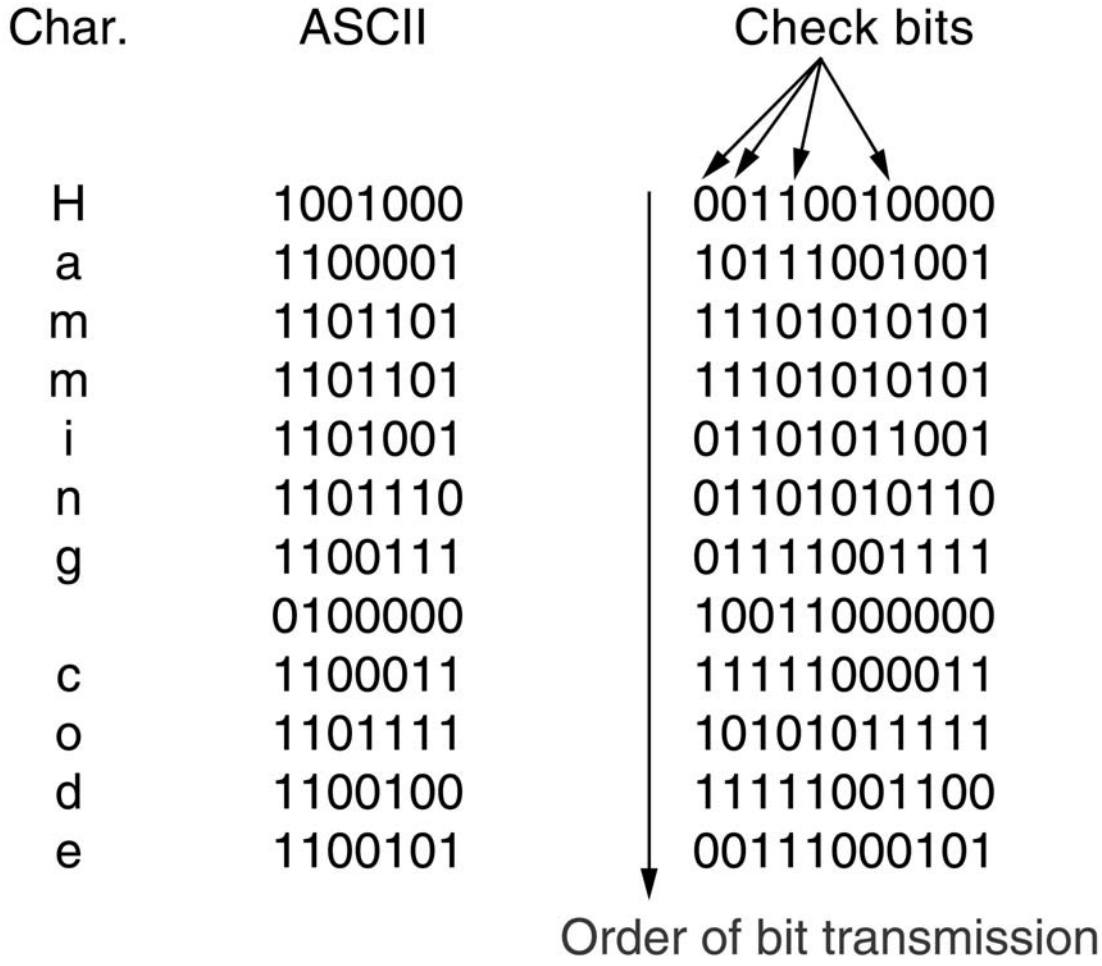


⬡ Bits de paridade

Funcionamento:

Quando uma palavra-código chega⇒ contador =0;
Para cada bit verificador (k=1, 2, 4, 8, 16, ...):
    Verificar paridade. Se errada⇒ contador=contador+k;
Se contador = 0 ⇒Ok
Senão: Contador possui o bit que está invertido.

Exemplo: Se os bits 1,2 e 8 estão errados⇒bit 11 invertido, pois ele é o único bit que é verificado pelos bits 1, 2 e 8.



Transmissão na forma matricial: uma coluna por vez ⇒ correção de erros em rajada.

3.2.2 Códigos de detecção de erro

- ✓ Canais wireless são ruidosos: correção de erros;
- ✓ Canais metálicos ou óticos possuem baixas taxas de erros: detecção de erros;

Exemplo: Mensagem =1.000 bits, taxa de erro 10^{-6} por bit	
Correção	Detecção
10 bits de verificação (2^{10} = 1.024) 1Mbit dados⇒ 10.000 bits check *Overhead*: 10.000 / 1 M = 1%	1 bit de paridade por bloco 1Mbit dados⇒ 2.001 bits de check *Overhead*: 2.001 / 1M = 0,2 %

Código polinomial ou CRC (*Cyclic Redundancy Check*)

- ✓ Método de detecção de erros largamente utilizado;
- ✓ Trata cadeias de bits como polinômios com coeficientes 0 ou 1;
- ✓ Um quadro de k-bits é entendido como sendo a lista dos coeficientes de um polinômio de k termos e grau k-1:
  110001⇒ 6 termos: 1,1,0,0,0,1 ⇒ $x^5+x^4+x^0$ polinômio de grau 5
- ✓ A fonte e o destino devem possuir um mesmo polinômio gerador G(x);
- ✓ O *frame* M(x) deve ser maior que o polinômio G(x);

O CRC adiciona um *checksum* ao final do *frame* de tal forma que o polinômio resultante seja divisível por G(x). Quando o destino recebe este polinômio, ele o divide por G(x). Se o resto for zero, o *frame* estará correto.

