Chp. 03 – Link Layer

3.1 – Princípios de Projeto da Camada de Enlace
3.2 – Detecção e Correção de Erro
3.3 – Protocolos Elementares da Camada de Enlace
3.4 – Protocolos de Janela Deslizante
3.5 – Exemplos de Protocolos da Camada de Enlace

Referências Bibliográficas

- Andrew S. Tanenbaum - “Computer Networks” - Prentice Hall; Englewook Cliffs; New Jersey; 1989 2nd; 2011 5th.
- Luis F.G. Soares et al. - “Redes de Computadores – LANs, MANs e WANs às Redes ATM”; Editora Campus; ISBN: 85-7001-998-X

- Eleri Cardozo; Maurício Magalhães - “Redes de Computadores: Modelo OSI/X.25”, Dep.to de Engenharia de Computação e Automação Industrial, FEEC, UNICAMP, 1996.
- Eleri Cardozo; Maurício Magalhães - “Redes de Computadores: Arquitetura TCP/IP” - Dep.to de Engenharia de Computação e Automação Industrial, FEEC, UNICAMP, 1994.

Chp. 03 – Link Layer

- “objetivo” - descrever, entender e exemplificar os princípios de projeto da camada de enlace de dados;
- … entender algoritmos para comunicação eficiente e confiável entre 02 “hosts” adjacentes no nível da camada de enlace.
  - … adjacentes - 02 máquinas fisicamente conectadas por meio de um canal (e.g., cabo coaxial, linha telefônica ou canal sem fio ponto a ponto).
  - … fato de um canal se comportar como um fio é o fato de os bits serem entregues na ordem exata em que são enviados.
  - … limitações como taxa de transferência, erros no canal e atraso de propagação têm implicações na eficiência da transferência de dados.

3.1 – Princípios de Projeto da Camada de Enlace

- Destacam-se como “funções da camada de enlace”:
- prover interface de serviço bem definida à camada de rede;
- tratar erros de transmissão ao longo do canal de comunicação;
- regular o fluxo de dados, de tal forma que receptores lentos não sejam atropelados por transmissores rápidos.

- … para alcançar estes objetivos a camada de enlace:
  - encapsula pacotes da camada de rede em quadros para transmissão;
  - cada quadro contém um “header”, um campo de carga útil ou “payload”, que conterá o pacote, e um campo final ou “trailer”.

… 3.1 – Princípios de Projeto da Camada de Enlace

- … muitos dos princípios presentes na camada de enlace, como controle de erros e o controle de fluxo, são também encontrados em outros protocolos, e.g., protocolos de transporte.
  - … não importa realmente a camada na qual estudaremos muitos destes princípios, pois os princípios são quase idênticos.

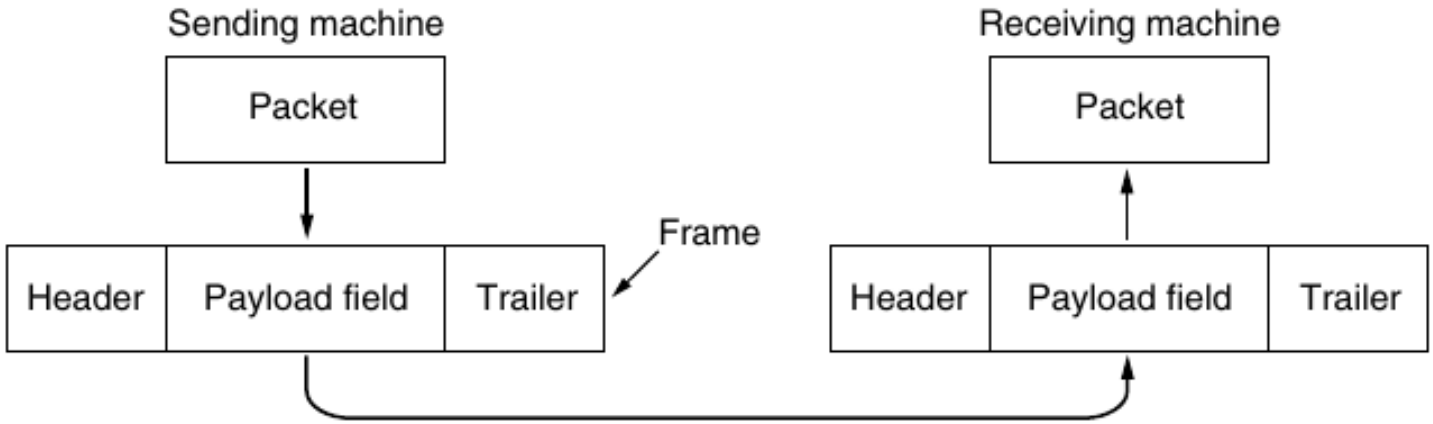


Figure 3-1. Relationship between packets and frames.

3.1.1 – Serviços da Camada de Enlace

- “serviços da camada de enlace” -
  - … transferir dados de entidades da camada de rede do “host” origem para entidades da camada de rede do “host” destino;
  - … tarefa da camada de enlace é transmitir os bits ao “host” de destino, de forma que eles possam ser entregues à camada de rede deste “host”.

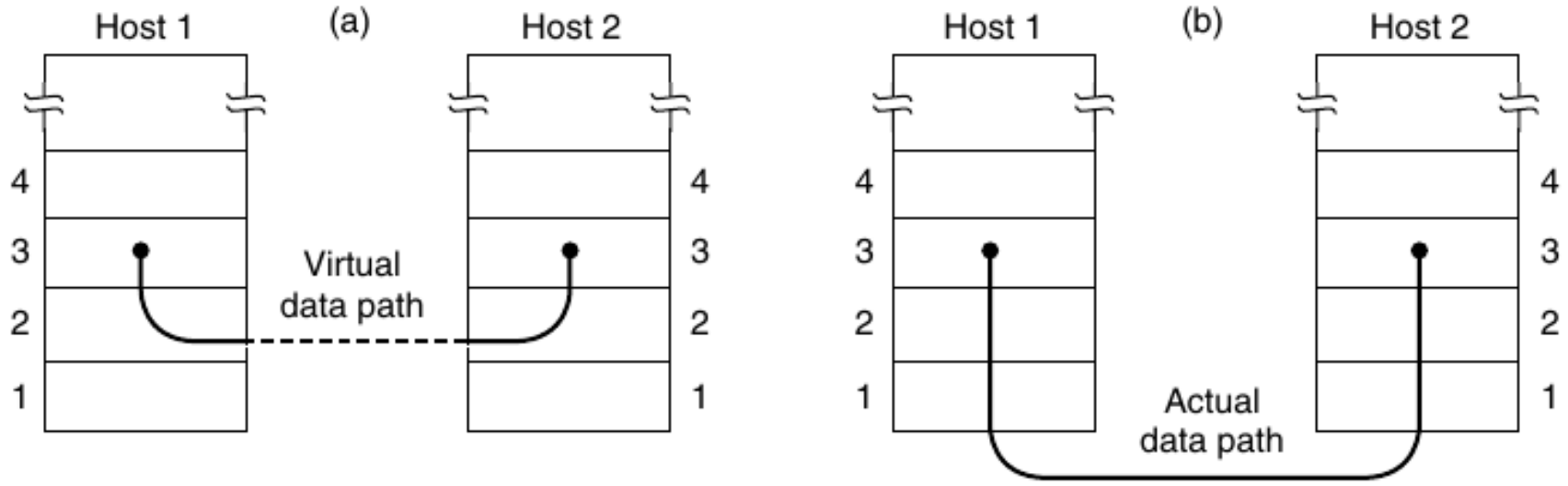


Figure 3-2. (a) Virtual communication. (b) Actual communication.

... 3.1.1 – Serviços da Camada de Enlace

- “serviços típicos da camada de enlace” - dentre os serviços oferecidos com frequência pela camada de enlace, destacam-se:
  - serviço sem conexão e sem confirmação;
  - serviço sem conexão com confirmação;
  - serviço orientado a conexão com confirmação.

… 3.1.1 – Serviços da Camada de Enlace

- “serviço sem conexão e sem confirmação” - … “host” origem envia quadros independentes ao “host” destino, sem que o “host” de destino confirme o recebimento desses quadros.
  - … apropriado quando a taxa de erros é muito baixa e a recuperação de erros fica a cargo de camadas mais altas;
  - … apropriado para o tráfego em tempo real, no qual, a exemplo da fala humana, os dados atrasados causam mais problemas que dados recebidos com falhas.

… 3.1.1 – Serviços da Camada de Enlace

- “serviço sem conexão com confirmação” - “host” origem envia quadros ao “host” destino, mas cada quadro enviado pelo “host” origem é individualmente confirmado.
  - … transmissor sabe se um quadro chegou corretamente ou não e, caso não chegue dentro de um intervalo de tempo específico, o quadro poderá ser reenviado – útil em canais não confiáveis.
  - … em canais confiáveis, como em fibra óptica, o uso de um protocolo de enlace de dados muito sofisticado pode ser desnecessário mas, em canais sem fio, com sua inerente falta de confiabilidade, o custo compensa.

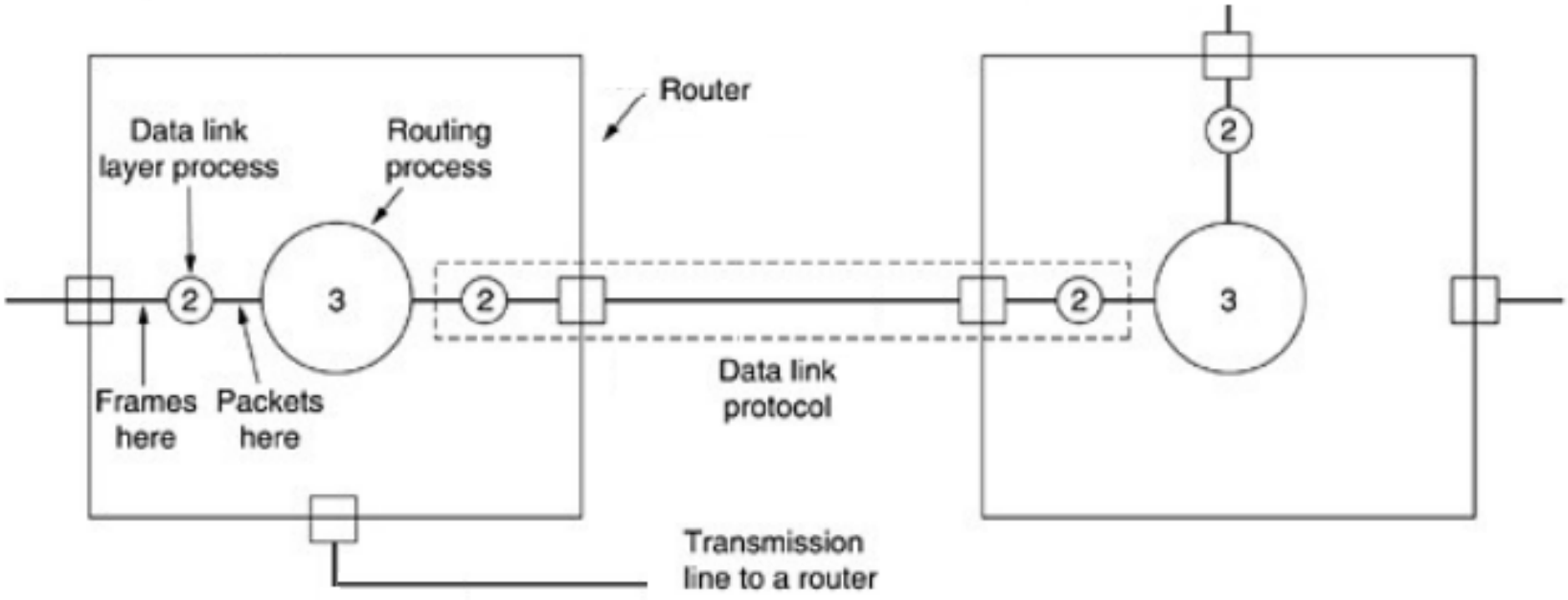
... 3.1.1 – Serviços da Camada de Enlace

- “serviço orientado a conexão com confirmação” - “host” origem e “host” destino estabelecem uma conexão antes de que os dados sejam transferidos.
  - … cada quadro enviado pela conexão é numerado, e a camada de enlace de dados garante que cada quadro será de fato recebido;
  - … garante que todos os quadros serão recebidos uma única vez, ou seja, elimina-se as duplicações bem como mantém-se a ordem.

- “serviço orientado a conexão” - ... pressupõe as fases de estabelecimento da conexão, transferência dos dados e finalização da conexão, na qual libera-se as variáveis, os buffers e os outros recursos reservados quando a conexão foi estabelecida.

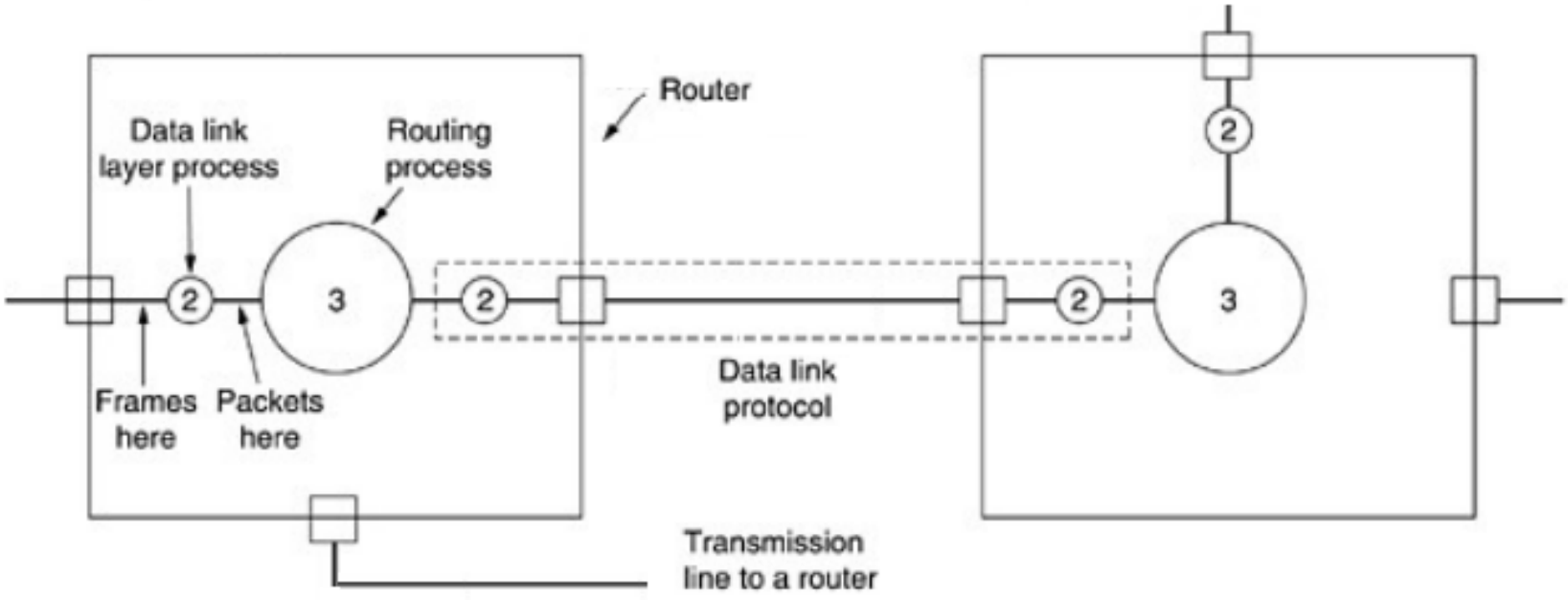
… 3.1.1 – Serviços da Camada de Enlace

- e.g., … considere uma sub-rede de uma rede metropolitana consistindo de roteadores conectados por linhas telefônicas;
  - … quando um quadro chega a um roteador, o “hardware” verifica se há erros e depois repassa o quadro à camada de enlace de dados, cujo “software” pode estar incorporado a um chip da interface de rede.



… 3.1.1 – Serviços da Camada de Enlace

- … software da camada de enlace verifica se esse é o quadro esperado e, se for o caso, passa o pacote contido no campo de carga útil - “payload” ao software de roteamento.
  - … “software” de roteamento (camada de rede), por sua vez, seleciona a linha de saída apropriada e repassa o pacote ao software da camada de enlace de dados, que o retransmite.



