

# Camada de Enlace de Dados

Carlos E. Pereira - UFRGS/DELET

**GCAR**

# Camada de Enlace de Dados

- aborda algoritmos que permitem uma comunicação eficiente e confiável entre dois computadores adjacentes em nível da camada de enlace de dados (adjacentes no sentido de estarem fisicamente conectadas)

Carlos E. Pereira - UFRGS/DELET

**GCAR**

# Tarefas da Camada de Enlace de Dados

- Enquadramento (Delimitação de quadros)
- Controle de Erros
- Controle de Fluxo
- Gerenciamento de Enlace

Carlos E. Pereira - UFRGS/DELET

**GCAR**

## Enquadramento

- Fluxo de bits é dividido em quadros, sendo calculado um ‘checksum’ (dígito/código de verificação)

Carlos E. Pereira - UFRGS/DELET

**GCAR**

# Delimitação de Quadros

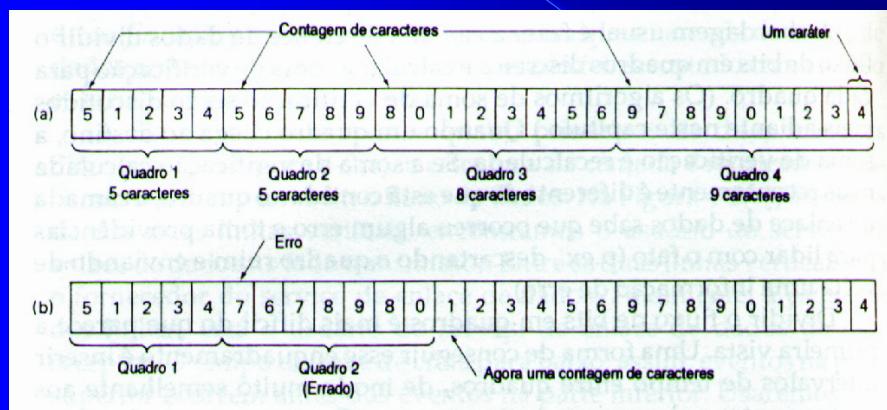
## ● 1. Contagem de Caracteres

- um campo do cabeçalho é usado para determinar número de caracteres do quadro
- problema: erros na transmissão (no campo com o número de caracteres)

Carlos E. Pereira - UFRGS/DELET

GCAR

## Contagem de Caracteres



Carlos E. Pereira - UFRGS/DELET

GCAR

# Delimitação de Quadros

## ● 2. Caracteres Iniciais e Finais com Inserção de Caracteres (character stuffing)

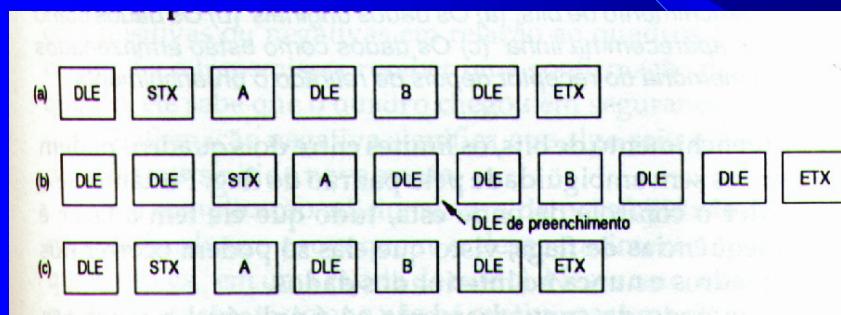
- DLE STX e DLE ETX (DLE = Data Link Escape)
- em caso de transmissão de arquivos binários:  
inclusão de DLE em cada seqüencia DLE que  
aparecer no arquivo (estes caracteres são  
removidos na recepção)
- desvantagem (perda de 8 bits a cada inserção)