Geradores padrões:

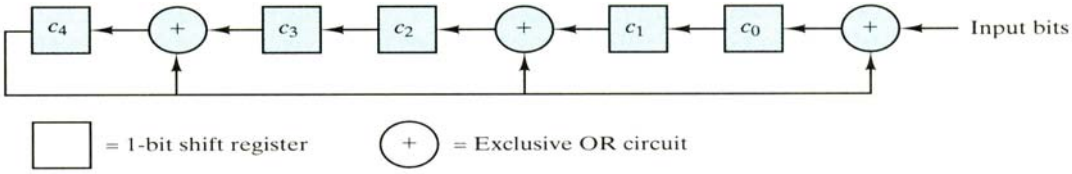
CRC-12	=	$x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$	caracter de 6 bits
CRC-16	=	$x^{16} + x^{15} + x^2 + 1$	caracter de 8 bits
CRC-CCITT	=	$x^{16} + x^{12} + x^5 + 1$	caracter de 8 bits

- ✓ A checagem de 16-bit detecta todos erros simples ou duplos, todos errros com números ímpares de bits, todas sequências de erros com 16 bits ou menos, 99,997% das sequências de erro de 17 bits e 99,998% das de 18 ou mais bits.
- ✓ Pode ser implementado em *hardware* como um circuito de *shift register*.

Frame : 1 1 0 1 0 1 1 0 1 1
Generator: 1 0 0 1 1
Message after appending 4 zero bits: 1 1 0 1 0 1 1 0 0 0 0

```
                1 1 0 0 0 0 1 0 1 0
      1 0 0 1 1 | 1 1 0 1 0 1 1 0 1 1 0 0 0 0
                  1 0 0 1 1
                  ---------
                  1 0 0 1 1
                  1 0 0 1 1
                  ---------
                  0 0 0 0 1
                  0 0 0 0 0
                  ---------
                    0 0 0 1 0
                    0 0 0 0 0
                    ---------
                      0 0 1 0 1
                      0 0 0 0 0
                      ---------
                        0 1 0 1 1
                        0 0 0 0 0
                        ---------
                          1 0 1 1 0
                          1 0 0 1 1
                          ---------
                            0 1 0 1 0
                            0 0 0 0 0
                            ---------
                              1 0 1 0 0
                              1 0 0 1 1
                              ---------
                                0 1 1 1 0
                                0 0 0 0 0
                                ---------
                                  1 1 1 0   <-- Remainder
```

Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 0

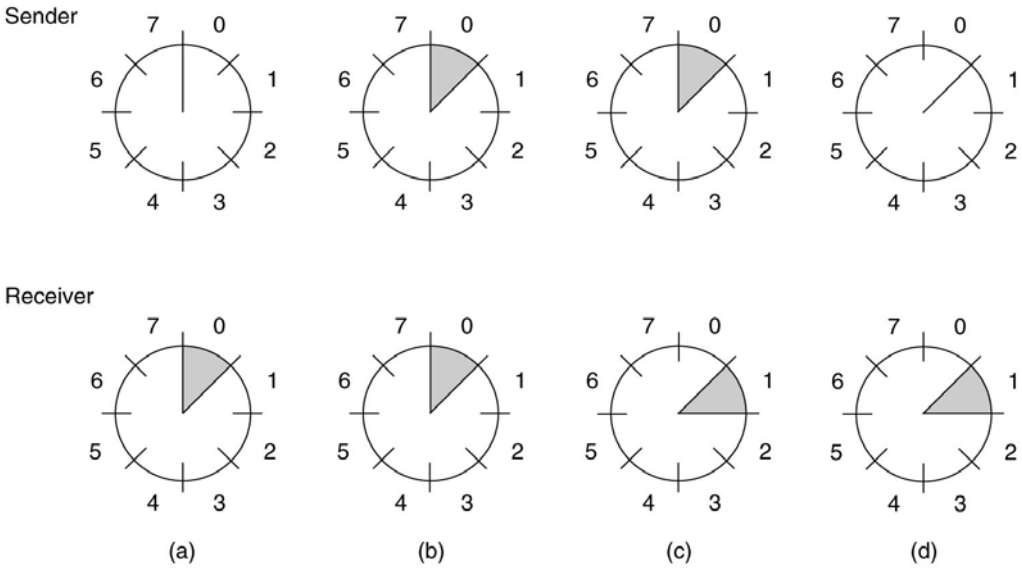


(a) Shift-register implementation

	c_4	c_3	c_2	c_1	c_0	$c_4 \oplus c_3$	$c_4 \oplus c_1$	$c_4 \oplus$ input	input	
Initial	0	0	0	0	0	0	0	1	1	
Step 1	0	0	0	0	1	0	0	0	0	
Step 2	0	0	0	1	0	0	1	1	1	
Step 3	0	0	1	0	1	0	0	0	0	
Step 4	0	1	0	1	0	1	1	0	0	Message to be sent
Step 5	1	0	1	0	0	1	1	1	0	
Step 6	1	1	1	0	1	0	1	0	1	
Step 7	0	1	1	1	0	1	1	1	1	
Step 8	1	1	1	0	1	0	1	1	0	
Step 9	0	1	1	1	1	1	1	1	1	
Step 10	1	1	1	1	1	0	0	1	0	Five zeros added
Step 11	0	1	0	1	1	1	1	0	0	
Step 12	1	0	1	1	0	1	0	1	0	
Step 13	1	1	0	0	1	0	1	1	0	
Step 14	0	0	1	1	1	0	1	0	0	
Step 15	0	1	1	1	0	1	1	0	—	

Circuit with shift registers for dividing by the polynomial $X^5 + X^4 + X^2 + 1$.