3.1.2 – Enquadramento

- “enquadramento” - … aceita-se um fluxo de bits brutos e tenta entregá-lo ao destino, no entanto, não há uma garantia de que esse fluxo de bits seja livre de erros.
- … estratégia adotada é dividir o fluxo de bits em quadros e calcular o soma verificação - “checksum” para cada quadro;
- … no destino, o “checksum” é recalculado e, caso, seja diferente do que está contido no quadro, a camada de enlace de dados identifica que há erros de transmissão;
- … providências neste caso, incluem descarte do quadro defeituoso e envio de relatório de erros à origem.

… 3.1.2 – Enquadramento

- “problema” - … divisão do fluxo de bits em quadros é mais difícil do que parece à primeira vista, pois a rede raramente oferece garantias em relação a temporização.

- “solução” - … como é arriscado contar com a temporização para marcar início e fim do quadro, outros métodos foram criados:
  - contagem de caracteres;
  - bytes de “flags”, com inserção de bytes;
  - “flags” iniciais e finais, com inserção de bits;
  - violações de codificação da camada física.

… 3.1.2 – Enquadramento

- “contagem de caracteres” - ... utiliza um campo no cabeçalho para especificar o número de caracteres do quadro.
  - … quando vê a contagem de caracteres, a camada de enlace de dados de destino sabe quantos caracteres devem vir em seguida e, consequentemente, onde está o fim do quadro.

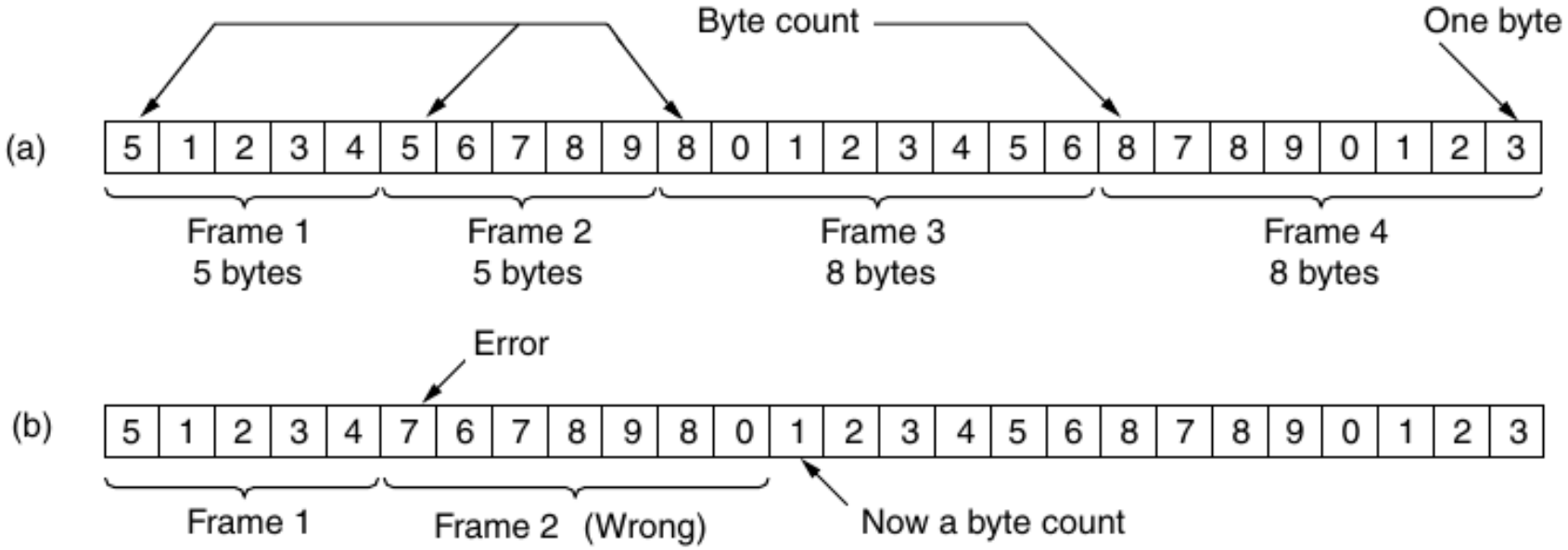


Figure 3-3. A byte stream. (a) Without errors. (b) With one error.

… 3.1.2 – Enquadramento

- “bytes de flags, com inserção de bytes” - … contorna-se o problema de ressincronização após um erro, fazendo cada quadro começar e terminar com bytes especiais.

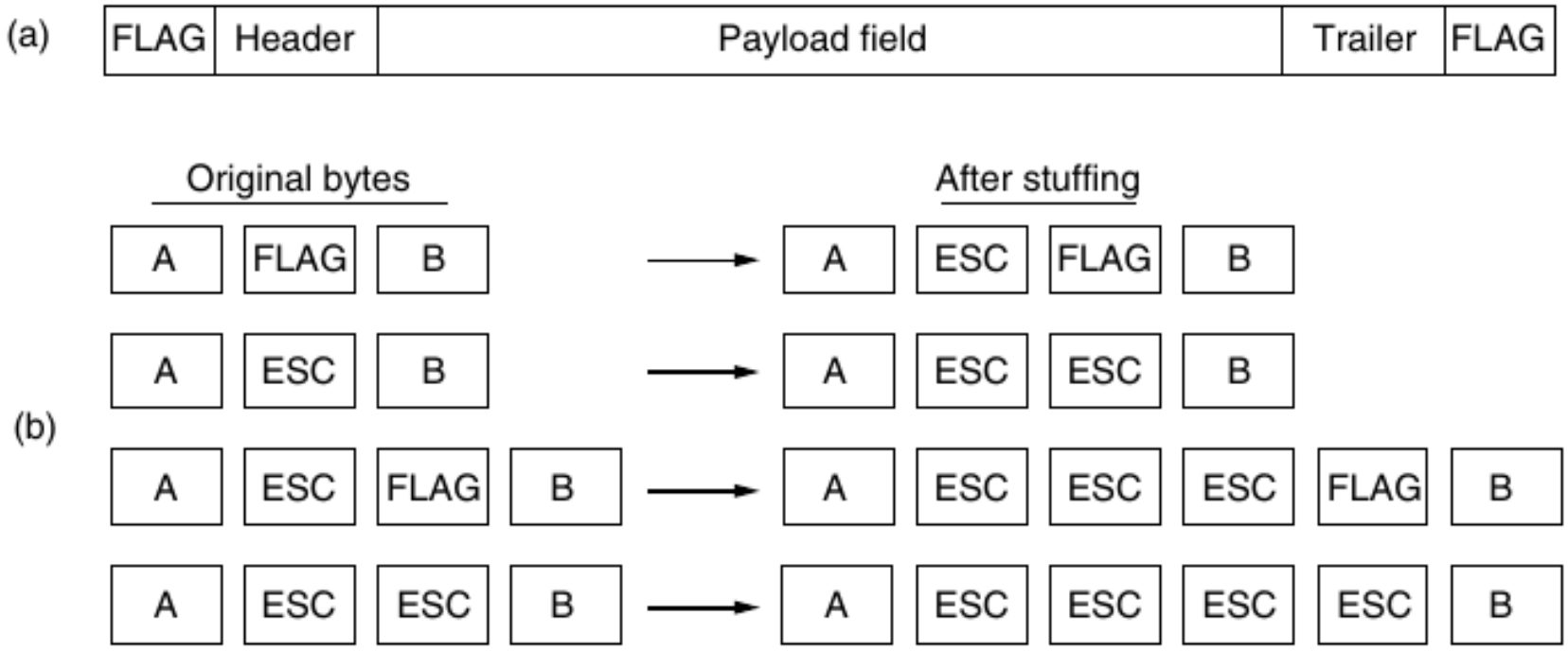


Figure 3-4. (a) A frame delimited by flag bytes.
(b) Four examples of byte sequences before and after byte stuffing.

… 3.1.2 – Enquadramento

- “desvantagem” - … utilização desse método de enquadramento depende da utilização de caracteres de 8 bits.
  - … nem todos os códigos de caracteres utilizam caracteres de 8 bits, e.g., UNICODE emprega caracteres de 16 bits.
  - … com o desenvolvimento das redes, as desvantagens da inclusão do comprimento do quadro se tornam cada vez mais óbvias.
  - … nova técnica teve de ser desenvolvida para permitir o uso de caracteres com tamanhos arbitrários de bits → flexibilidade.

- “flags iniciais e finais com inserção de bits” - permite que os quadros de dados contenham um nro. arbitrário de bits;
  - … nro. arbitrário de bits, ou seja, permite o uso de códigos de caracteres com um nro arbitrário de bits por caractere.

... 3.1.2 – Enquadramento

- … de acordo com 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, o transmissor insere um bit 0 no fluxo de bits sendo enviado;

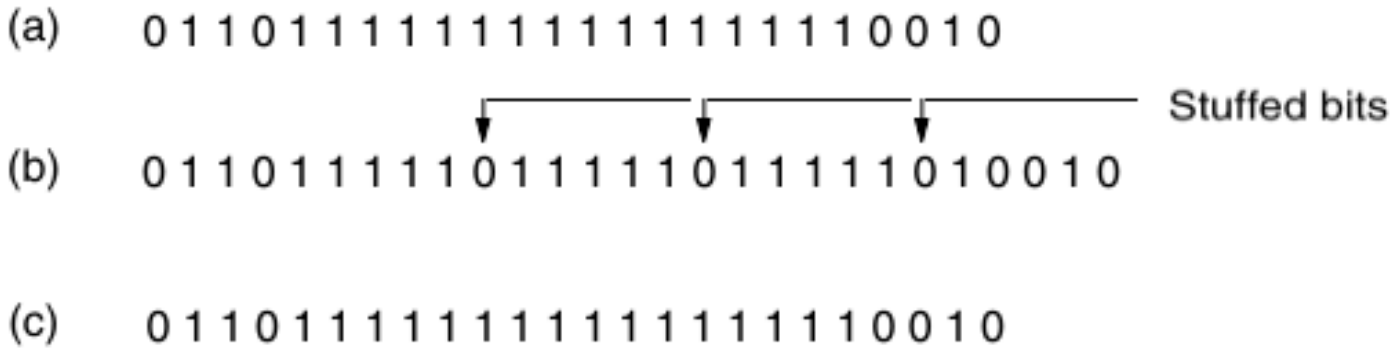


Figure 3-5. Bit stuffing. (a) The original data. (b) The data as they appear on the line. (c) The data as they are stored in the receiver's memory after destuffing.

... 3.1.2 – Enquadramento

- … essa inserção de bits é semelhante à inserção de bytes na qual um byte de ESC é inserido no fluxo de caracteres enviado antes de ocorrer um byte de “flag” nos dados.

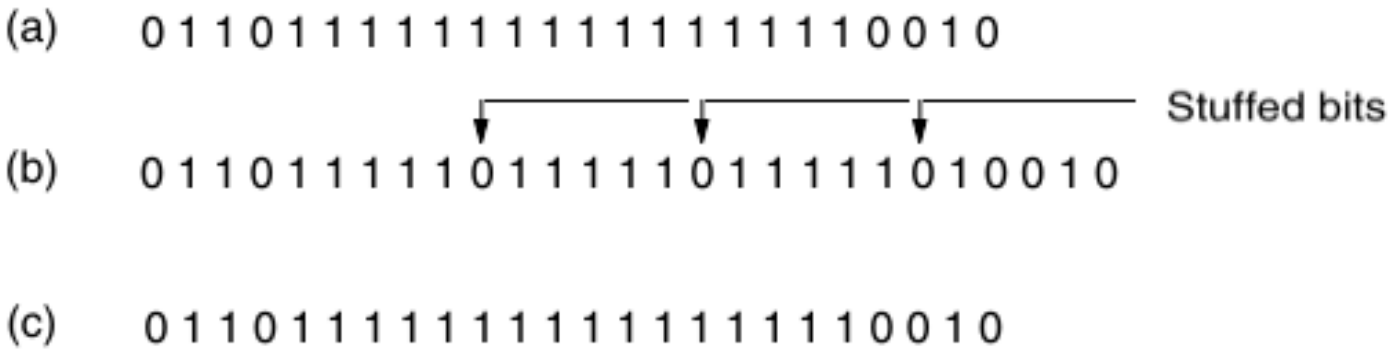


Figure 3-5. Bit stuffing. (a) The original data. (b) The data as they appear on the line. (c) The data as they are stored in the receiver’s memory after destuffing.

… 3.1.2 – Enquadramento

- “violações de codificação da camada física” - … aplicável em redes onde empregam decodificação no meio físico que contém alguma nível de redundância.
  - … algumas redes locais codificam 1 bit de dados utilizando 2 bits físicos, bit 1 contém uma transição alto-baixo, e bit 0 é uma transição baixo-alto;
  - … esquema significa que todo bit de dados tem um a transição interme-diária, facilitando a localização dos limites de bits pelo receptor;
  - … combinações alto-alto e baixo-baixo não são usadas para dados, mas são empregadas na delimitação de quadros em alguns protocolos.

3.1.3 – Controle de Erro

- "**controle de erro**" - … como ter certeza de que os quadros serão entregues na camada de rede de destino e na ordem apropriada?
  - … suponha que o transmissor simplesmente continue a enviar os quadros sem se importar em saber se eles estão chegando de maneira correta;
  - … pode ser uma ótima opção para serviços sem conexão e sem confirmação, mas sem dúvida não é para serviços orientados a conexões.

- "solução" - ... uma forma de garantir entrega confiável é retornar ao transmissor algum "feedback" sobre o que está acontecendo no outro extremo da linha – receptor;
  - confirmação positiva - quadro chegou em segurança ao destino;
  - confirmação negativa - algo saiu errado e quadro deve ser retransmitido.

... 3.1.3 – Controle de Erro

- “complicação adicional” - … problemas de “hardware” podem fazer com que um quadro desapareça completamente e, neste caso, o receptor não reagirá de forma alguma;
  - ... protocolo no qual o transmissor envia um quadro e depois espera por uma confirmação, positiva ou negativa, permanecerá suspenso para sempre caso um quadro tenha sido completamente perdido.

- “solução” - introdução de “timers” na camada de enlace de dados sempre que o transmissor envia um quadro;
  - … “timer” é ajustado para ser desativado após um intervalo suficiente-mente longo para o quadro chegar ao destino, ser processado e ter sua confirmação enviada de volta ao transmissor.

... 3.1.3 – Controle de Erro

- “camada de enlace” - … “timers” e “nros. de sequência”
- “razão” - garantir que cada quadro seja realmente passado para a camada de rede do destino exatamente uma vez.

3.1.4 – Controle de Fluxo

- “princípio de projeto” - … como encaminhar quadros pelo transmissor sem que ocorra transbordo no receptor ??
  - … muitos dos esquemas de controle de fluxo tem por princípio regras bem definidas sobre quando um transmissor pode enviar o quadro seguinte;
  - … tais regras impedem que os quadros sejam enviados até que o receptor tenha concedido permissão para transmissão, implícita ou explícita.

- e.g., quando uma conexão é estabelecida, o receptor pode informar "**você está autorizado a enviar N quadros, e na sequência aguardar até ser informado que deve prosseguir**."

3.2 – Detecção e Correção de Erros

- … embora os erros sejam raros na parte digital, eles ainda são comuns nos “loops” locais do sistema de telefonia ou mesmo na comunicação sem fio à medida que se torna mais comum;
  - … nestes casos as taxas de erros são várias ordens de grandeza piores e não se comparam com as taxas de erros dos troncos de fibra óptica.

- “conclusão” - erros de transmissão ainda estarão presentes por muitos anos nos sistemas de comunicação.

3.2.1 – Códigos de Correção de Erros

- 02 estratégias básicas para tratar os erros:
  - “códigos de correção de erros” - incluir informações redundantes suficientes em cada bloco de dados, com isso, o receptor é capaz de deduzir quais foram os dados transmitidos.
  - “códigos de detecção de erros” - incluir redundância suficiente apenas para permitir detectar se houve ou não erro, mas sem identificar de qual erro.

- “códigos de correção de erros” - são extensamente utilizados em enlaces sem fios, conhecidos por serem ruidosos e propensos a erros em comparação com a fiação de cobre ou a fibra óptica.

… 3.2.1 – Códigos de Correção de Erros

- “formalização” - quadro consiste em “m” bits de dados e de “r” bits redundantes ou bits de verificação.
  - seja o tamanho total n (isto é, n = m + r), então, com freqüência, uma unidade de “n” bits que contém bits de dados e bits de verificação é chamada palavra de código - “codeword” de “n” bits.