Carlos E. Pereira - UFRGS/DELET

GCAR

# Delimitação de Quadros

## ● Inserção de caracteres



Carlos E. Pereira - UFRGS/DELET

GCAR

# Delimitação de Quadros

## ● 3. Flags iniciais e finais (bit stuffing)

- flag: símbolo inicial e final de quadro com um número qualquer de bits (previamente definido)
- ex: 0111110 (protocolo HDLC) => na transmissão de arquivos binários uma seqüência de cinco 1s consecutivos é sempre inserido um 0 de forma a evitar o aparecimento do flag
- vantagem: somente 1 bit adicional em cada inserção

Carlos E. Pereira - UFRGS/DELET

GCAR

# Delimitação de Quadros

## ● O que ocorre se 0111110 deve ser transmitido ?

### ● ex:

- 01111101011001011111011 (sinal a ser transmitido)
- 011111010110010111110011 (após bit stuffing)
- 01111101011001011111011 (sinal recuperado)

Carlos E. Pereira - UFRGS/DELET

GCAR

## Detecção e Correção de Erros

- **Erros isolados:** 1 bit em 1 quadro
- **Erros em rajada:** todo o quadro ou mais de um quadro é deturpado

Carlos E. Pereira - UFRGS/DELET

GCAR

## Detecção e Correção de Erros

- **Detecção de erro:** a partir do quadro recebido conclui-se que houve erro na transmissão e solicita-se reenvio
- **Correção de erro:** o quadro contém informações redundantes de forma a permitir a identificação de qual bit contém erro. Não necessita reenvio.

Carlos E. Pereira - UFRGS/DELET

GCAR

# Detecção e Correção de Erros

- **Palavra de código:** mensagem contendo  $m$  bits de dados e  $r$  bits redundantes => tamanho total  $n = m+r$
- **Distância de Hamming:** número de posições de bits em que duas palavras de código diferem => indica o número de erros que deve ocorrer (inversão de bits) para tornar uma palavra de código em outra válida

Carlos E. Pereira - UFRGS/DELET

GCAR

# Detecção e Correção de Erros

- Em geral  $2^m$  mensagens são válidas, porém nem todas possíveis  $2^n$  palavras de código são válidas
- Dado um conjunto de símbolos (palavras de código) válidos, determina-se a **distância de Hammig do conjunto** como sendo a menor distância de Hammig entre duas palavras de código válidas do conjunto

Carlos E. Pereira - UFRGS/DELET

GCAR

## Detecção e Correção de Erros

- Detecção de **d** erros: é possível caso a distância de Hamming do conjunto seja igual a **d+1**
- ex: paridade  
Distância de Hammig = 2, logo permite a **detecção de erros** em 1 único bit
- análise do código: 0000 0001 1000 1111

Carlos E. Pereira - UFRGS/DELET

GCAR

## Detecção e Correção de Erros

- **Correção de d erros:** é possível caso a distância de Hamming do conjunto seja igual a **2d+1**
- implica que, após a ocorrência de d erros a palavra recebida estará a uma distância de Hamming **d** de somente uma palavra válida (estará no mínimo a uma distância **d+1** de outra).

Carlos E. Pereira - UFRGS/DELET

GCAR

# Detecção e Correção de Erros

- supondo um código com  $n=m+r$  bits, cada uma das  $2^m$  mensagens válidas tem  $n$  palavras de código inválidas a uma distância igual a 1 (inversão de 1 único bit ou erros simples)
- logo, para permitir reconhecimento do erro, cada mensagem válida deve ter associado a ela  $(n+1)$  seqüências de bits
- logo o limite teórico é  $(n+1)*2^m \leq 2^n$  ou ainda  $m+r+1 \leq 2^r$  (logo, dado m posso saber r)

Carlos E. Pereira - UFRGS/DELET

GCAR

# Detecção e Correção de Erros

## ● Tabela

m	r
1	2
4	3
8	4
16	5
32	6

Carlos E. Pereira - UFRGS/DELET

GCAR

## Código de Hamming

- bits da palavra de código são numerados a partir da esquerda (início b1)
- todos os bits que são potências de 2 (1,2,4,...) são considerados bits de verificação (V)
- os outros bits (3,5,6,7,9,...) são preenchidos como bits de dados
- um bit de dados pode contribuir em diversos bits de verificação (ex: b5 contribui no 1 e 4)