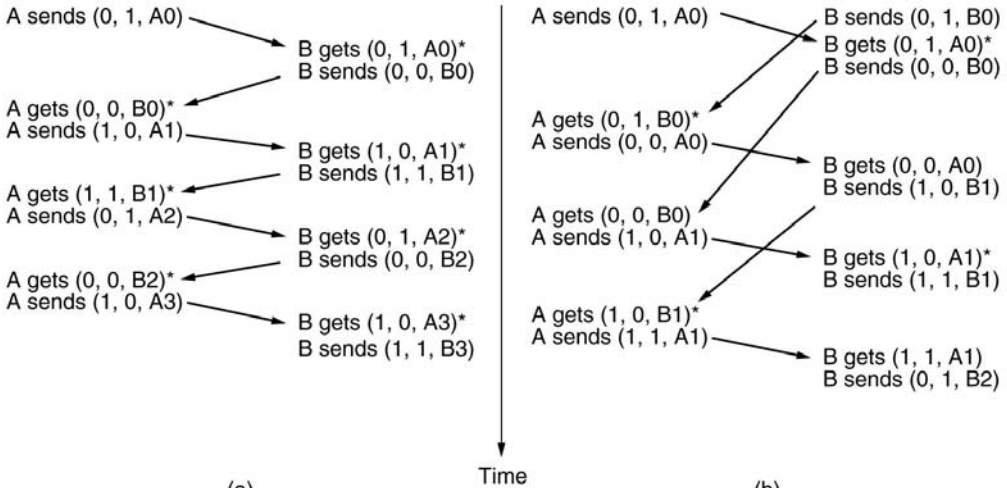
3.4 Protocolos de Janelas Deslizantes



- ✓ Em qualquer instante de tempo o transmissor mantém um conjunto de nº de seqüência dos quadros que ele pode transmitir (janela de transmissão);
- ✓ O receptor mantém um conjunto de números dos quadros que ele pode receber (janela de recepção);
- ✓ As janelas são delimitadas por seus valores inferior e superior e não precisam ser iguais no transmissor e no receptor (tamanho e limites podem diferir);
- ✓ Os quadros podem ser recebidos fora de ordem na camada de enlace, mas os pacotes devem ser entregues em ordem para a camada de rede;
- ✓ A janela do transmissor (buffer) guarda os frames transmitidos mas ainda não confirmados. Se encher⇒ transmissor pára a camada de rede;
- ✓ Quando ocorre um ack⇒ incrementa o limite inferior;
- ✓ A janela do receptor tem sempre o mesmo tamanho. Quadros com números fora da janela são descartados;
- ✓ Quando um quadro com nº seq. igual ao limite inferior é recebido, ele é passado à rede e um ack é gerado⇒ a janela é girada;
- ✓ Um tamanho de janela=1⇒quadros em ordem.
- ✓ **Piggybacking:**
  - Intercalação de quadros e enviar o *ack* anexado ao quadro de dados.
  - Quando chega um quadro de dados o receptor não envia um *ack* em seguida⇒ ele espera por algum dado a ser transmitido e anexa o *ack* do quadro anterior no seu campo de controle.
  - Se não há quadros a transmitir, o receptor envia um quadro de *ack* (por *timeout*).

3.4.1 Protocolo de Janela Deslizante de um bit

- ✓ O transmissor envia um quadro e aguarda sua confirmação antes de enviar o quadro seguinte: *stop-and-wait*;
- ✓ Transmissão em ambos sentidos (half-duplex);
- ✓ Apenas um dos lados pode iniciar a transmissão;
- ✓ Quando um frame chega, a máq. B verifica se é uma duplicata. Se não for, recebe o frame e passa o pacote à rede. Incrementa a janela de recepção;
  - O campo de ack contém o nº de seq. do último *frame* recebido sem erro;
  - Se o nº do *ack* é o mesmo do nº de seqüência que o transmissor esta tentado enviar, ele sabe que o *frame* foi recebido e pode pegar o próximo, senão, o transmissor continuará tentando enviar o *frame*;



(a) caso normal (com piggybacking)

(b) caso anormal: A e B enviam juntos
Metade dos quadros contém cópias, embora não haja erros de transmissão.