- e.g., dadas duas palavras de código, digamos 10001001 e 10110001, é possível determinar quantos bits correspondentes apresentam diferenças … neste caso, 3 bits divergentes.
  - … para determinar as diferenças, aplique a operação OR exclusivo entre as duas palavras de código, e conte o nro. de bits 1 no resultado.

… 3.2.1 – Códigos de Correção de Erros

- “diferença” entre 02 “code words” - definida como o nro. de bits nos quais as “code words” se diferem.
  - “Hamming Distance” - mínima diferença entre 02 “code words”

- Código com “Hamming Distance” de “n”, implica que qualquer combinação de até “n – 1” erros de bit pode ser detectada;
  - … qualquer combinação até “(n – 1)/2” de erros de bits por código pode ser corrigida se o receptor interpretar toda palavra de código não válida como a palavra de código válida mais próxima.
  - … este método é formalmente chamado de “Maximum Likelihood Decoding” ou “Nearest Neighbor Decoding”.

... 3.2.1 – Códigos de Correção de Erros

- Aumentando a Distância de Hamming, escolhendo “code words” mais longas, seremos capazes de aumentar a confiabilidade de um código tanto mais quanto maior a “code word”.

- Obs.: Redundância de um Código determina seu poder de detecção e correção de erros de transmissão.

Posição do bit		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
bits codificados		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
bits de paridade	p1	X		X		X		X		X		X		X		X		X		X		...
	p2		X	X			X	X			X	X			X	X			X	X		
	p4				X	X	X	X					X	X	X	X					X	
	p8								X	X	X	X	X	X	X	X						
	p16																X	X	X	X	X	

… 3.2.1 – Códigos de Correção de Erros

- e.g. “hamming code” - … em uma mensagem de “n” bits, inclui-se “c” bits de verificação => “n + c” bits na mensagem;
- … para “c” bits de verificação, temos: “n = 2^c -1”, assim o nro máximo de bits na mensagem será “n = 2^c – c – 1”;
- … e.g., para “n = 4” e “c = 3” (bits de paridade) => 7 bits na msg; para “n = 11” e “c = 4” (bits de paridade) => 15 bits na msg.

… 3.2.1 – Códigos de Correção de Erros

- “Hamming Code” - bits na palavra de código são numerados de “1” a “m+c” e o “i-ésimo” bit é colocado na posição “2^i” … ... para $1 <= i <= \log_2^{(m+c)}$
- … bits de verificação são colocados no palavra de código de tal maneira que a soma da posição dos bits que eles ocupam aponte para o bit errado para qualquer erro de 1 único bit;

Posição do bit		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
bits codificados		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
bits de paridade	p1	X		X		X		X		X		X		X		X		X		X		…
	p2		X	X			X	X			X	X			X	X			X	X		
	p4				X	X	X	X					X	X	X	X					X	
	p8								X	X	X	X	X	X	X	X						
	p16																X	X	X	X	X	

… 3.2.1 – Códigos de Correção de Erros

- "Hamming Code" - bits na palavra de código são numerados de "1" a "m+c" e o "i-ésimo" [0 .. m+c-1] bit é colocado na possição "2^i" … ... para $1 <= i <= \log_2^{(m+c)}$
  - … bits de verificação são colocados no palavra de código de tal maneira que a soma da posição dos bits que eles ocupam aponte para o bit errado para qualquer erro de 1 único bit;
  - … qdo um posição é escrita como a soma de potências de 2, p.ex., (1 + 2 + 4), estas potências também apontam para os bits de verificação que cobrem o bit em questão … neste caso bit de dados – 7.

3.2.2 – Códigos de Detecção de Erros

- “baixa taxa de erros” - … detecção de erros e retransmissão em geral é mais eficiente para lidar com o erro ocasional.

- “Cyclic Redundancy Check” - algoritmo de código cíclico ou “cyclic code” mais utilizado em redes de computadores;
  - ISO especifica um algoritmo similar denominado FCS (Frame Check Sequence) utilizado em seus protoclos IEEE802.

… 3.2.2 – Códigos de Detecção de Erros

- CRC contempla o produto de bits de verificação de uma primitiva, cuja amostragem é definida segundo um critério.
  - … bits escolhidos fazem parte do fator (polinômio gerador) e, portanto, é transmitido o produto, ou seja, primitiva * fator;
  - .. no destino faz-se a divisão do que é recebido pelo fator, se resto é zero significa ausência de erro ou erro não detectável;
  - … método de divisão é específico e o fator (polinômio gerador) usado determinam o leque de erros de transmissão que podem ser detectados.

… 3.2.2 – Códigos de Detecção de Erros

- Seja uma primitiva - sequência de “n” bits que pode ser representada por um polinômio de grau “n-1”;

  Somatório $b_i * x^i$ com i = [0 .. (n-1)]

  b_i é o coeficiente do bit na posição “i”

  x^i indica o literal do bit na posição “i”

- e.g., Considere a sequência de bits “10011”. Como esta sequência pode ser representada por polinômio ?

“$1*x^4 + 0*x^3 + 0*x^2 + 1*x + 1$” ou “$x^4 + x + 1$”

… 3.2.2 – Códigos de Detecção de Erros

- Sejam as operações binárias (ou-exclusivo):
  - 0 + 0 = 0 - 0 = 0
  - 0 + 1 = 0 - 1 = 1
  - 1 + 0 = 1 - 0 = 1
  - 1 + 1 = 1 - 1 = 0

- Não há “carry bit” na adição e nem “borrow bit” na subtração, ou seja, para todo “i” … $x^i + x^i = 0$
- Para multiplicar dois códigos, basta multiplicar os polinômios correspondentes aos códigos.

… 3.2.2 – Códigos de Detecção de Erros

- Dado codificado em 4 bits (p.ex., $x^2 + 1$) é multiplicado por "x + 1"
- Código gerado é o de paridade cuja "code rate" de ¾.
- É também um código cíclico, mas não é simétrico.

Table 3.4 — A Cyclic Code

Data Word	Polynomial	Multiplied By	Produces	Code Word
0 0 0	0	$x+1$	0	0 0 0 0
0 0 1	1	$x+1$	$x+1$	0 0 1 1
0 1 0	x	$x+1$	x^2+x	0 1 1 0
0 1 1	$x+1$	$x+1$	x^2+1	0 1 0 1
1 0 0	x^2	$x+1$	x^3+x^2	1 1 0 0
1 0 1	x^2+1	$x+1$	x^3+x^2+x+1	1 1 1 1
1 1 0	x^2+x	$x+1$	x^3+x	1 0 1 0
1 1 1	x^2+x+1	$x+1$	x^3+1	1 0 0 1

… 3.2.2 – Códigos de Detecção de Erros

- e.g., Calcule o polinômio cuja primitiva é “1 0 0 1 1” e o fator (polinômio gerador) é “1 1 0 0”

  “1 0 0 1 1” $\rightarrow x^4 + x + 1$

  “1 1 0 0” $\rightarrow x^3 + x^2$

  $(x^4 + x + 1) * (x^3 + x^2) = x^7 + x^6 + x^4 + x^3 + x^3 + x^2$

  … $x^7 + x^6 + x^4 + \underline{x}^3 + \underline{x}^3 + x^2 \rightarrow x^3 + x^3$ é 0 …

  $(x^4 + x + 1) * (x^3 + x^2) = x^7 + x^6 + x^4 + x^2$

… 3.2.2 – Códigos de Detecção de Erros

- “polinômios” - podem ser objetos de todas as oper. aritméticas.
  - … como a primitiva é multiplicada por um fator (polinômio gerador) para criar o código a ser transmitido, então no destino deve-se aplicar o processo inverso, i.e, usar o fator (polinômio gerador) como divisor.

- Considere-se o polinômio “$x^7 + x^6 + x^4 + x^3 + x^2$” quando dividido pelo fator “$x^5 + x^2 + 1$” resultará “$x^2 + x$” com resto “x”
  - … para fazer o polinômio divisível pelo fator basta subtrair o resto dele, contudo, o receptor será capaz de somente detectar o erro;
  - … então o mais indicado é usar o resto como um “checksum”.

… 3.2.2 – Códigos de Detecção de Erros

- "fator" - utilizado para gerar o "checksum", também é denominado Polinômio Gerador (Generator Polynomial) do código;
  - … multiplica-se o polinômio da primitiva por um termo igual à parcela de mais alto grau do polinômio gerador de modo que a primitiva seja deslocada para a esquerda tantos bits quantos do "checksum";
  - … então divide-se o polinômio da primitiva pelo polinômio gerador e subtrai-se o resto.

- … uma vez que o CRC é um código linear, todo padrão de erro E deve ser igual a alguma palavra código T;
  - … para um código conhecido esta propriedade pode ser usada para calcular a taxa residual de erro.

… 3.2.2 – Códigos de Detecção de Erros

- "Controle de Erro" - tem o objetivo de diminuir a taxa de erro de um determinado canal de comunicação;
  - … contudo, nem todos os erros podem ser detectados, então sempre existe uma 'Taxa de Erro Residual' (RER – "Residual Error Rate");

- Suponha-se que a probabilidade de erros de transmissão em uma msg. seja "p", e que o método de controle de erro identifique uma fração "f" deste conjunto de erros, então:
  - RER = p * (1 – f) … esta equação implica que, por instância, a taxa residual de erros é da ordem de 10^{-9} (10^-9) ou menos.

… 3.2.2 – Códigos de Detecção de Erros

- Seja P um polinômio primitiva e G um polinômio gerador de grau "r", então o resto da divisão é polinômio R com grau "r-1", onde:

$$R = P * x^r / G$$

- Palavra Código "$T = P * x^r – R$"

- Um erro de transmissão adiciona um polinômio E ao código … e o receptor descobre o erro por:

$$(T + E) / G = T/G + E/G = E/G$$

… 3.2.2 – Códigos de Detecção de Erros

- Um erro de comunicação é indetectável se E/G = 0
  - … se E é diferente de zero e de grau menor que G, a divisão sempre tem resto, ou seja, todas as rajadas menores ou iguais a “r” são detectáveis.
  - ... posição dentro da primitiva T “code word” onde ocorre a rajada de erro é irrelevante para detecção do erro;

- Padrão de Erro “E” não se transforma em um múltiplo de G pela simples multiplicação de um fator x^i (obviamente para $x^i \neq G$).
  - … como a ocorrência do erro é aleatória, então para um código de comprimento “n+r” bits é possível calcular a probabilidade de ocorrência de erro (independente da posição).

… 3.2.2 – Códigos de Detecção de Erros

- Existem “2^(n + r)” possibilidades de erros, sendo que o nro. de múltiplos inteiros de um polinômio gerador de grau “r” em uma palavra de código é “2 ^ n”
- … cada múltiplo pode ser considerado como uma soma finita de “n” fatores, no qual cada fator é obtido por um deslocamento à esquerda do polinômio gerador.
- $2^n / 2^{(n+r)} = 1 / 2^r$
- e.g., considere r = 16 $\rightarrow 10^{-5}$ de todos os erros.

… 3.2.2 – Códigos de Detecção de Erros

- Projetar polinômios geradores para detectar a maior classe possível de erros de comunicação não é uma “tarefa fácil”.
  - … um bom polinômio gerador tem pelo menos 01 fator (x + 1) e deve ser compartilhado com todos os envolvidos;
- CRC-12 é um polinômio largamente utilizado, para gerar soma verificação - “checksum” de grau 12

$$x^{12} + x^{11} + x^3 + x^2 + 1$$

  - … grau é 12, então este código pode detectar rajadas de erros de até 12 bits, não somente rajadas de 12 bits.

… 3.2.2 – Códigos de Detecção de Erros

- A CCITT (hoje ITU-T) recomenda polinômio gerador de grau 16.

$$x^{16} + x^{12} + x^5 + 1$$

  - … grau é 16, então este código pode detectar rajadas de erros de até 16 bits, não somente rajadas de 16 bits.

- Em aritmética binária, este código pode ser escrito como:

$$(x+1) * (x^{15} + x^{14} + x^{13} + x^{12} + x^4 + x^3 + x^2 + x + 1)$$

  - … pode-se ver que qualquer polinômio multiplicado pelo fator (x+1) tem um número par de parcelas (ie, bits não zero).
  - … verificar o resultado (nro par de parcelas) … !!!

… 3.2.2 – Códigos de Detecção de Erros

- Isto significa que todo “E” com um número ímpar de parcelas, produzido por um número ímpar de erros de bits é detectável;
- CRC-CCITT detecta
  - 100% em caso de 2 bits de erro;
  - 99,997% em caso de rajadas de 17 bits;
  - 99,998% em rajadas maiores que 17 bits.

- O polinômio utilizado pelo protocolo BSC (Binary Synchronous Communication) da IBM é muito próximo deste polinômio:

$$x^{16} + x^{15} + x^2 + 1$$

… 3.2.2 – Códigos de Detecção de Erros

- Comitê IEEE 802 padronizou um CRC-32 bits:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

- Codificação e Decodificação do “checksum” CRC exigem tempo e podem degradar o desempenho do autômato;
- … implementação é tipicamente feita em “hardware”, por motivos óbvios de necessidade de esforço computacional.

- Observe-se que o desempenho do algoritmos tem uma linearidade com o grau do polinômio gerador.

3.3 – Protocolos Elementares de Enlace de Dados

- “objetivo” - … para introduzir o estudo de protocolos, examina-remos 03 protocolos com grau de complexidade crescente.

- Neste contexto, é útil explicitar algumas suposições:
- … “host” A deseja enviar um longo fluxo de dados ao “host” B utilizando um serviço confiável e orientado a conexões;
- … há processos independentes que se comunicam pelo envio de msgs. nas camadas física, de enlace e de redes;
- “hosts” não sofrerão panes, isto é, esses protocolos lidam com erros de comunicação, mas não com os problemas causados por “hosts” que sofrem panes e são reinicializados.

… 3.3 – Protocolos Elementares de Enlace de Dados

- … quando a camada de enlace de dados aceita um pacote, ela o encapsula um quadro, acrescentando-lhe um cabeçalho e um "trailer" de enlace de dados (Fig. 3.11).

```
#define MAX_PKT 1024                                    /* determines packet size in bytes */

typedef enum {false, true} boolean;                     /* boolean type */
typedef unsigned int seq_nr;                            /* sequence or ack numbers */
typedef struct {unsigned char data[MAX_PKT];} packet;   /* packet definition */
typedef enum {data, ack, nak} frame_kind;               /* frame_kind definition */

typedef struct {                                        /* frames are transported in this layer */
  frame_kind kind;                                      /* what kind of frame is it? */
  seq_nr seq;                                           /* sequence number */
  seq_nr ack;                                           /* acknowledgement number */
  packet info;                                          /* the network layer packet */
} frame;
```

Figure 3-11. Some definitions needed in the protocols to follow - *protocol.h*.

… 3.3 – Protocolos Elementares de Enlace de Dados

- “premissa” - … inicialmente o receptor aguarda algo acontecer, ou seja, a camada de enlace está esperando algo acontecer através de uma chamada de procedimento “wait_for_event(&event)”

```
/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);
```

Figure 3-11. Some definitions needed in the protocols to follow - *protocol.h*.

… 3.3 – Protocolos Elementares de Enlace de Dados

- "premissa" - … procedimentos de biblioteca "to_network_layer" possibilitam a recepção de um pacote e "from_network_layer" possibilitam o envio de um pacote.

```
/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);
```

Figure 3-11. Some definitions needed in the protocols to follow - *protocol.h*.

… 3.3 – Protocolos Elementares de Enlace de Dados

- "premissa" - … procedimentos de biblioteca "to_physical_layer" para transmitir um quadro e "from_physical_layer" para receber um quadro e armazená-lo no "frame" "r".

```
/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);
```

Figure 3-11. Some definitions needed in the protocols to follow - *protocol.h*.