Carlos E. Pereira - UFRGS/DELET

**GCAR**

## Código de Hamming

- ex: mensagem 1001000 (m=7,  
● V V 1 V 0 0 1 V 0 0 0  
b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11  
  
b1= 0 (b3 xor b5 xor b7 xor b9 xor b11)  
b2= 0 (b3 xor b6 xor b7 xor b10 xor b11)  
b4= 1 (b5 xor b6 xor b7)  
b8= 0 (b9 xor b10 xor b11)  
● logo código enviado seria 00110010000

Carlos E. Pereira - UFRGS/DELET

**GCAR**

## Código de Hamming

Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	11111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	00101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

Carlos E. Pereira - UFRGS/DELET

GCAR

## Código de Hamming

- inicializa-se um contador em zero
- verifica-se a paridade de cada bit de verificação
- se a paridade não estiver correta soma-se o valor da posição do bit de verificação ao contador
- no final: contador em zero = transmissão OK  
contador não zero = indica bit onde houve o erro

Carlos E. Pereira - UFRGS/DELET

GCAR

## Código de Hamming

- Supondo erro em um bit na transmissão 00110010001 em vez de 00110010000
- checagem:  
00110010001 cálculo dos bits de verificação: b1= 1 b2= 1 b4= 1 b8= 1  
uma vez que b1, b2 e b8 diferem, temos que erro está no bit 11

Carlos E. Pereira - UFRGS/DELET

GCAR

## Código de Hamming

- Aplicação para detecção de erro em rajada (vários bits afetados):
  - juntar k quadros formando uma matriz
  - transmitir a matriz por coluna
  - reconstruir a matriz na recepção
- Se um quadro for destruído, apenas 1 bit de cada quadro original é afetado, possibilitando sua correção

Carlos E. Pereira - UFRGS/DELET

GCAR

## Códigos de Detecção de Erros

- **Correção de erros:** usadas especialmente em caso de longos tempos de propagação, na maioria dos casos prefere-se somente a detecção e o re-envio
- ex: taxa de erro  $10^{-6}$  por bit (erros isolados) em um canal com tamanho de 1000 bits
  - Hamming: exigiria 10 bits, o que numa transmissão de 1 MByte implicaria em overhead de 10000 bits
  - Paridade: a cada 1000 blocos uma nova transmissão seria necessária ( $1000 \text{ bits} + 1 \text{ paridade} + 1000 \text{ paridade} = \text{overhead } 2001 \text{ bits}$ )

Carlos E. Pereira - UFRGS/DELET

GCAR

## Códigos de Detecção de Erros

- **Código de Redundância Cíclica (CRC)**
  - cadeias de bits são tratadas como polinômios
  - $k$  bits = polinômio  $x^k + x^{k-1} + x^{k-2} + \dots + x^0$
  - aritmética polinomial em módulo 2 (soma e subtração = XOR)
- transmissor e receptor devem concordar em relação ao polinômio gerador  $G(x)$

Carlos E. Pereira - UFRGS/DELET

GCAR

## Algoritmo de cálculo do CRC

- definir  $r$  como o grau de  $G(x)$ . Acrescentar  $r$  bits zero à extremidade de baixa ordem do quadro, de modo que ele passe a conter  $m+r$  bits e corresponda ao polinômio  $x^rM(x)$
- dividir (módulo 2)  $G(x)$  por  $x^rM(x)$
- subtraia (em módulo 2) o resto da divisão e acrescente no polinômio original (formando  $T(x)$ ) polinômio a ser transmitido, que é divisível por  $G(x)$  )