<u>Outro caso anormal: A com timeout pequeno</u>

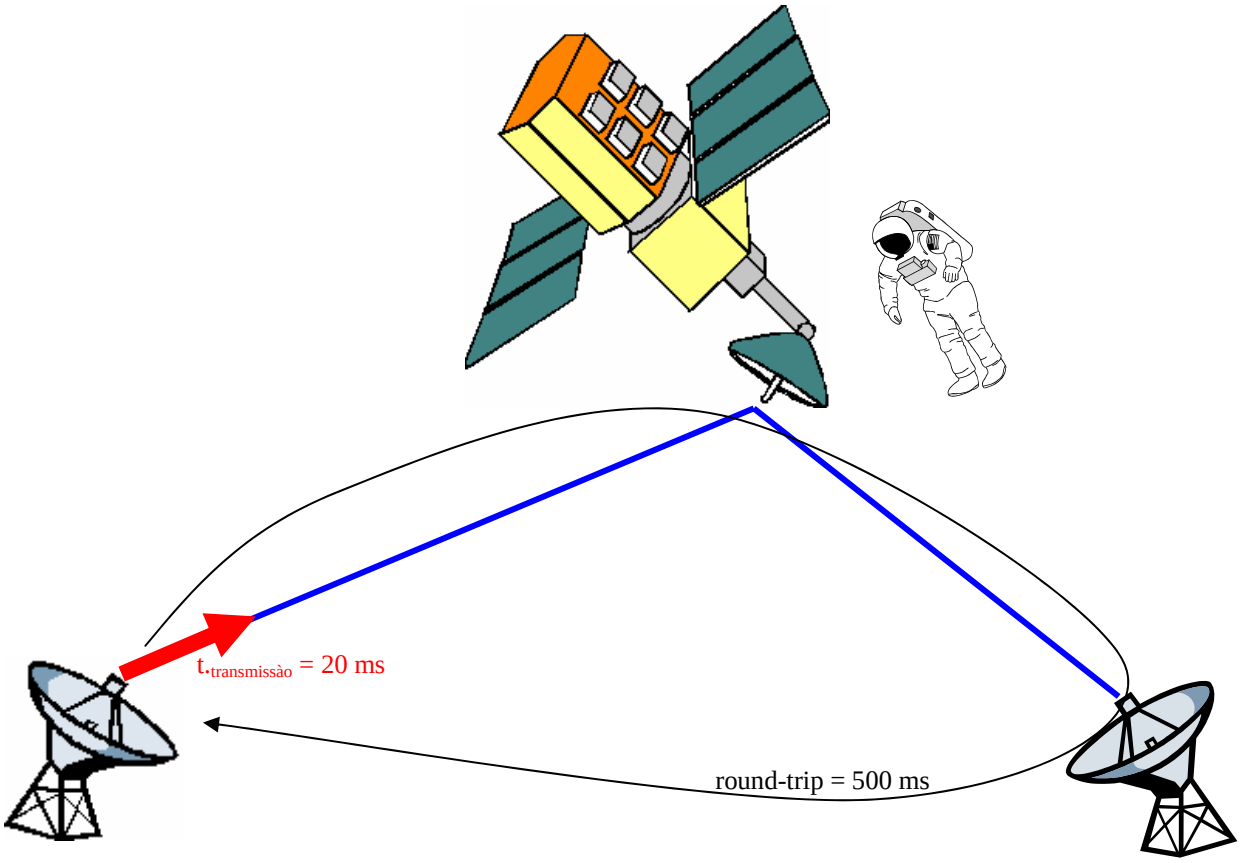
1. A tenta enviar o quadro A0 para B.
2. O *timeout* de A é pequeno e A envia repetidos quadros A0 (seq=0,ack=1,A0);
3. Quando B recebe o primeiro A0, ele incrementa *frame_expected* para 1 e rejeita os demais quadros (0,1,A0);
4. B continua esperando um ack=0, mas como todas as cópias têm ack=1, ele não pega um novo pacote da camada de rede;
5. Para cada quadro (inclusive os repetidos) que B recebe de A, ele envia um novo quadro B0 (0,1,B0). Quando um deles chega a A, A reconhece o *ack* e envia um novo quadro A1 (1,0,A1);
6. Nenhuma combinação de quadros perdidos ou *timeout* prematuros leva o protocolo à falha.

Eficiência da utilização da largura de banda

✓ *round-trip* = tempo de propagação de ida e volta de um quadro.

Exemplo:

Canal de satélite de 50kbps, com *round-trip*= 500 ms. Enviar quadros de 1.000 bits com o protocolo stop-and-wait (tamanho da janela w = 1).



$T_{geração\ frame}$ = 1.000 bit / 50.000 bit/s = 1/50 s = 20 ms
tempo para chegar ao receptor: (500 ms / 2) + 20 ms = 270 ms
tempo para retorno de ack = 500 ms /2 = 250 ms

tempo total = $tempo_{receptor}$ + $tempo_{ack}$ = 520 ms

Porcentagem de bloqueio do transmissor = 500 / 520 * 100 = 96%
Porcentagem de uso do canal = 100 – 96 = 4 %

Problema: Canal ocioso, pois o transmissor deve aguardar por um *ack* até que possa enviar uma nova mensagem.
Solução: **Pipeling**

PIPELING

- ✓ Possibilitar que o transmissor envie uma quantidade w de quadros antes de bloquear, ao invés de mandar somente 1 quadro.
- ✓ Se w for escolhido apropriadamente ⇒ transmissão de quadros contínua por um tempo igual ao *round-trip* sem enchimento da janela.

Exemplo: Qual deve ser w para o exemplo anterior ?

w = tempo total / $t_{\text{geração frame}}$ = 520 ms / 20 ms = 26 quadros (transmissor)

Quando o 26º quadro acabar de ser transmitido, estará chegando o 1º *ack*;
Os *acks* chegarão a cada 20 ms, fazendo com que haja novas transmissões;
Sempre haverá 25 ou 26 quadros não confirmados no transmissor.

Problema: E se houver perda ou dano em algum quadro?