… 3.3 – Protocolos Elementares de Enlace de Dados

- … ignoraremos todos os detalhes de atividades paralelas na camada de enlace de dados, e presumiremos que ela se dedica em tempo integral apenas ao tratamento do nosso canal.

```
/* Start an auxiliary timer and enable the ack_timeout event. */
void start_ack_timer(void);

/* Stop the auxiliary timer and disable the ack_timeout event. */
void stop_ack_timer(void);

/* Allow the network layer to cause a network_layer_ready event. */
void enable_network_layer(void);

/* Forbid the network layer from causing a network_layer_ready event. */
void disable_network_layer(void);

/* Macro inc is expanded in-line: increment k circularly. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

Figure 3-11. Some definitions needed in the protocols to follow - *protocol.h*.

… 3.3 – Protocolos Elementares de Enlace de Dados

- … quando um “frame” chega no receptor, o campo “checksum” é recalculado e, se incorreto, a camada de enlace é informada → “event = ckcum_err”
- … se o “frame” não contiver erros, a camada de enlace é informada → “event = frame_arrival” … e então a camada de enlace aceita o “frame” para inspeção;
- … na sequência, repassa uma porção do “frame” para a camada de rede, ou seja, sob quaisquer circunstâncias o cabeçalho do “frame” jamais é repassado para a camada de rede;
- “frame” - composto de “kind”, “seq” e “ack” contendo informações de controle e “info” contém dado atual a ser transportado. … 04 elementos são coletivamente chamados de “header”.

… 3.3 – Protocolos Elementares de Enlace de Dados

- … procedimentos “start_ack_timer” e “stop_ack_timer” controlam um “timer” auxiliar usado para gerar reconhecimento sob condições especiais;
- … procedimentos “enable_network_layer” e “disable_network_ layer” são usados nos protocolos mais sofisticados, para os quais não mais supomos que a camada de rede sempre terá pacotes;
- … quando a camada de enlace de dados habilita a camada de rede, esta passa a ter permissão para causar uma interrupção sempre que tiver um pacote para enviar.

… 3.3 – Protocolos Elementares de Enlace de Dados

- … nros. de seqüência dos quadros estão sempre na faixa de 0 a MAX_SEQ (inclusive), onde MAX_SEQ tem um valor diferente para os diversos protocolos;
- … com freqüência, é necessário aumentar um número de seqüência em uma unidade, de forma circular.

- Declarações da Fig. 3.11 fazem parte de cada um dos protocolos apresentados a seguir – são apresentadas em conjunto para economizar espaço e facilitar a consulta.

3.3.1 – Protocolo Simplex sem Restrições

- “Protocolo Utopia” - oferece transmissão de dados em um único sentido, ou seja, do transmissor para o receptor.
- … canal de comunicação é livre de erros e que o receptor é capaz de processar toda a entrada de uma forma infinitamente rápida.
- … protocolo absolutamente imaginário, que denominaremos "utopia", está descrito na Fig. 3.12.

… 3.3.1 – Protocolo Simplex sem Restrições

- … procedimento transmissor é executado na camada de enlace de dados do "host" origem, enquanto o receptor na camada de enlace do "host" destino.

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
    frame s;                          /* buffer for an outbound frame */
    packet buffer;                    /* buffer for an outbound packet */

    while (true) {
        from_network_layer(&buffer);  /* go get something to send */
        s.info = buffer;              /* copy it into s for transmission */
        to_physical_layer(&s);        /* send it on its way */
    }                                 /* Tomorrow, Creeps in this petty pace from day to day
                                         To the last syllable of recorded time. – Macbeth, V, v */
}
```

Figure 3-12. A utopian simplex protocol.

… 3.3.1 – Protocolo Simplex sem Restrições

- … transmissor é um “loop” infinito que envia os dados o mais rápido possível ... busca um pacote da camada de rede; cria um quadro utilizando a variável “s” e transmite o quadro ao destino.

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
    frame s;                          /* buffer for an outbound frame */
    packet buffer;                    /* buffer for an outbound packet */

    while (true) {
        from_network_layer(&buffer);  /* go get something to send */
        s.info = buffer;              /* copy it into s for transmission */
        to_physical_layer(&s);        /* send it on its way */
    }                                 /* Tomorrow, Creeps in this petty pace from day to day
                                         To the last syllable of recorded time. – Macbeth, V, v */
}
```

Figure 3-12. A utopian simplex protocol.

… 3.3.1 – Protocolo Simplex sem Restrições

- … camadas de rede do transmissor e do receptor estão sempre prontas à espera de informações; ... tempo de processamento pode ser ignorado; … espaço disponível em buffer é infinito.

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void receiver1(void)
{
    frame r;
    event_type event;                    /* filled in by wait, but not used here */

    while (true) {
        wait_for_event(&event);          /* only possibility is frame_arrival */
        from_physical_layer(&r);         /* go get the inbound frame */
        to_network_layer(&r.info);       /* pass the data to the network layer */
    }
}
```

Figure 3-12. A utopian simplex protocol.

3.3.2 – Protocolo Stop-and-Wait

- “premissa” - … continuamos supondo que o canal de comunicação é “simplex” e não apresenta erros (canal sem erros).
- … “host” A deseja enviar um longo fluxo de dados ao “host” B utilizando um serviço confiável e orientado a conexões.

- “problema” - .. como impedir que o transmissor inunde o receptor, mais rapidamente do que este é capaz de processar..
  - … se o receptor necessitar de um tempo “t” para executar “from_physical _layer” e “to_network_layer”, o transmissor terá de enviar os dados em uma velocidade média menor que um quadro por tempo “t”.

… 3.3.2 – Protocolo Stop-and-Wait

- “problema” - .. como impedir que o transmissor inunde o receptor, mais rapidamente do que este é capaz de processar..
  - … se o receptor necessitar de um tempo “t” para executar “from_physical _layer” e “to_network_layer”, o transmissor terá de enviar os dados em uma velocidade média menor que um quadro por tempo “t”.

- “solução” - receptor enviar um “feedback” ao transmissor, permitindo a transmissão do próximo quadro.
  - … após envio de um quadro, o protocolo exige que o transmissor espere sua vez, até a chegada do pequeno quadro fictício (confirmação).
  - … como fica a premissa do canal ser simplex, ou seja, “host” A deseja enviar um longo fluxo de dados ao “host” B ??

… 3.3.2 – Protocolo Stop-and-Wait

- “stop-and-wait” - … protocolos nos quais o transmissor envia um quadro e em seguida espera por uma confirmação antes de continuar sua operação são chamados “stop-and-wait”.
  - … embora o tráfego de dados neste exemplo seja “simplex”, ou seja, do transmissor para o receptor, há quadros em ambas as direções.
  - … canal de comunicação entre as duas camadas de enlace deve ser capaz de realizar a transferência bidirecional de informações.

- “stop-and-wait” - … protocolo com rígida alternância de fluxo, ou seja, transmissor envia um quadro e na sequência receptor envia outro; e o ciclo se repete na mesma ordem.

… 3.3.2 – Protocolo Stop-and-Wait

- … enlace de dados do transmissor não precisa sequer inspecionar o quadro recebido, pois só há uma possibilidade - quadro recebido é sempre uma confirmação.

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
    frame s;                        /* buffer for an outbound frame */
    packet buffer;                  /* buffer for an outbound packet */
    event_type event;               /* frame_arrival is the only possibility */

    while (true) {
        from_network_layer(&buffer);    /* go get something to send */
        s.info = buffer;                /* copy it into s for transmission */
        to_physical_layer(&s);          /* bye-bye little frame */
        wait_for_event(&event);         /* do not proceed until given the go ahead */
    }
}
```

Figure 3-13. A simplex stop-and-wait protocol.

… 3.3.2 – Protocolo Stop-and-Wait

- " receptor1" ≠ "receptor2" - … após entregar um pacote à camada de rede, o receptor2 envia um quadro de confirmação de volta ao transmissor, antes de entrar mais uma vez no "loop" de espera.

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void receiver2(void)
{
    frame r, s;                         /* buffers for frames */
    event_type event;                   /* frame_arrival is the only possibility */
    while (true) {
        wait_for_event(&event);         /* only possibility is frame_arrival */
        from_physical_layer(&r);        /* go get the inbound frame */
        to_network_layer(&r.info);      /* pass the data to the network layer */
        to_physical_layer(&s);          /* send a dummy frame to awaken sender */
    }
}
```

Figure 3-13. A simplex stop-and-wait protocol.

3.3.3 – Stop-and-Wait + Canal com Ruído

- “premissa” - canal de comunicação é passível de erros, além disso quadros podem ser danificados ou perdidos.
  - … supõe-se que se um quadro for danificado, o “hardware” receptor detectará essa ocorrência ao calcular o soma verificação.

- “solução” - … seja uma variação do protocolo 2, p.ex., com a inclusão de um “timer” - temporizado.
  - … transmissor envia um quadro, mas o receptor retorna um quadro de confirmação se os dados são recebidos corretamente;
  - … se um quadro danificado chegar ao receptor, ele seria descartado e após um certo tempo, o transmissor alcança seu “timeout” e retransmite.
  - … processo é repetido até que o quadro chegue intacto.

… 3.3.3 – Stop-and-Wait + Canal com Ruído

- “solução” - … vejamos se é viável uma variação do protocolo 2, por exemplo, com a inclusão de um “timer”.
  - … transmissor envia um quadro, mas o receptor retorna um quadro de confirmação se os dados são recebidos corretamente;
  - … se um quadro danificado chegar ao receptor, ele seria descartado e após um certo tempo, o transmissor alcança seu “timeout” e retransmite.
  - … processo é repetido até que o quadro chegue intacto.

- Por que a inclusão do “timer” não funciona !?

… 3.3.3 – Stop-and-Wait + Canal com Ruído

- e.g., considere o cenário / situação:
- … camada de rede de A envia o pacote 1 à sua camada de enlace de dados, pacote é corretamente recebido em B e repassado à camada de rede de B, assim, B envia uma confirmação para A;
- … quadro de confirmação (B → A) se perde por completo - simplesmente nunca chega ao destino.
- … resultado é que todo o processo de troca de dados simplesmente para de funcionar !!

- Obs.: Tudo seria muito mais simples se o canal tivesse adulterado e perdido apenas quadros de dados, e não de controle.

... 3.3.3 – Stop-and-Wait + Canal com Ruído

- e.g., (cont.) considere o cenário / situação:
- … eventualmente, camada de enlace de dados de A tem seu limite de tempo esgotado e como não recebeu uma confirmação, ela presume que seu quadro de dados se perdeu e retransmite;
- … quadro duplicado chega perfeitamente à camada de enlace de dados de B e é repassado de imediato, sem maiores problemas, à camada de rede.

- Obs.: Se A envia um arquivo para B → partes do arquivo serão duplicadas → sistema falha !!

… 3.3.3 – Stop-and-Wait + Canal com Ruído

- “constatação” - … receptor precisa distinguir entre um quadro que ele está recebendo pela 1ª vez de uma retransmissão.

- “solução” - … fazer o transmissor incluir um nro. de seqüência no cabeçalho de cada quadro enviado.
- … receptor poderá verificar o nro. de seqüência de cada quadro recebido para confirmar se é um novo ou se é uma duplicata.

… 3.3.3 – Stop-and-Wait + Canal com Ruído

- … nro. de seqüência de 1 bit (0 ou 1) … a cada instante, o receptor espera o próximo número de seqüência.

```
#define MAX_SEQ 1                                   /* must be 1 for protocol 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
    seq_nr next_frame_to_send;                      /* seq number of next outgoing frame */
    frame s;                                        /* scratch variable */
    packet buffer;                                  /* buffer for an outbound packet */
    event_type event;

    next_frame_to_send = 0;                         /* initialize outbound sequence numbers */
    from_network_layer(&buffer);                    /* fetch first packet */
    . . .
}
```

Figure 3-14. A positive acknowledgement with retransmission protocol.

... 3.3.3 – Stop-and-Wait + Canal com Ruído

```
#define MAX_SEQ 1                                   /* must be 1 for protocol 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
    . . .
    while (true) {
        s.info = buffer;                            /* construct a frame for transmission */
        s.seq = next_frame_to_send;                 /* insert sequence number in frame */
        to_physical_layer(&s);                      /* send it on its way */
        start_timer(s.seq);                         /* if answer takes too long, time out */
        wait_for_event(&event);                     /* frame_arrival, cksum_err, timeout */
        if (event == frame_arrival) {
            from_physical_layer(&s);                /* get the acknowledgement */
            if (s.ack == next_frame_to_send) {
                stop_timer(s.ack);                  /* turn the timer off */
                from_network_layer(&buffer);        /* get the next one to send */
                inc(next_frame_to_send);            /* invert next_frame_to_send */
            }
        }
    }
}
```

Figure 3-14. A positive acknowledgement with retransmission protocol.

… 3.3.3 – Stop-and-Wait + Canal com Ruído

- “Positive Acknowledgment with Retransmission” ou “PAR” … protocolos nos quais o transmissor espera por uma confirmação positiva antes de passar para o próximo item de dados;
  - … são também denominados “Automatic Repeat reQuest” ou “ARQ” … solicitação de repetição automática.

- … após enviar um quadro, o transmissor ativa o “timer” ou reinicializa o “timer” para permitir a contagem de outro intervalo.
- … intervalo deve ser definido de forma que haja tempo suficiente para o quadro chegar ao receptor e ser processado e para o quadro de confirmação ser enviado de volta ao transmissor.

… 3.3.3 – Stop-and-Wait + Canal com Ruído

```
void receiver3(void)
{
    seq_nr frame_expected;
    frame r, s;
    event_type event;

    frame_expected = 0;
    while (true) {
        wait_for_event(&event);                 /* possibilities: frame_arrival, cksum_err */
        if (event == frame_arrival) {           /* a valid frame has arrived */
            from_physical_layer(&r);            /* go get the newly arrived frame */
            if (r.seq == frame_expected) {      /* this is what we have been waiting for */
                to_network_layer(&r.info);      /* pass the data to the network layer */
                inc(frame_expected);            /* next time expect the other sequence nr */
            }
            s.ack = 1 – frame_expected;         /* tell which frame is being acked */
            to_physical_layer(&s);              /* send acknowledgement */
        }
    }
}
```

Figure 3-14. A positive acknowledgement with retransmission protocol.

3.4 – Protocolo de Janela Deslizante

- “Protocolo Stop-and-Wait” … assume como premissa que os quadros de dados são transmitidos em apenas um sentido;
  - … em situações reais, há necessidade de transmissão nos 02 sentidos.

- “transmissão full-duplex” - … 02 canais de comunicação distintos e uso de cada um para um tráfego de dados “simplex”;
  - … tem-se 02 circuitos físicos separados, cada um com um canal "direto" (para dados) e um canal "inverso" (para confirmações), mas neste caso a largura de banda do canal inverso é quase totalmente perdida;
  - … na verdade, o usuário paga por 02 circuitos, mas usa apenas a capacidade para dados de 01 deles.

3.4 – Protocolo de Janela Deslizante

- “solução” - … usar o mesmo circuito para dados em ambos os sentidos, ou seja, quadros de dados enviados de A para B são misturados com os quadros de confirmação A para B.

- “alternativa” - … retardar temporariamente as confirmações e enviá-las junto com o próximo quadro de dados - “piggybacking”.
  - … vantagem do piggybacking em relação ao envio de quadros ACKs distintos é a melhor utilização da largura de banda disponível para o canal.
  - … campo “ack” do cabeçalho de quadro precisa de apenas alguns bits, enquanto um quadro separado precisa de um cabeçalho, da confirmação e de um campo soma verificação.

… 3.4 – Protocolo de Janela Deslizante

- “problema” - … quanto tempo a camada de enlace deve esperar por um pacote ao qual deverá acrescentar a confirmação ?
  - … se a camada de enlace esperar durante um intervalo de tempo maior que o permitido pelo “timeout”, o quadro será retransmitido, o que invalidará todo o processo de confirmação !?

- “solução” - … se um novo pacote chegar da camada de rede, a confirmação será acrescentada ao pacote.
  - … caso contrário, se nenhum pacote tiver chegado até o final desse intervalo de tempo, a camada de enlace de dados simplesmente enviará um quadro de confirmação em separado.

… 3.4 – Protocolo de Janela Deslizante

- “janela deslizante” - tamanho 1, com um nro. de seqüência de 3 bits, permitindo até 08 quadros.