Carlos E. Pereira - UFRGS/DELET

GCAR

## Cálculo CRC

- ex:  $G(x) = x^4 + x + 1$  mensagem 1101011011

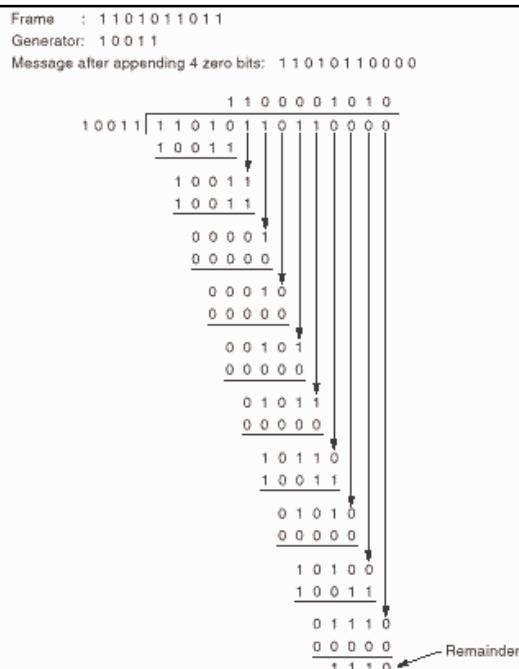
$$\begin{array}{r} 1100001010 \\ 10011 \longdiv{11010110110000} \\ \underline{10011} \downarrow \\ 10011 \\ \underline{10011} \downarrow \\ 00001 \\ \underline{00000} \downarrow \\ 00010 \\ \underline{00000} \\ \dots \text{resto} = 1110 \end{array}$$

Carlos E. Pereira - UFRGS/DELET

GCAR

## Cálculo do CRC

Carlos E. Pereira - UFRGS/DELET



GCAR

## Uso do CRC na recepção

- no receptor  $T(x)$  é dividido por  $G(x)$ . Caso haja erros  $T(x)$  passa a ser  $T(x) + E(x)$ .  
=> o resultado da divisão será  $E(x)/G(x)$
- para que erros possam ser detectados  $E(X)/G(x)$  deve ser diferente de zero

Carlos E. Pereira - UFRGS/DELET

GCAR

## Abrangência do uso de CRC

- Exemplo: detecção de 2 erros simples isolados  $E(x) = x^i + x^j$  onde  $i > j$   
ou ainda  $E(x) = x^j (x^{i-j} + 1)$
- para que todos os erros duplos sejam detectados **G(x) não deve dividir  $x^k + 1$  para qualquer k até um máximo valor  $i \leq j$**   
(máximo tamanho do quadro)  
ex:  $x^{15} + x^{14} + 1$  não divide  $x^k + 1$  para  $k < 32768$

Carlos E. Pereira - UFRGS/DELET

GCAR

## “CRCs - padrões”

- CRC-12:  $x^{12} + x^{11} + x^3 + x^2 + x + 1$   
para caracteres de 6 bits
- CRC-16:  $x^{16} + x^{15} + x^2 + 1$   
CRC-CCITT:  $x^{16} + x^{12} + x^5 + 1$   
detectam todos os erros simples e duplos, todos os erros com número ímpar de bits, todos os erros em rajada com no máximo 16 bits, 99.997 % das rajadas de erro de 17 bits
- vantagem: um simples circuito de deslocamento pode ser usado para cálculos

Carlos E. Pereira - UFRGS/DELET

GCAR

# Protocolo “Utópico”

- ⇒ Camada de rede sempre tem dados para transmitir
- ⇒ transmissor fica sempre transmitindo
- ⇒ receptor tem buffer infinito
- ⇒ canal não possui erros de transmissão

Carlos E. Pereira - UFRGS/DELET

GCAR

```
/* Protocol 1 (utopia) provides for data transmission in one direction only, from
   sender to receiver. The communication channel is assumed to be error free,
   and the receiver is assumed to be able to process all the input infinitely fast.
   Consequently, the sender just sits in a loop pumping data out onto the line as
   fast as it can. */

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_physicalLayer(&s);           /* send it on its way */
    }                                /* Tomorrow, and tomorrow, and tomorrow,
                                       Creeps in this petty pace from day to day
                                       To the last syllable of recorded time
                                       - Macbeth, V, v */
}

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_physicalLayer(&r); /* go get the inbound frame */
        to_network_layer(&r.info); /* pass the data to the network layer */
    }
}
```