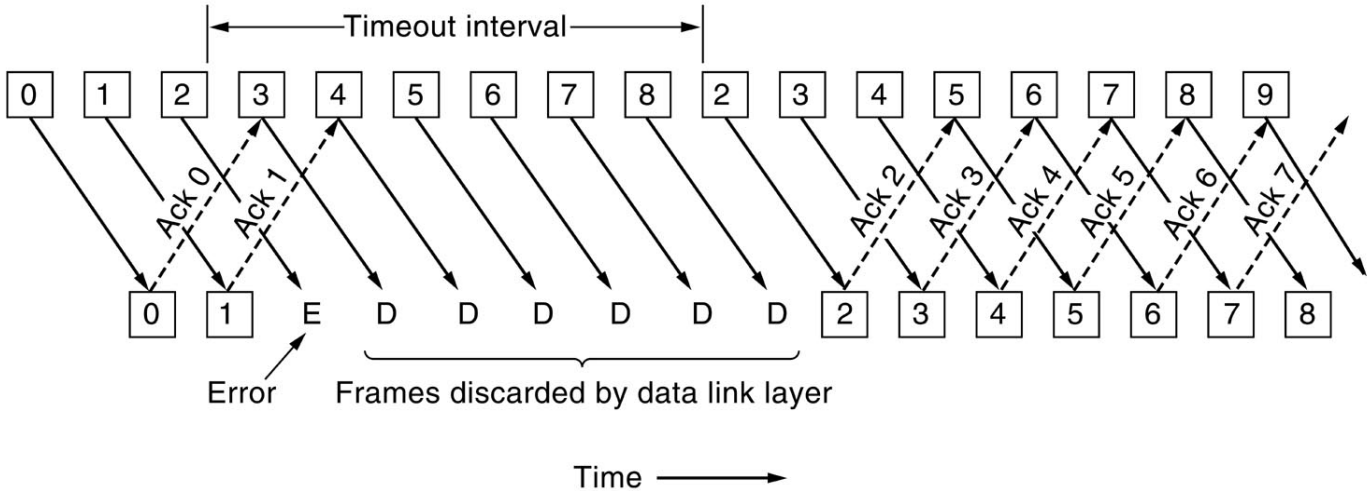
Soluções:

a. *go back n*: O receptor descarta todos os quadros após o errado;
Janela de recepção = 1. O transmissor irá dar *timeout* e retransmitir todos os quadros a partir do errado.
Não é boa estratégia em canais com altas taxas de erro.

b. Retransmissão seletiva: O receptor guarda todos os quadros bons e aguarda até que o transmissor perceba o erro e retransmita o quadro errado.
Requer grande quantidade de memória no *buffer* do receptor. w > 1.

3.4.2 *Go back n*

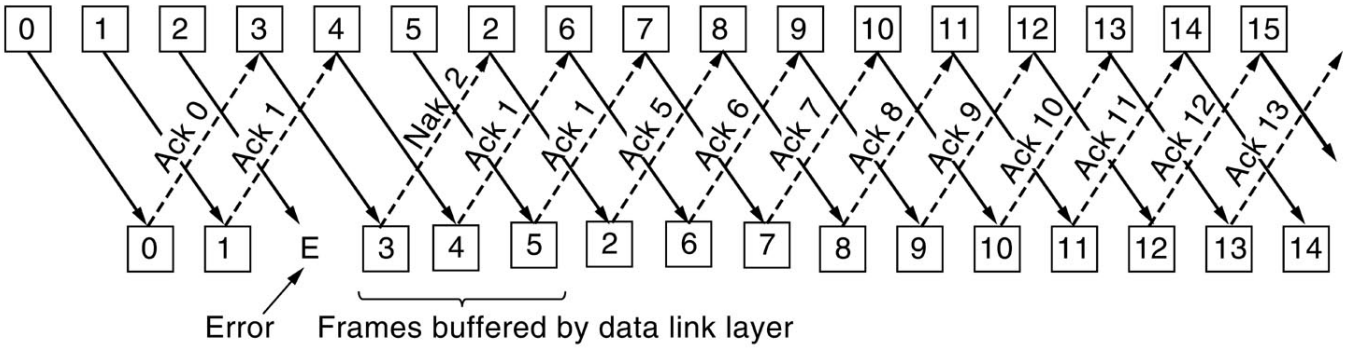
- ✓ Janela de recepção de tamanho 1;
- ✓ O receptor só aceita o próximo quadro que deve ser entregue a camada de rede e rejeita todos os demais (sem envio de *acks* para os quadros descartados) ⇒ quadros entregues em ordem;



- ✓ Na presença de um erro, o transmissor irá interromper a transmissão e retransmitirá todos os quadros já transmitidos, a partir do quadro errado.

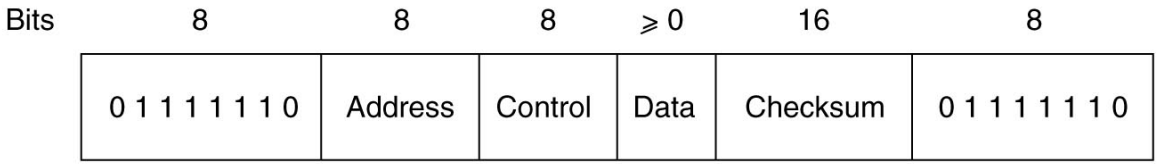
3.4.3 Retransmissão Seletiva - *Selective Repeat*

- ✓ Tanto o transmissor quanto o receptor mantém janelas maiores que 1;
- ✓ O receptor descarta um quadro incorreto, mas continua a receber os quadros corretos subseqüentes, enquanto houver buffer;
- ✓ Quando ocorrer *timeout* no quadro não confirmado, o transmissor o retransmite;
- ✓ O receptor pode enviar um NAK (*negative acknowledgement)* para forçar a retransmissão antes de um *timeout*.