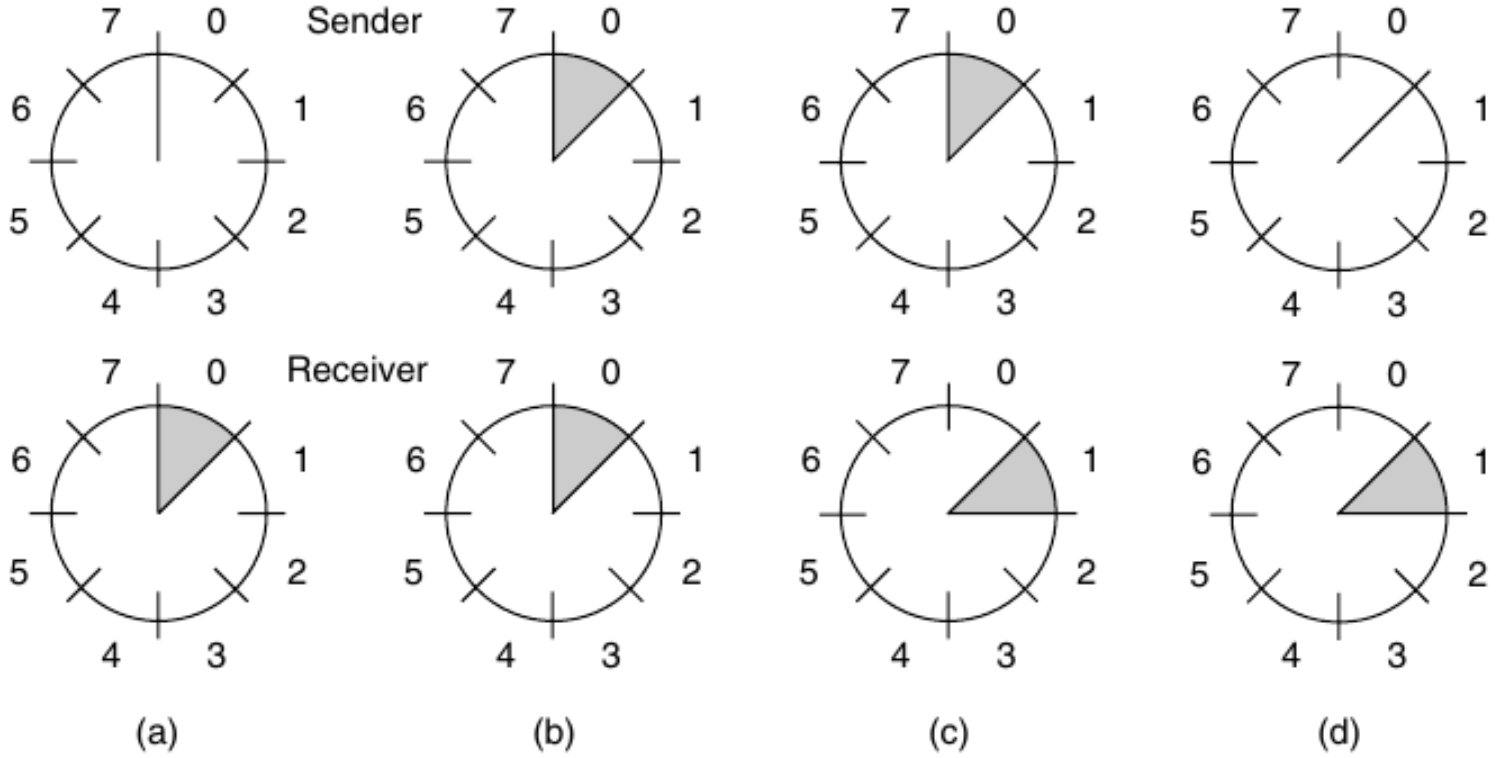


Figure 3-15. A sliding window of size 1, with a 3-bit sequence number. (a) Initially. (b) After the first frame has been sent. (c) After the first frame has been received. (d) After the first acknowledgement has been received.

3.4.1 – Protocolo de Janela Deslizante de 1 Bit

- ... vamos examinar primeiro um protocolo de janela deslizante com um tamanho máximo de janela = 1;
  - … este protocolo utiliza o stop-and-wait, pois o transmissor envia um quadro e aguarda sua confirmação antes de enviar o quadro seguinte.

- Fig. 3.16 representa esse tipo de protocolo - … como os demais, esse protocolo começa definindo algumas variáveis.
  - “next_frame_to_send” - informa o próximo quadro que TX enviará;
  - “frame_expected” - quadro que o receptor está esperando.

… 3.4.1 – Protocolo de Janela Deslizante de 1 Bit

```
/* Protocol 4 (Sliding window) is bidirectional. */

#define MAX_SEQ 1                                   /* must be 1 for protocol 4 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void protocol4 (void)
{
    seq_nr next_frame_to_send;                      /* 0 or 1 only */
    seq_nr frame_expected;                          /* 0 or 1 only */
    frame r, s;                                     /* scratch variables */
    packet buffer;                                  /* current packet being sent */
    event_type event;

    next_frame_to_send = 0;                         /* next frame on the outbound stream */
    frame_expected = 0;                             /* frame expected next */
    from_network_layer(&buffer);                    /* fetch a packet from the network layer */
    s.info = buffer;                                /* prepare to send the initial frame */
    s.seq = next_frame_to_send;                     /* insert sequence number into frame */
    s.ack = 1 – frame_expected;                     /* piggybacked ack */
    to_physical_layer(&s);                          /* transmit the frame */
    start_timer(s.seq);                             /* start the timer running */
```

… 3.4.1 – Protocolo de Janela Deslizante de 1 Bit

```
while (true) {
    wait_for_event(&event);                  /* frame_arrival, cksum_err, or timeout */
    if (event == frame_arrival) {            /* a frame has arrived undamaged */
        from_physical_layer(&r);             /* go get it */
        if (r.seq == frame_expected) {       /* handle inbound frame stream */
            to_network_layer(&r.info);       /* pass packet to network layer */
            inc(frame_expected);             /* invert seq number expected next */
        }
        if (r.ack == next_frame_to_send) {   /* handle outbound frame stream */
            stop_timer(r.ack);               /* turn the timer off */
            from_network_layer(&buffer);     /* fetch new pkt from network layer */
            inc(next_frame_to_send);         /* invert sender's sequence number */
        }
    }
    s.info = buffer;                         /* construct outbound frame */
    s.seq = next_frame_to_send;              /* insert sequence number into it */
    s.ack = 1 – frame_expected;              /* seq number of last received frame */
    to_physical_layer(&s);                   /* transmit a frame */
    start_timer(s.seq);                      /* start the timer running */
}
}
```

Figure 3-16. A 1-bit sliding window protocol.

… 3.4.1 – Protocolo de Janela Deslizante de 1 Bit

- e.g., … suponha que A esteja tentando enviar seu quadro “0” ao B e que B esteja tentando enviar seu quadro “0” ao A;
  - … imagine que A envia um quadro a B, mas o intervalo de “timeout” de A é curto demais, então A pode completar o “timeout” repetidas vezes, enviando uma série de quadros idênticos, todos com seq = 0 e ack = 1.
  - … quando o 1º quadro válido chegar a B, ele será aceito, e “frame_expected” será definido como “1”.
  - … todos os quadro “s” subseqüentes serão rejeitados, porque B agora está esperando quadros com número de seqüência “1”, e não “0”.
  - … além disso, como todas as cópias têm ack = 1 e B ainda está aguardando uma confirmação de “0”, B não buscará um novo pacote em sua camada de rede.

… 3.4.1 – Protocolo de Janela Deslizante de 1 Bit

- (cont.) e.g., … suponha que A esteja tentando enviar seu quadro “0” ao B e que B esteja tentando enviar seu quadro “0” ao A;
  - … após a chegada de todas as cópias rejeitadas, B enviará um quadro para A contendo seq = 0 e ack = 0.
  - … eventualmente, um desses quadros chegará sem erros à máquina A, fazendo com que A comece a enviar o próximo pacote.
  - … nenhuma combinação de quadros perdidos ou “timeouts” prematuros pode fazer o protocolo entregar pacotes duplicados à camada de rede, ignorar um pacote ou chegar a um impasse.

… 3.4.1 – Protocolo de Janela Deslizante de 1 Bit

- Se B espera pelo 1º quadro de A antes de enviar um de seus quadros, a seqüência será (a) → todos quadros serão aceitos.

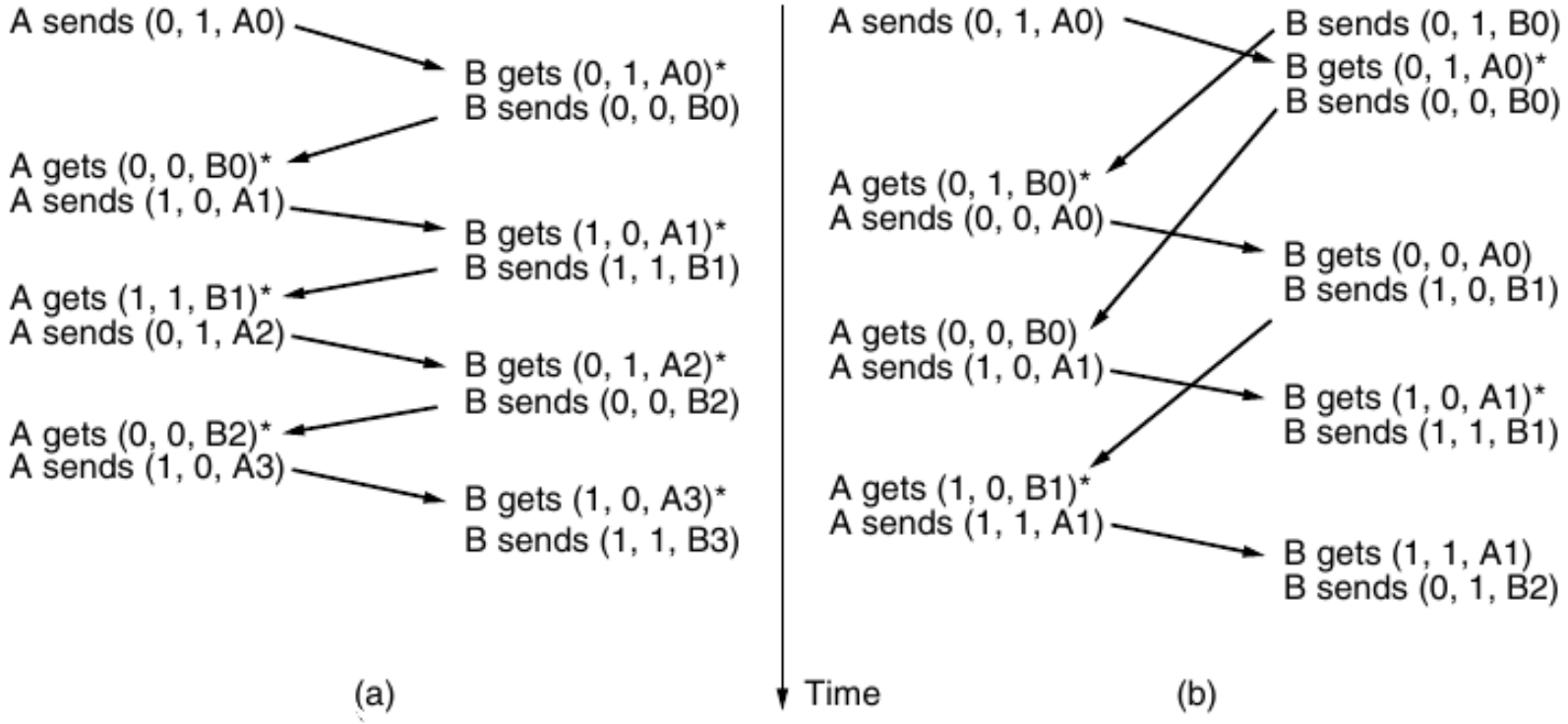


Figure 3-17. Two scenarios for protocol 4. (a) Normal case. (b) Abnormal case. The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

… 3.4.1 – Protocolo de Janela Deslizante de 1 Bit

- ”Situação Peculiar” - … ambos os lados enviam simultaneamente um pacote inicial → dificuldade de sincronização.

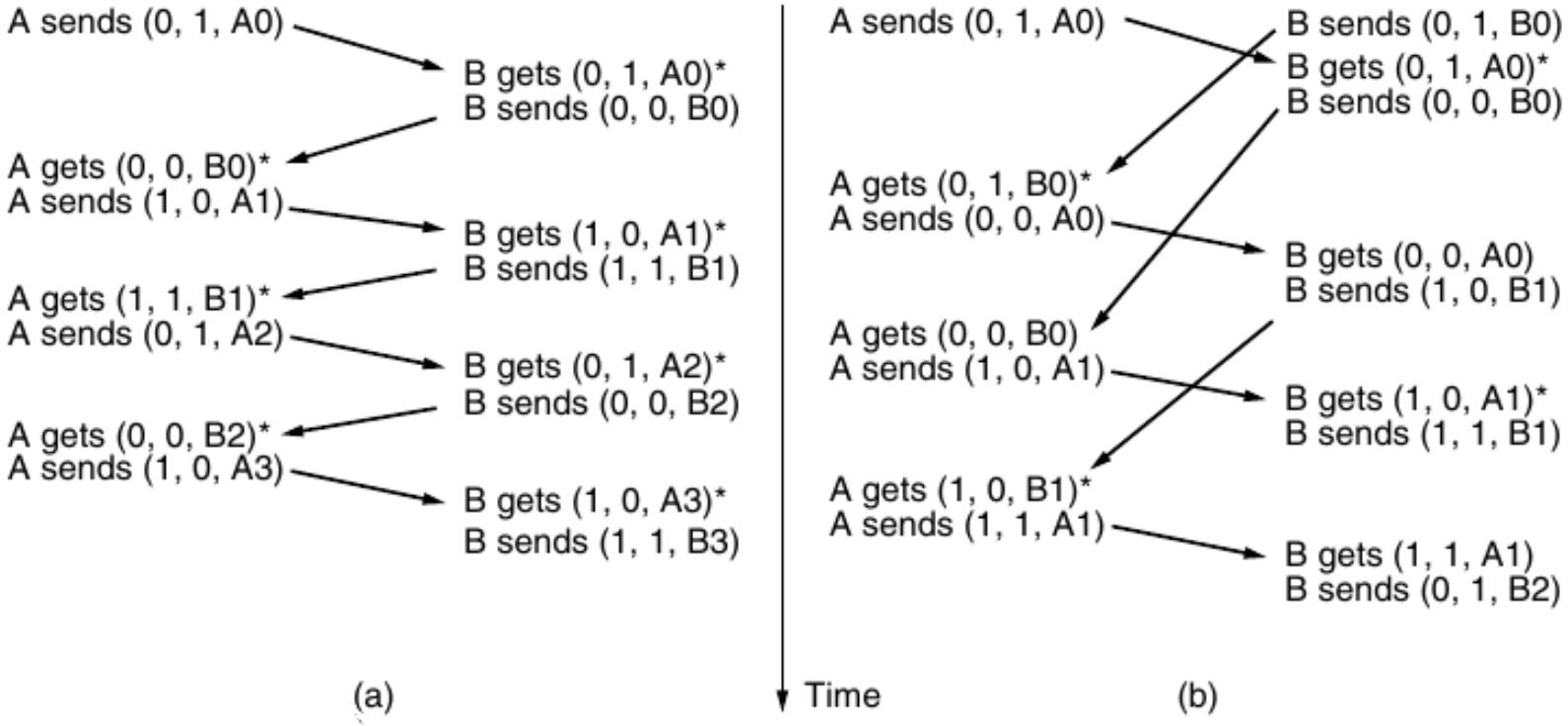


Figure 3-17. Two scenarios for protocol 4. (a) Normal case. (b) Abnormal case. The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

… 3.4.1 – Protocolo de Janela Deslizante de 1 Bit

- Se A e B iniciarem a comunicação ao mesmo tempo, seus 1os quadros se cruzarão e as camadas de enlace recairão em (b)

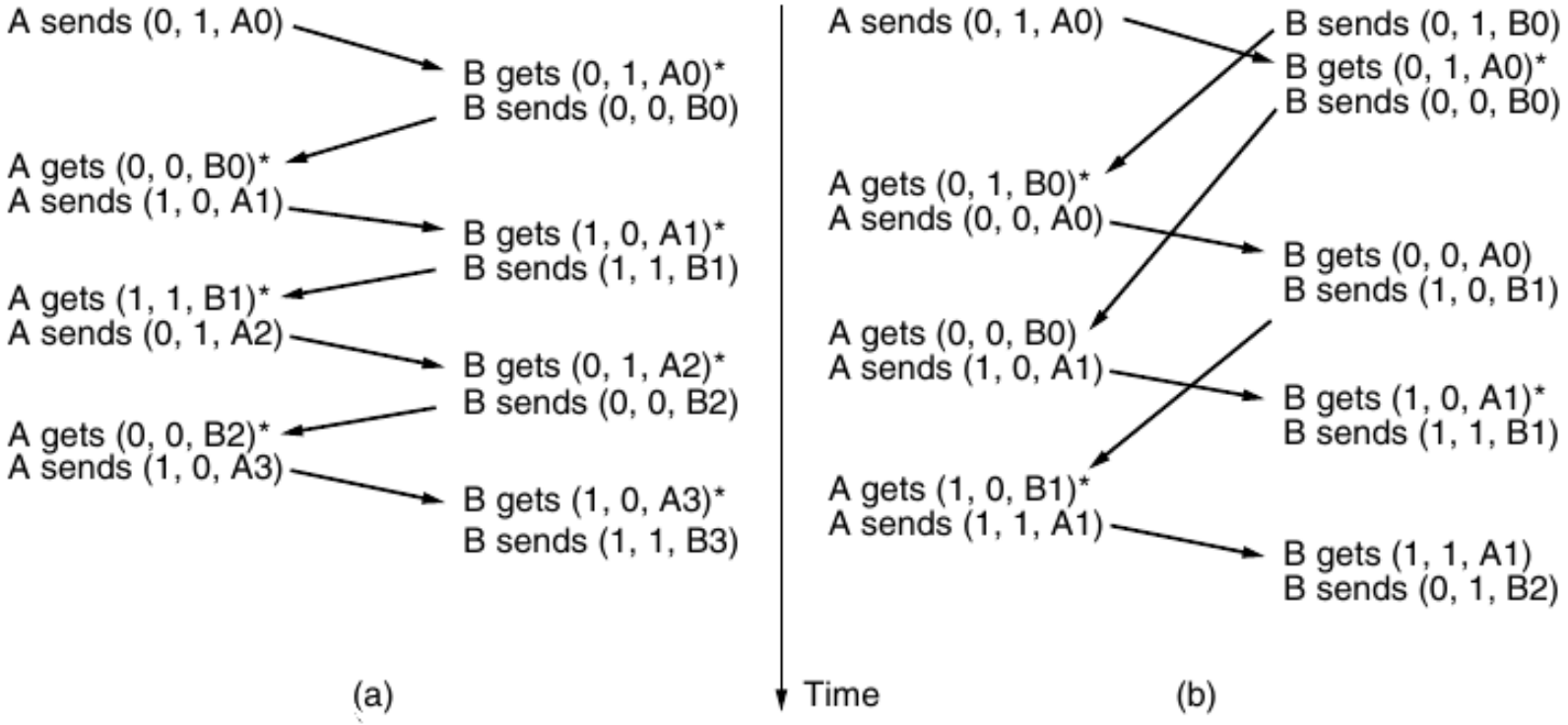


Figure 3-17. Two scenarios for protocol 4. (a) Normal case. (b) Abnormal case. The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

… 3.4.1 – Protocolo de Janela Deslizante de 1 Bit

- Na situação anormal de (b), 1/2 dos quadros contém cópias, embora não haja erros de transmissão.

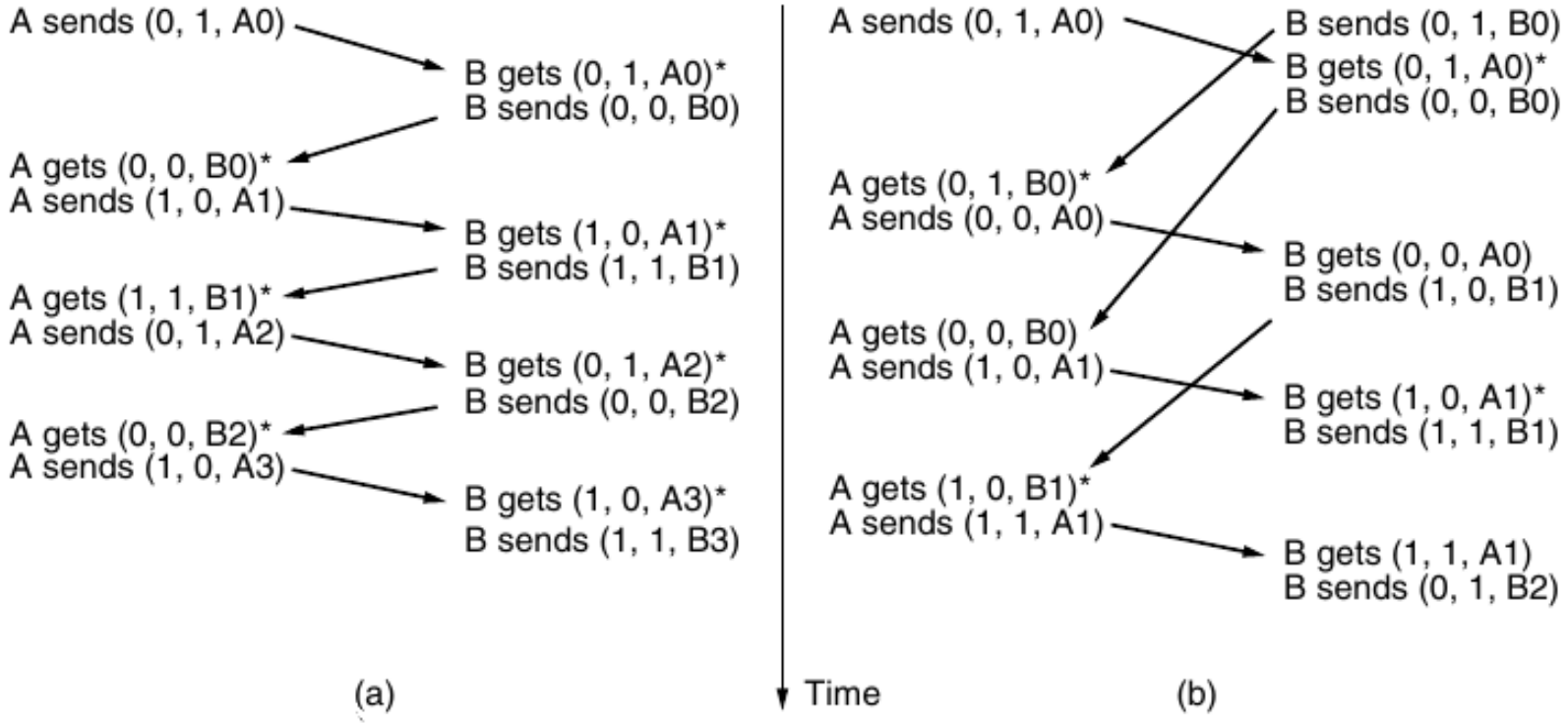


Figure 3-17. Two scenarios for protocol 4. (a) Normal case. (b) Abnormal case. The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

… 3.4.1 – Protocolo de Janela Deslizante de 1 Bit

- Situações similares podem ocorrer com “timeouts” prematuros, mesmo quando está claro que um lado começa primeiro.

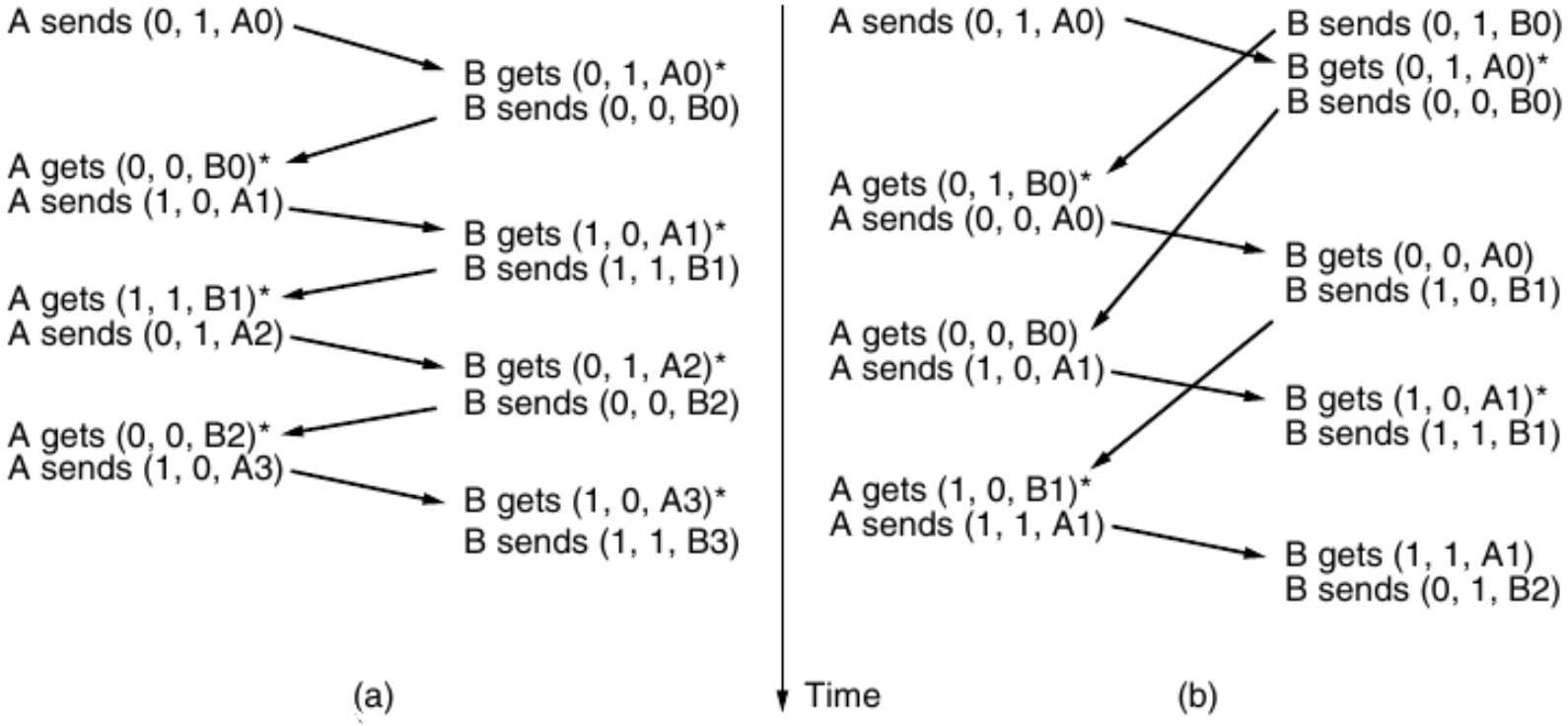


Figure 3-17. Two scenarios for protocol 4. (a) Normal case. (b) Abnormal case. The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

3.4.2 – Protocolo Go-back-N

- “Janela Deslizante” … “suposição” - … tempo de transmissão para a chegada de um quadro até o receptor + tempo de transmissão para o retorno da confirmação é insignificante.
  - … esta suposição é, as vezes, nitidamente falsa !!

- Necessidade de uma janela grande do lado transmissor surge sempre que o produto da largura de banda * RTT é grande.
  - … se largura de banda for alta, mesmo para um retardo moderado, o transmissor esgotará sua janela rapidamente;
  - … se retardo for alto, o transmissor irá esgotar sua janela até mesmo no caso de uma largura de banda moderada.

… 3.4.2 – Protocolo Go-back-N

- “largura de banda” * “RTT” - ... capacidade do canal
  - … então cabe ao transmissor a capacidade de preencher a janela sem interrupções, a fim de operar com eficiência máxima - “pipelining”.

- e.g., … se a capacidade do canal for “R” bps, se o tamanho do quadro for “L” bits e o tempo de propagação de ida e volta for RTT seg. → transmissão de um único quadro será L/R seg.
  - … depois do último bit de um quadro tiver sido enviado, haverá um retardo RTT/2 antes desse bit chegar ao receptor, e outro retardo de RTT/2 até o recebimento da confirmação, totalizando um retardo igual a RTT.
  - … no protocolo stop-and-wait, a linha está ocupada durante o tempo “L/R” e está ociosa durante todo o tempo de RTT, o que resulta em L / (L + R*RTT) de utilização do canal de dados.

… 3.4.2 – Protocolo Go-back-N

- Fig. 3.18 - “pipelining” e recuperação de erros - efeito de um erro quando (a) tamanho da janela receptora é igual a 1.

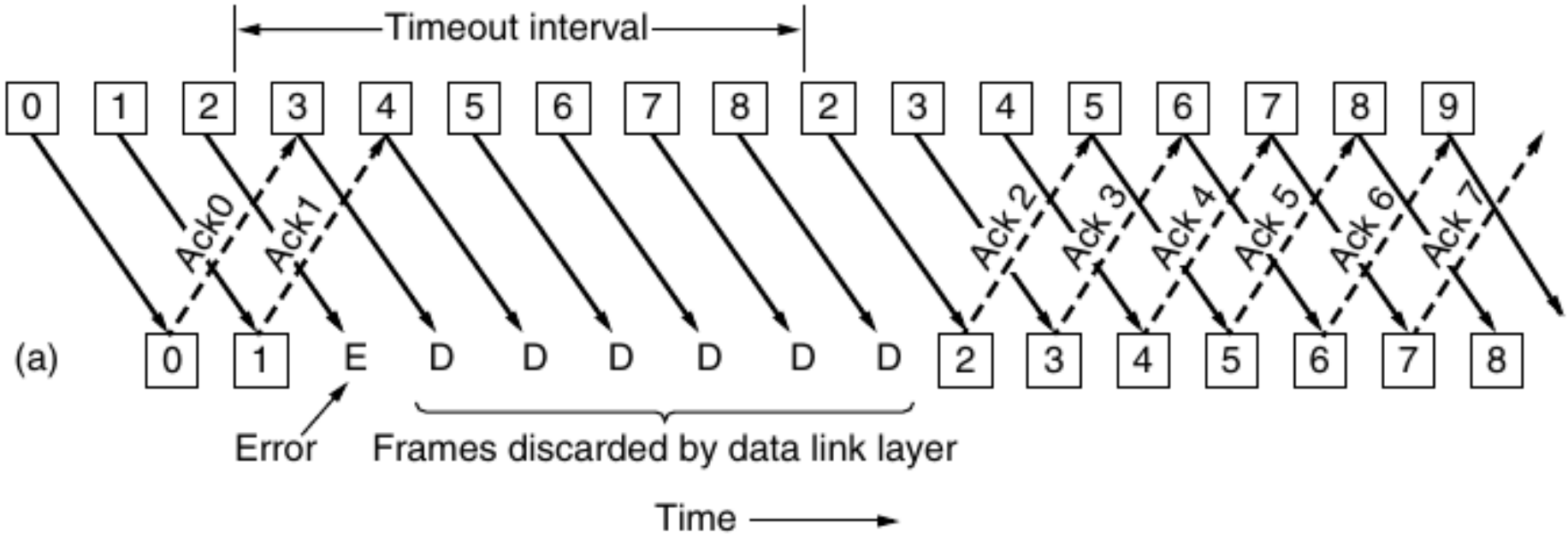


Figure 3-18. Pipelining and error recovery. Effect of an error when (a) receiver’s window size is 1 and (b) receiver’s window size is large.

… 3.4.2 – Protocolo Go-back-N

- Fig. 3.18 - “pipelining” e recuperação de erros - efeito de um erro quando (b) tamanho da janela receptora é grande.

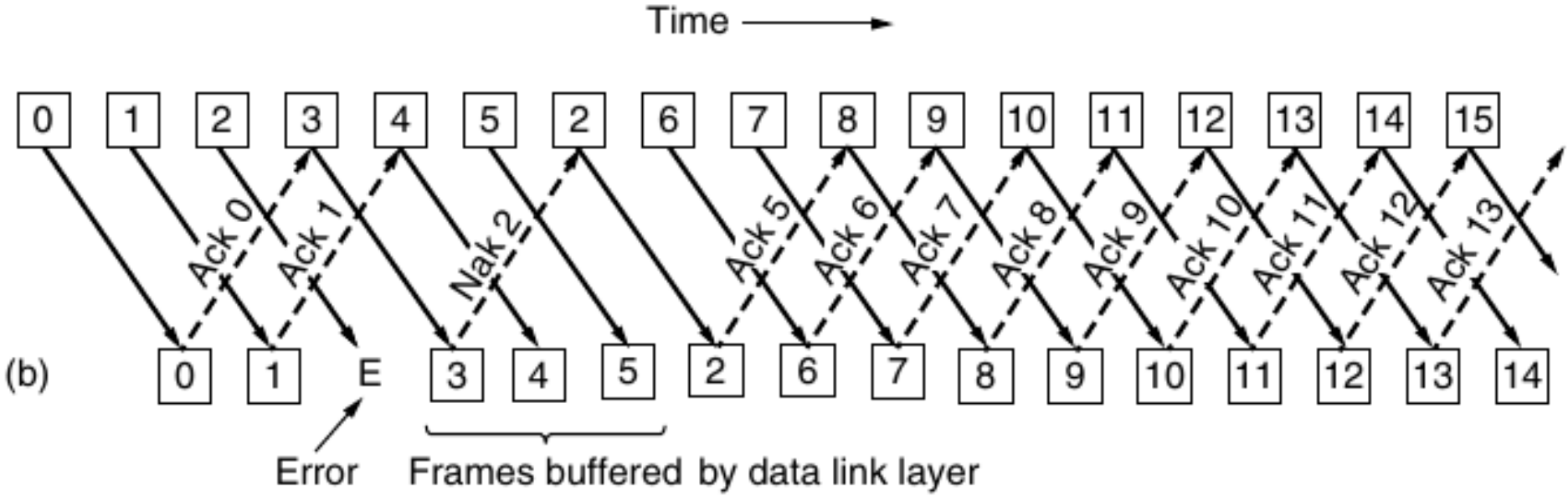


Figure 3-18. Pipelining and error recovery. Effect of an error when (a) receiver’s window size is 1 and (b) receiver’s window size is large.