3.6.1 HDLC – *High-Level Data Link Control*

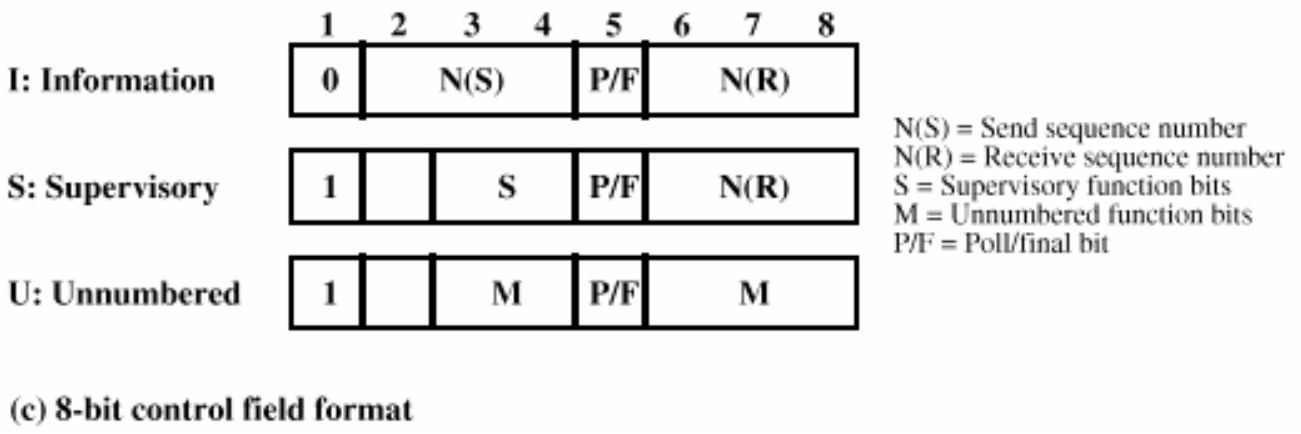
- ✓ Derivado do protocolo SNA IBM denominado SDLC - Synchronous Data Link Control;
- ✓ Adotado e modificado pelo ANSI com o nome ADCCP (Advanced Data Communication Control Procedure);
- ✓ Adotado e modificado pela ISO com o nome HDLC;
- ✓ Adotado e modificado pelo CCITT para o LAP - Link Access Procedure como parte do padrão X-25;
- ✓ Posteriormente modificado para LAPB para torná-lo mais compatível com a última versão do HDLC;
- ✓ Orientado a bit;
- ✓ Utiliza inserção de bits;
- ✓ Transmissão síncrona na forma de frames;
- ✓ Mesmo tipo de frame suporta dados e controle.



Campos:

Flag: início e fim de quadro (01111110);
Address: Usado em linhas com vários terminais (estações secundárias);
Control: Usado para números de seqüência, confirmações e outras finalidades;
Data: Dados do usuário, com tamanho arbitrário (não existe máximo, mas a eficiência diminui para quadros longos devido à maior prob. de rajadas);
Checksum: CRC.