… 3.4.2 – Protocolo Go-back-N

- Fig. 3.17 ... protocolo de pipelining no qual a camada de enlace receptora aceita apenas quadros em ordem.

```
/* Protocol 5 (Go-back-n) allows multiple outstanding frames. The sender may transmit up
   to MAX_SEQ frames without waiting for an ack. In addition, unlike in the previous
   protocols, the network layer is not assumed to have a new packet all the time. Instead,
   the network layer causes a network_layer_ready event when there is a packet to send. */

#define MAX_SEQ 7
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
  /* Return true if a <= b < c circularly; false otherwise. */
  if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
     return(true);
  else
     return(false);
}
```

… 3.4.2 – Protocolo Go-back-N

- Fig. 3.17 ... protocolo de pipelining no qual a camada de enlace receptora aceita apenas quadros em ordem;
  - … quadros que vierem depois de um quadro com erro são descartados.

```
static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[ ])
{
  /* Construct and send a data frame. */
  frame s;                                        /* scratch variable */

  s.info = buffer[frame_nr];                      /* insert packet into frame */
  s.seq = frame_nr;                               /* insert sequence number into frame */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);/* piggyback ack */
  to_physical_layer(&s);                          /* transmit the frame */
  start_timer(frame_nr);                          /* start the timer running */
}
```

… 3.4.2 – Protocolo Go-back-N

- Observe que no máximo MAX_SEQ quadros, e não MAX_SEQ + 1, mesmo que haja 0, 1, 2,..., MAX_SEQ. nros de seqüência.

```
void protocol5(void)
{
  seq_nr next_frame_to_send;          /* MAX_SEQ > 1; used for outbound stream */
  seq_nr ack_expected;                /* oldest frame as yet unacknowledged */
  seq_nr frame_expected;              /* next frame expected on inbound stream */
  frame r;                            /* scratch variable */
  packet buffer[MAX_SEQ + 1];         /* buffers for the outbound stream */
  seq_nr nbuffered;                   /* number of output buffers currently in use */
  seq_nr i;                           /* used to index into the buffer array */
  event_type event;

  enable_network_layer();                  /* allow network_layer_ready events */
  ack_expected = 0;                        /* next ack expected inbound */
  next_frame_to_send = 0;                  /* next frame going out */
  frame_expected = 0;                      /* number of frame expected inbound */
  nbuffered = 0;                           /* initially no packets are buffered */
  ... ...
```

… 3.4.2 – Protocolo Go-back-N

```
while (true) {
    wait_for_event(&event);                          /* four possibilities: see event_type above */
    switch(event) {
        case network_layer_ready:   /* the network layer has a packet to send */
                                    /* Accept, save, and transmit a new frame. */
            from_network_layer(&buffer[next_frame_to_send]);   /* fetch new packet */
            nbuffered = nbuffered + 1;              /* expand the sender's window */
            send_data(next_frame_to_send, frame_expected, buffer);   /* transmit the frame */
            inc(next_frame_to_send);                /* advance sender's upper window edge */
            break;
            ... ...
    }
    if (nbuffered < MAX_SEQ)    enable_network_layer();
    else    disable_network_layer();
  }
}
```

… 3.4.2 – Protocolo Go-back-N

```
while (true) {
    wait_for_event(&event);                    /* four possibilities: see event_type above */
    switch(event) {
        ... ...
        case frame_arrival:                    /* a data or control frame has arrived */
            from_physical_layer(&r);           /* get incoming frame from physical layer */
            if (r.seq == frame_expected) {
                /* Frames are accepted only in order. */
                to_network_layer(&r.info);     /* pass packet to network layer */
                inc(frame_expected);           /* advance lower edge of receiver's window */
            }
        ... ...
    }
    if (nbuffered < MAX_SEQ)    enable_network_layer();
    else    disable_network_layer();
}
}
```

… 3.4.2 – Protocolo Go-back-N

```
while (true) {
    wait_for_event(&event);                    /* four possibilities: see event_type above */
    switch(event) {
        ... ...
        case frame_arrival:                    /* a data or control frame has arrived */
            from_physical_layer(&r);           /* get incoming frame from physical layer */
            ... ...
            /* Ack n implies n – 1, n – 2, etc.  Check for this. */
            while (between(ack_expected, r.ack, next_frame_to_send)) {
                /* Handle piggybacked ack. */
                nbuffered = nbuffered – 1;     /* one frame fewer buffered */
                stop_timer(ack_expected);      /* frame arrived intact; stop timer */
                inc(ack_expected);             /* contract sender's window */
            }
            break;
    }
    if (nbuffered < MAX_SEQ)    enable_network_layer();
    else    disable_network_layer();
  }
}
```

… 3.4.2 – Protocolo Go-back-N

```
while (true) {
    wait_for_event(&event);                          /* four possibilities: see event_type above */
    switch(event) {
        ... ...
        case cksum_err: break;                       /* just ignore bad frames */
        case timeout:                                /* trouble; retransmit all outstanding frames */
            next_frame_to_send = ack_expected;       /* start retransmitting here */
            for (i = 1; i <= nbuffered; i++) {
                send_data(next_frame_to_send, frame_expected, buffer);/* resend frame */
                inc(next_frame_to_send);             /* prepare to send the next one */
            }
    }
    if (nbuffered < MAX_SEQ)    enable_network_layer();
    else    disable_network_layer();
  }
}
```

… 3.4.2 – Protocolo Go-back-N

- Obs.: Tem-se no máximo MAX_SEQ quadros, e não MAX_SEQ + 1 pendentes em qualquer instante …
  - … ainda que tenhamos MAX_SEQ + 1 nros de seqüência distintos, ou seja, 0, 1, 2,..., MAX_SEQ.

- Para saber por que essa restrição é necessária, considere a situação a seguir, com MAX_SEQ = 7.
  - transmissor envia quadros de 0 a 7;
  - confirmação com piggyback com quadro 7 volta ao TX;
  - TX envia mais 08 quadros, novamente com nros de seqüência de 0 a 7;
  - chega outra confirmação com piggyback correspondente ao quadro 7.

3.4.3 – Protocolo de Repetição Seletiva

- “Go-back-N” - … funciona bem quando há poucos erros;
  - ... mas se a linha estiver muito ruidosa, há desperdício de largura de banda com os quadros retransmitidos.
- “alternativa” … lidar com erros é permitir que RX aceite no “buffer” os quadros subseqüentes a um quadro danificado/perdido.

- TX e RX mantêm uma janela de nros. de seqüência aceitáveis;
  - … janela do transmissor é medido a partir de 0 e atinge um número máximo predefinido, MAX_SEQ.
  - … janela do receptor tem sempre um tamanho fixo e igual a MAX_SEQ, pois o receptor tem um buffer reservado para cada nro. de sequência dentro de sua janela de recepção que é fixa.

… 3.4.3 – Protocolo de Repetição Seletiva

- Sempre que um quadro chega, função “between” verifica seu nro. de seqüência para confirmar se ele se enquadra na janela.
  - … se isso ocorrer e se o quadro ainda não tiver sido recebido, ele será aceito e armazenado na camada de enlace;
  - … essa ação é executada sem levar em conta se o quadro contém ou não o próximo pacote esperado pela camada de rede;
  - … mantém-se o quadro na camada de enlace, mas não o repassa à camada de rede, até que todos os quadros de nros. mais baixos tenham sido entregues à camada de rede na ordem correta.

```
/* Protocol 6 (Selective repeat) accepts frames out of order but passes packets to the
   network layer in order. Associated with each outstanding frame is a timer. When the timer
   expires, only that frame is retransmitted, not all the outstanding frames, as in protocol 5. */
```

… 3.4.3 – Protocolo de Repetição Seletiva

- … associado a cada buffer há um bit (arrived) que informa se o buffer está cheio ou vazio.

```
/* Protocol 6 (Selective repeat) accepts frames out of order but passes packets to the
   network layer in order. Associated with each outstanding frame is a timer. When the timer
   expires, only that frame is retransmitted, not all the outstanding frames, as in protocol 5. */

#define MAX_SEQ 7                                   /* should be 2^n – 1 */
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout} event_type;
#include "protocol.h"
boolean no_nak = true;                              /* no nak has been sent yet */
seq_nr oldest_frame = MAX_SEQ + 1;                  /* initial value is only for the simulator */

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Same as between in protocol 5, but shorter and more obscure. */
  return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}
```

… 3.4.3 – Protocolo de Repetição Seletiva

- … associado a cada buffer há um bit (arrived) que informa se o buffer está cheio ou vazio.

```
static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet buffer[ ])
{
/* Construct and send a data, ack, or nak frame. */
  frame s;                                        /* scratch variable */

  s.kind = fk;                                    /* kind == data, ack, or nak */
  if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
  s.seq = frame_nr;                               /* only meaningful for data frames */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
  if (fk == nak) no_nak = false;                  /* one nak per frame, please */
  to_physical_layer(&s);                          /* transmit the frame */
  if (fk == data) start_timer(frame_nr % NR_BUFS);
  stop_ack_timer();                               /* no need for separate ack frame */
}
```

… 3.4.3 – Protocolo de Repetição Seletiva

```
void protocol6(void)
{
    seq_nr ack_expected;                    /* lower edge of sender's window */
    seq_nr next_frame_to_send;              /* upper edge of sender's window + 1 */
    seq_nr frame_expected;                  /* lower edge of receiver's window */
    seq_nr too_far;                         /* upper edge of receiver's window + 1 */
    int i;                                  /* index into buffer pool */
    frame r;                                /* scratch variable */
    packet out_buf[NR_BUFS];                /* buffers for the outbound stream */
    packet in_buf[NR_BUFS];                 /* buffers for the inbound stream */
    boolean arrived[NR_BUFS];               /* inbound bit map */
    seq_nr nbuffered;                       /* how many output buffers currently used */
    event_type event;

    enable_network_layer();                 /* initialize */
    ack_expected = 0;                       /* next ack expected on the inbound stream */
    next_frame_to_send = 0;                 /* number of next outgoing frame */
    frame_expected = 0;
    too_far = NR_BUFS;
    nbuffered = 0;                          /* initially no packets are buffered */
    for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
    ... ...
```

… 3.4.3 – Protocolo de Repetição Seletiva

```
void protocol6(void)
{
    ... ...
    while (true) {
        wait_for_event(&event);                    /* five possibilities: see event_type above */
        switch(event) {
            case network_layer_ready:              /* accept, save, and transmit a new frame */
                ... ...
            case frame_arrival:                    /* a data or control frame has arrived */
                ... ...
            case cksum_err:
                ... ...
            case timeout:
                ... ...
            case ack_timeout:
                ... ...
        }
        if (nbuffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
    }
}
```

… 3.4.3 – Protocolo de Repetição Seletiva

- …

```
case network_layer_ready:                /* accept, save, and transmit a new frame */
  nbuffered = nbuffered + 1;             /* expand the window */
  from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]); /* fetch new packet */
  send_frame(data, next_frame_to_send, frame_expected, out_buf); /* transmit the frame */
  inc(next_frame_to_send);               /* advance upper window edge */                 */
  break;

 ... ...

case cksum_err:
    if (no_nak) send_frame(nak, 0, frame_expected, out_buf); /* damaged frame */
    break;
case timeout:
    send_frame(data, oldest_frame, frame_expected, out_buf); /* we timed out */
    break;
case ack_timeout:
    send_frame(ack,0,frame_expected, out_buf);      /* ack timer expired; send ack */
```

… 3.4.3 – Protocolo de Repetição Seletiva

```
case frame_arrival:                          /* a data or control frame has arrived */
   from_physical_layer(&r);                  /* fetch incoming frame from physical layer */
   if (r.kind == data) {
      /* An undamaged frame has arrived. */
      if ((r.seq != frame_expected) && no_nak)
         send_frame(nak, 0, frame_expected, out_buf); else start_ack_timer();
      if (between(frame_expected,r.seq,too_far) && (arrived[r.seq%NR_BUFS]==false)) {
         /* Frames may be accepted in any order. */
         arrived[r.seq % NR_BUFS] = true;       /* mark buffer as full */
         in_buf[r.seq % NR_BUFS] = r.info;      /* insert data into buffer */
         while (arrived[frame_expected % NR_BUFS]) {
            /* Pass frames and advance window. */
            to_network_layer(&in_buf[frame_expected % NR_BUFS]);
            no_nak = true;
            arrived[frame_expected % NR_BUFS] = false;
            inc(frame_expected);     /* advance lower edge of receiver's window */
            inc(too_far);            /* advance upper edge of receiver's window */
            start_ack_timer();       /* to see if a separate ack is needed */
         }
      }
   }
   ... ...
   break;
```

… 3.4.3 – Protocolo de Repetição Seletiva

```
case frame_arrival:                          /* a data or control frame has arrived */
   from_physical_layer(&r);                  /* fetch incoming frame from physical layer */
   if (r.kind == data) {
   |  ... ...
   }

   if((r.kind==nak) && between(ack_expected,(r.ack+1)%(MAX_SEQ+1),next_frame_to_send))
      send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected, out_buf);

   while (between(ack_expected, r.ack, next_frame_to_send)) {
   |  nbuffered = nbuffered – 1;             /* handle piggybacked ack */
   |  stop_timer(ack_expected % NR_BUFS);    /* frame arrived intact */
   |  inc(ack_expected);                     /* advance lower edge of sender's window */
   }
   break;
```

3.5.1 – Protocolo HDLC

- Protocolo SDLC (Synchronous Data Link Control) … protocolo de controle de enlace de dados síncrono proposto pela IBM;
  - … IBM submeteu o SDLC ao ANSI (USA) e à ISO (Mundo) como um padrão nos USA e no mundo inteiro, respectivamente.

- ANSI modificou SDLC, transformando-o no ADCCP (Advanced Data Communication Control Procedure — procedimento de controle de comunicação de dados avançado),
- ISO alterou o SDLC, para transformá-lo no HDLC (High-level Data Link Control — controle de enlace de dados de alto nível).
- CCITT adotou e modificou o HDLC e o transformou em seu LAP (Link Access Procedure — procedimento de acesso de enlace), como parte do padrão de interface de rede X.25.

… 3.5.1 – Protocolo HDLC

- SDLC, ADCCP, HDLC e LAP … são orientados a bits e utilizam a técnica de inserção de bits para transparência de dados.
- … “delimitadores” - … flags “01111110” transmitidos de forma contínua e que contém no mínimo 03 campos e totaliza 32 bits, excluindo os “flags” de cada extremidade.

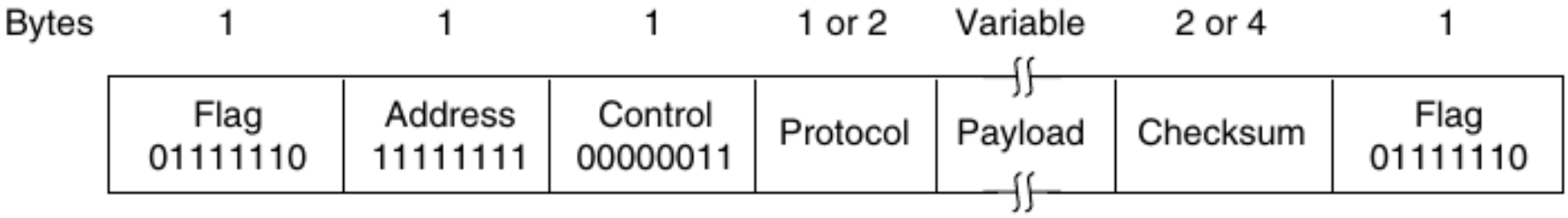


Figure 3-24. The PPP full frame format for unnumbered mode operation.