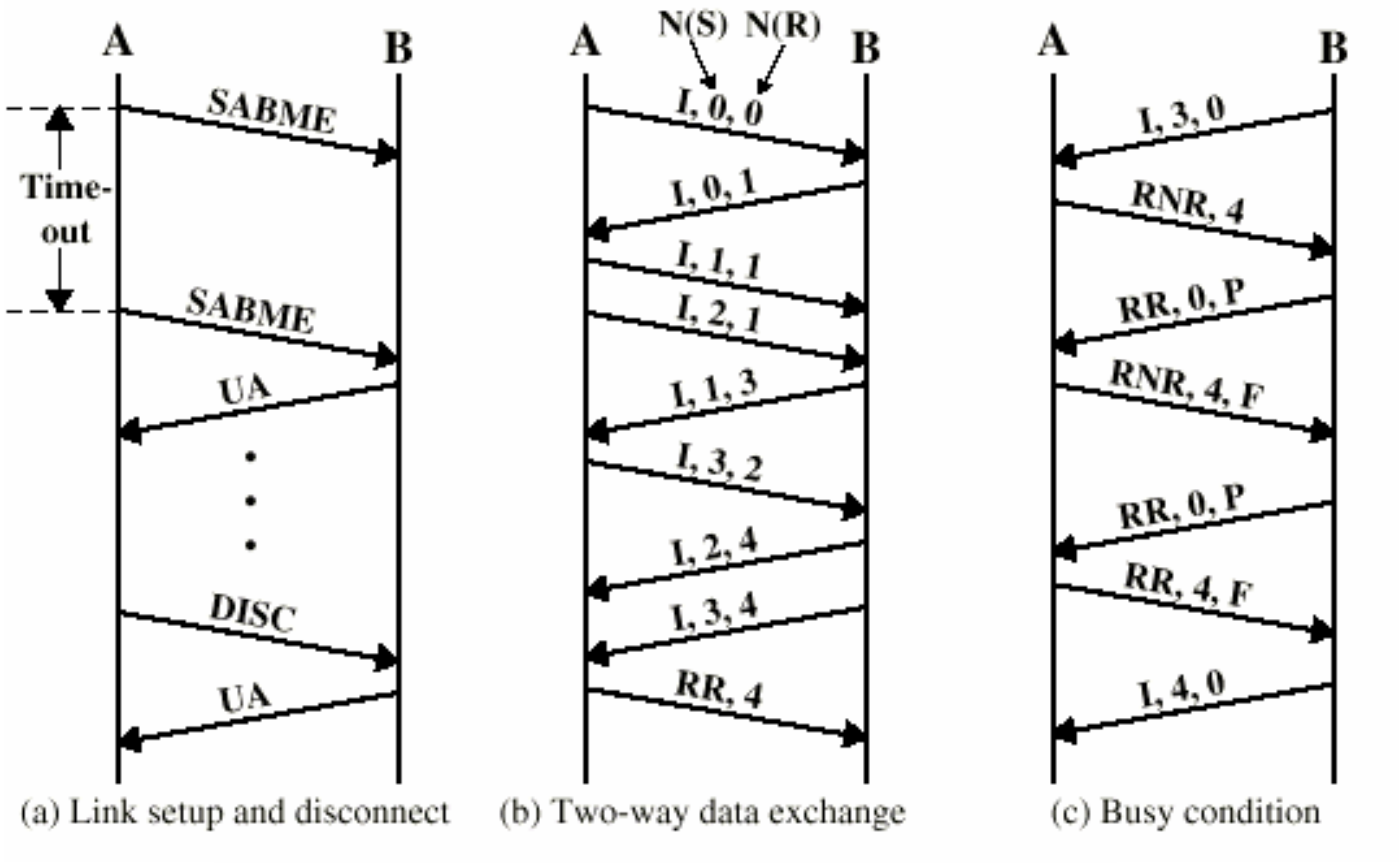
Campo de Controle



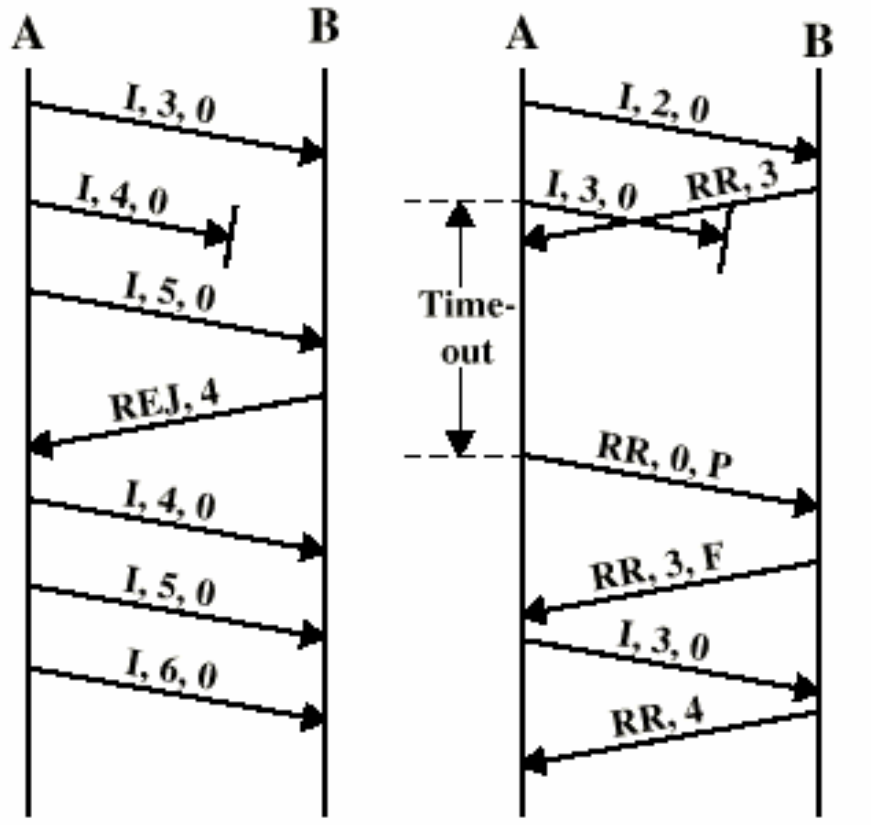
(c) 8-bit control field format

- ✓ Tipos de *Frames*:
  - Informação (I-frame);
  - Supervisão (S-frame);
  - Não-numerado (U-frame);

- ✓ N(S) - número de seqüência do frame para transmissão, com o tamanho de 3 bits usado em janela deslizante;
- ✓ N(R) - número de sequência do frame esperado. Ack(R) a ser transmitido de carona (piggybacking);
- ✓ P/F - Poll/Final - Usado pelo computador ou concentrador para fazer polling a um grupo de terminais;
- ✓ S - Tipo 0 - RECEIVE READY – RR Frame de confirmação Usado quando não há tráfego reverso para confirmar através de carona;
- ✓ S - Tipo 1 - Negative ack - REJECT – REJ confirmação negativa negative ack – NAK Quadros a partir de N(R) devem ser retransmitidos;
- ✓ S - Tipo 2 - RECEIVE NOT READY – RNR Confirma todos os frames anteriores a N(R) e pede ao transmissor para interromper transmissão temporariamente. Ao terminar o problema, envia RR, Rej ou determinados frames tipo U
- ✓ S - Tipo 3 - SELECTIVE REJECT – SR pede a transmissão apenas do quadro especificado. Existente nos protocolos HDLC e ADCCP. Não suportado no SDLC e LAPB
- ✓ DISC – Disconect: Anuncia a desconexão;
- ✓ SABM, SABME – Set Asynchronous Balanced (Extended) Mode
- ✓ UA – Unnumbered Acknowledgment: Ack não numerado para quadros de controle.



(a) Link setup and disconnect (b) Two-way data exchange (c) Busy condition



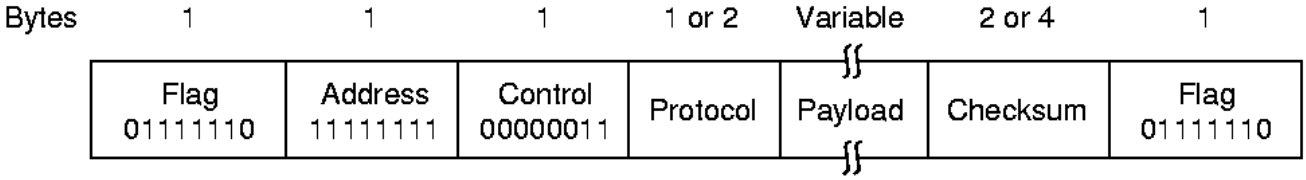
(d) Reject recovery (e) Timeout recovery

3.6.2 PPP – *Point-to-Point Protocol*

- ✓ Protocolo de enlace ponto-a-ponto definido pelo IETF, na RFC 1661;
- ✓ É um protocolo padrão Internet;
- ✓ Possui detecção de erros, suporta múltiplos protocolos, permite que endereços IP sejam negociados dinamicamente e permite autenticação;
- ✓ Usado em linhas discadas (*dial-up*) e conexões entre roteadores por linhas alugadas;
- ✓ O PPP possui:
  1. Um método de enquadramento que delimita inequivocamente o início e o fim de um quadro (com detecção de erros);
  2. Um protocolo de controle de enlace (LCP – *Link Control Protocol*) para estabelecimento, teste negociação de opções e finalização de uma conexão;
  3. Um protocolo de controle de rede (NCP – *Network Control Protocol*) para cada tipo de protocolo de rede suportado, de forma a negociar as opções disponíveis na camada de rede.

Formato do *frame*:

- ✓ Formato próximo ao do HDLC;
- ✓ Principal diferença: o PPP é orientado ao byte. O HDLC é orientado ao bit;



The PPP full frame format for unnumbered mode operation.

Flag: flag padrão HDLC, com inserção de caracteres se ocorrer dentro do quadro;

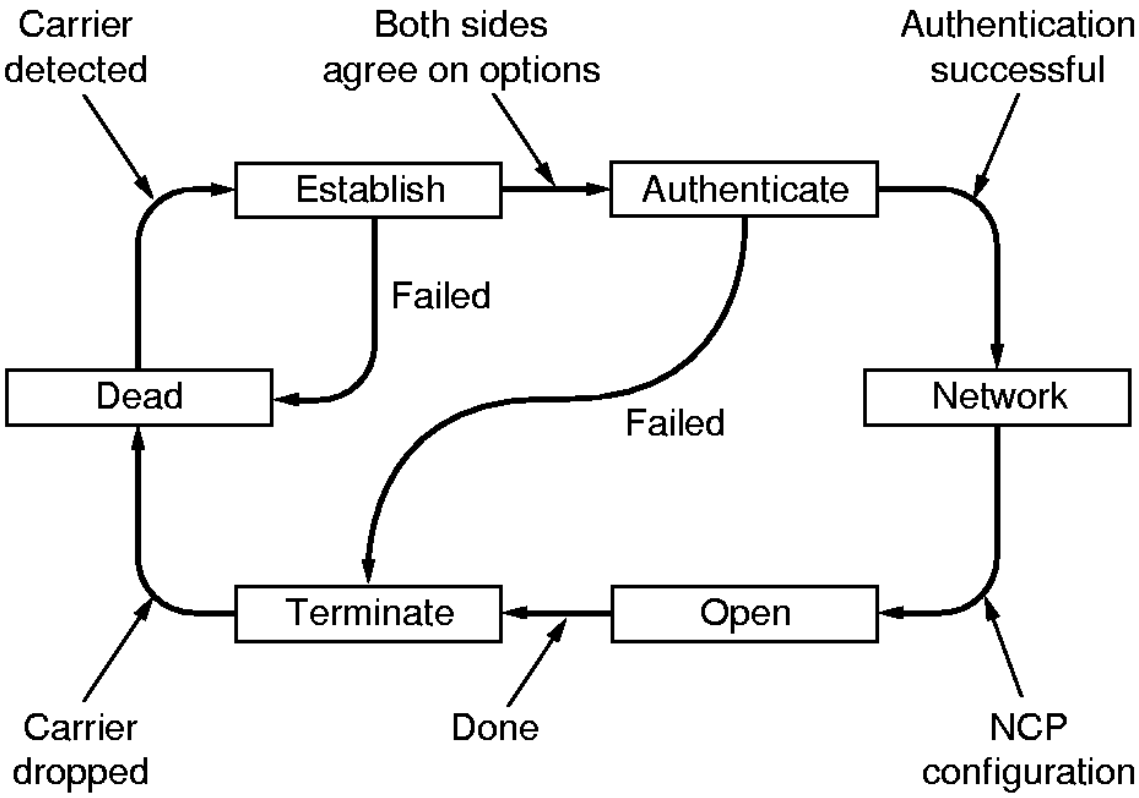
Address: sempre em 11111111 para indicar que todas as estações devem sempre aceitar o *frame* (broadcasting). Desta forma, não é preciso haver endereços na camada de enlace;

Control: Valor padrão é 00000011 (quadro não numerado) ⇒ PPP não utiliza numeração de seqüência e *ack* como padrão⇒ transmissão sem garantia. Pode-se utilizar quadros numerados em linhas ruidosas alterando-se o campo *control*;

OBS: como os valores padrões de *address* e *control* são constantes, o LCP fornece meios de omiti-los desde que haja previa negociação entre as partes.

Protocol: Informa o tipo do pacote (LCP, NCP, IP, IPX, AppleTalk, OSI CLNP, XNS, etc);
Payload: Dados a serem transmitidos. Tamanho variável (até um máximo pré-negociado ou um padrão de 1.500 bytes em caso contrário);
Checksum: Para detecção de erros.

Fases de uma conexão PPP



Dead: (Desconectado) Não existe portadora ou conexão física;
Establish: (Estabelecimento) Existe uma conexão física. O LCP inicia a negociação das opções de enlace;
Authenticate: (Autenticação) Verificação das identificações mútuas (se necessário);
Network: (Rede) O NCP é solicitado a configurar a camada de rede;
Open: (Aberto) Se a camada de rede está ok⇒ Inicia-se a transferência de dados;
Terminate: (Término) Volta à situação Desconectado, terminando a conexão através do LCP.