… 3.5.1 – Protocolo HDLC

- … todos os protocolos orientados a bits utilizam a estrutura de quadro apresentada abaixo, onde o campo “address” identifica um dos terminais nas linhas com vários terminais.
- … no caso de linhas ponto a ponto, este campo pode ser utilizado para fazer distinção entre comandos e respostas.

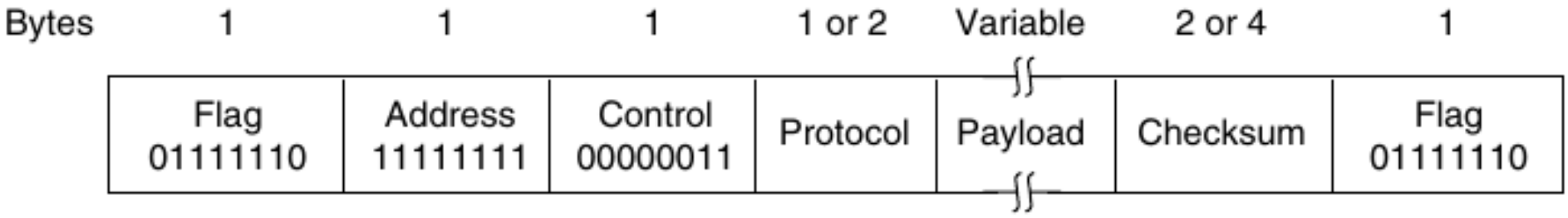


Figure 3-24. The PPP full frame format for unnumbered mode operation.

… 3.5.1 – Protocolo HDLC

- “controle” … usado para nro. de seqüência, confirmações e outras finalidades, como será discutido a seguir.
- “dados” - … pode ser arbitrariamente longo, embora a eficiência do “checksum” diminua com o aumento do tamanho do quadro, devido à maior probabilidade de erros em rajada.
- “checksum” - … variação do código de redundância cíclica.

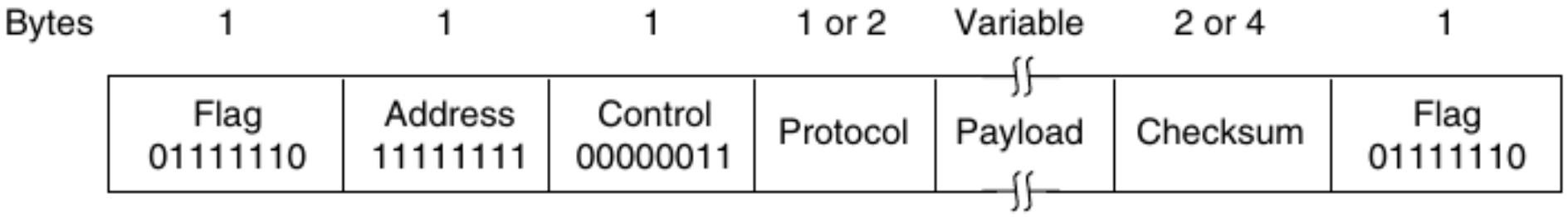
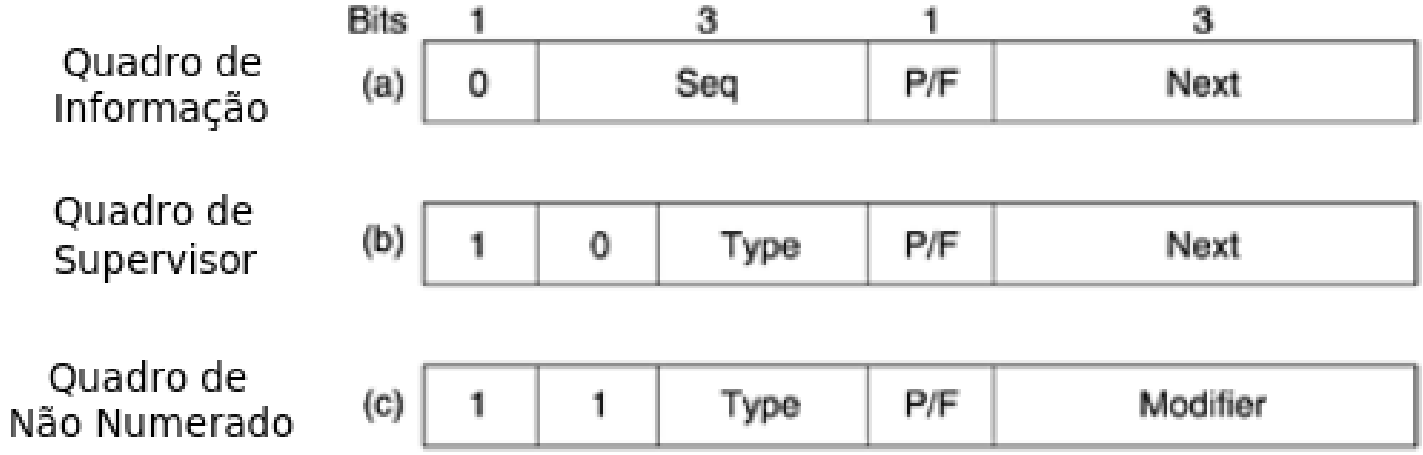


Figure 3-24. The PPP full frame format for unnumbered mode operation.

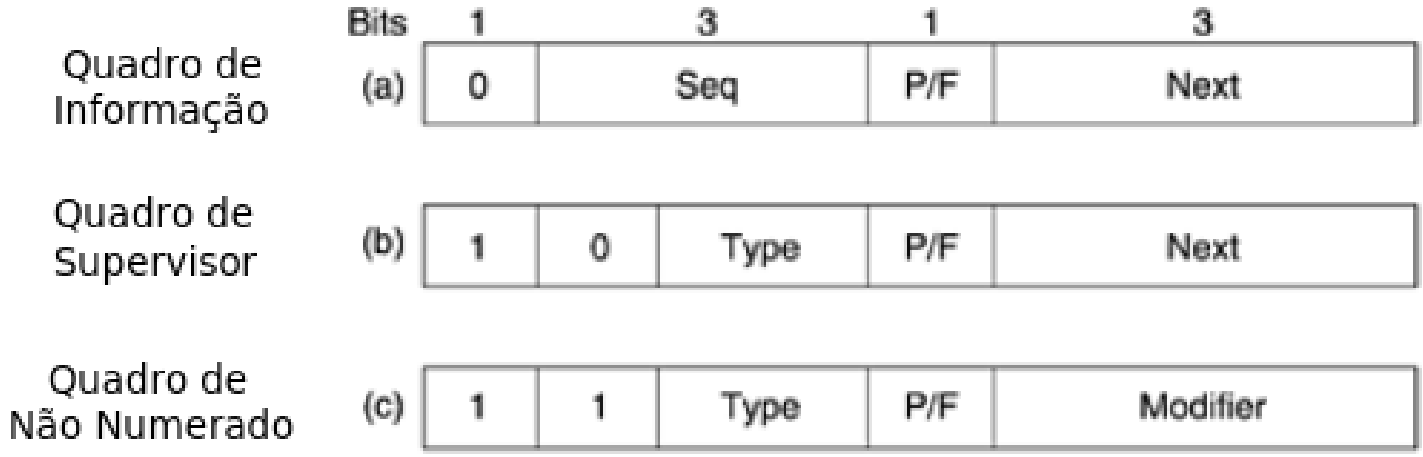
… 3.5.1 – Protocolo HDLC

- São 03 tipos de quadros: Quadro de Informação, Quadro Supervisor e Quadro Não Numerado.
- … protocolo utiliza uma janela deslizante, com o campo “seq” (nro. de sequência) de 3 bits, ou seja, até 07 quadros não confirmados pendentes;
- … próximo campo é confirmação transportada por piggyback.



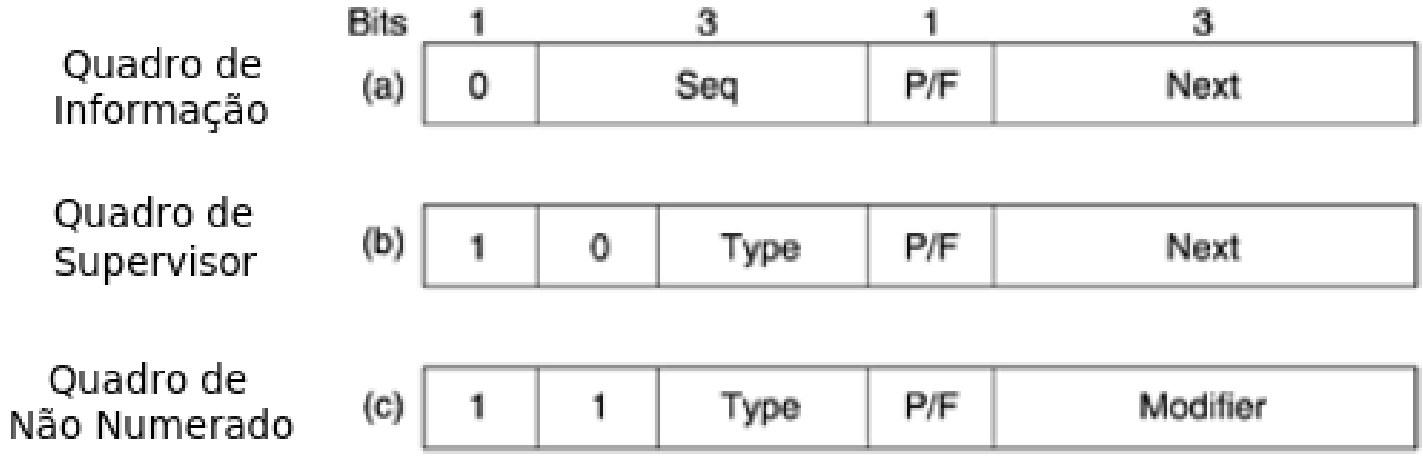
… 3.5.1 – Protocolo HDLC

- "P/F" - Poll/Final - … bit "P" ativado quando o terminal encaminha os quadros, com exceção do quadro final - "F" ativado.
  - … em alguns protocolos "P/F" é utilizado para forçar o outro "host" enviar imediatamente um quadro supervisor, em vez de aguardar o tráfego inverso para inserir nele as informações da janela.



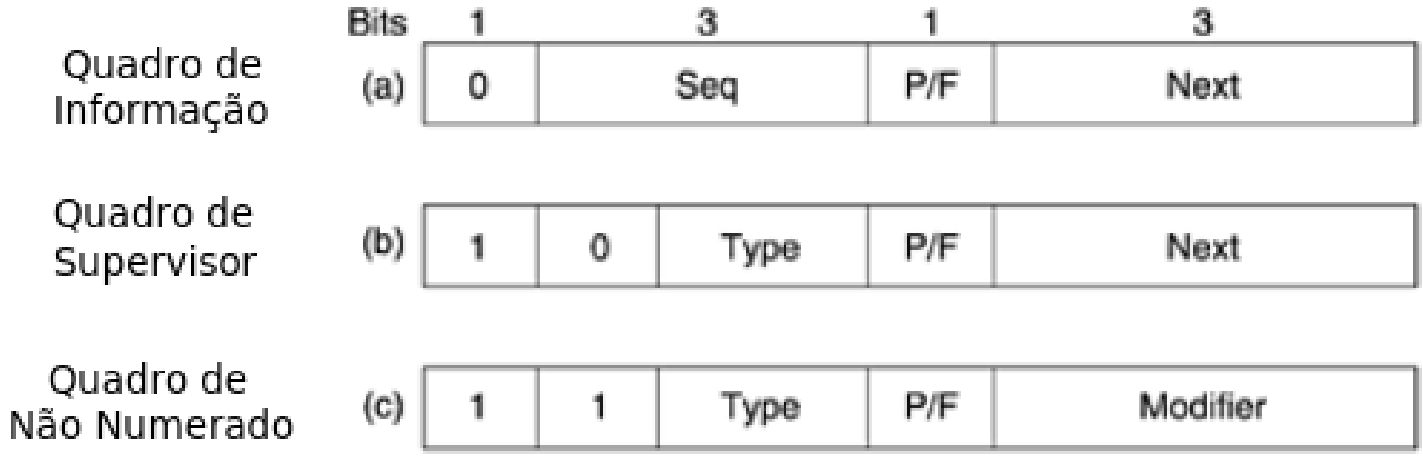
… 3.5.1 – Protocolo HDLC

- “tipo” = “0” .. quadro de confirmação “RECEIVE READY” usado para indicar o próximo quadro esperado.
- “tipo” = “1” .. REJECT … quadro de confirmação negativa.



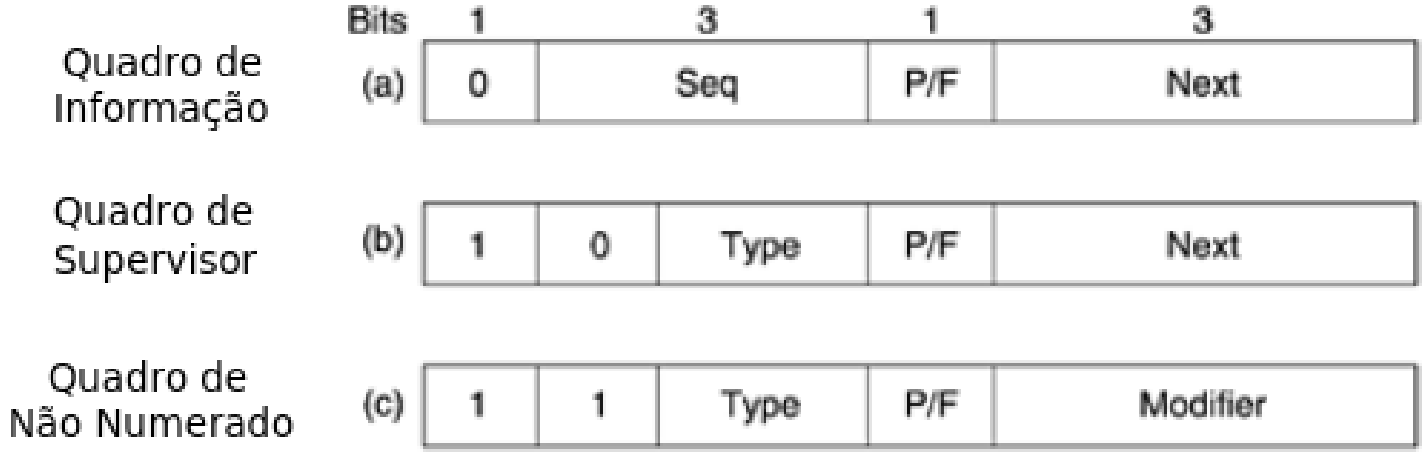
… 3.5.1 – Protocolo HDLC

- “tipo” = “2” .. RECEIVE NOT READY .. que confirma todos os quadros até (mas não incluindo) próximo, exatamente como RECEIVE READY, mas solicita que o TX interrompa o envio.
- “tipo” = “3” .. SELECTIVE REJECT .. solicita a retransmissão apenas do quadro especificado.



… 3.5.1 – Protocolo HDLC

- “Quadro Não Numerado” - … utilizado para fins de controle, mas também pode transportar dados quando se utiliza o serviço não confiável sem conexão.
- Obs.: … protocolos orientados a bits diferem consideravelmente nesse ponto, ao contrário dos outros dois tipos, nos quais eles são quase idênticos.



… 3.5.1 – Protocolo HDLC

- “Unnumbered Acknowledgment” - quadro de controle especial utilizado também para confirmação.
- … quadros de controle podem estar perdidos ou danificados, da mesma forma que os quadros de dados e, assim, eles também devem ser confirmados.
- … como apenas 01 quadro de controle pode estar pendente, nunca haverá qualquer ambigüidade em relação ao quadro de controle que está sendo confirmado.

… 3.5.1 – Protocolo HDLC

- Os quadros de controle restantes se referem à inicialização, ao polling e a relatórios de “status”.
- … também existe um quadro de controle que pode conter informações arbitrárias, o UI (Unnumbered Information).
- … esses dados não são repassados à camada de rede, mas se destinam à própria camada de enlace de dados do receptor.

- Apesar de sua ampla utilização, o HDLC está longe de ser perfeito. Uma discussão sobre uma variedade de problemas associados a ele pode ser encontrada em (Fiorini et al.,1994).