Carlos E. Pereira - UFRGS/DELET

GCAR

## Controle de Erros no Enlace

- Para garantir transmissões confiáveis através de retransmissão, o procedimento em geral utilizado é fazer com que o destinatário de um quadro envie ao remetente quadros com avisos de reconhecimento positivo ou negativo dos quadros recebidos
- reconhecimento pode ser enviado como quadro de controle do nível 2 ou ‘de carona’ em campo de controle de quadro com informação **GCAR**

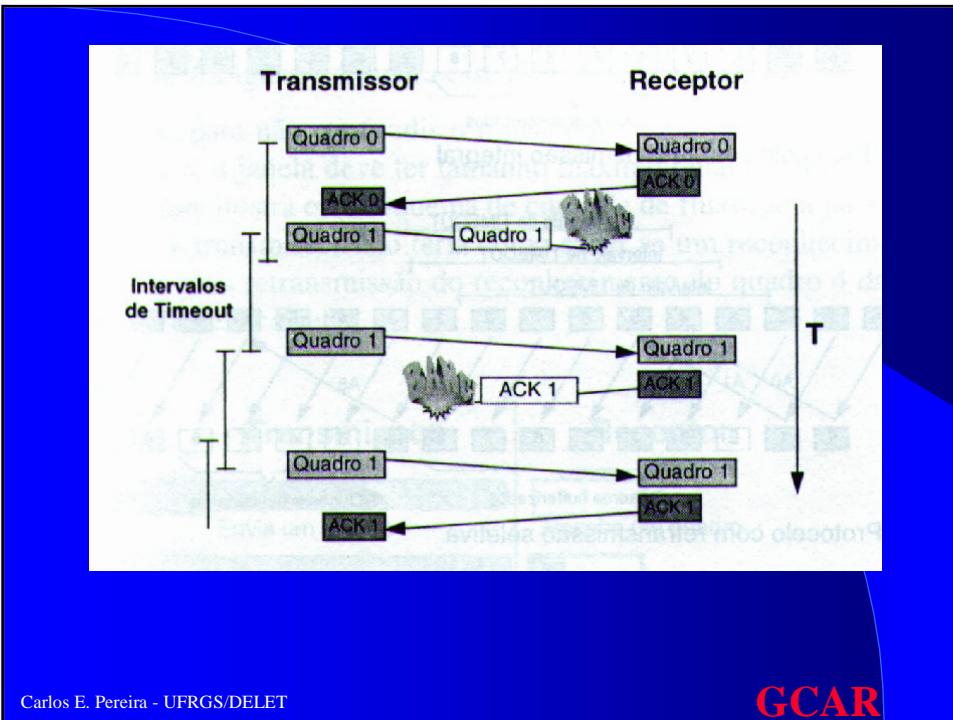
Carlos E. Pereira - UFRGS/DELET

## Controle de Erros no Enlace

- O que fazer se confirmação em caso de problemas na transmissão da mensagem ou da confirmação de recebimento ?
  - uso de temporizadores: controle de time-out

Carlos E. Pereira - UFRGS/DELET

**GCAR**



Carlos E. Pereira - UFRGS/DELET

## Protocolos Elementares de Enlace de Dados

- **simplex sem restrições:** transmissão somente num sentido, camadas sempre prontas a transmitir e receber. Supõe que camada de dados não apresenta erros e que receptor pode processar dados de forma infinitamente rápida
- **stop-and-wait simplex:** receptor demora para processar dados. Após processar o receptor envia um quadro para avisar transmissor

Carlos E. Pereira - UFRGS/DELET

GCAR

## Algoritmo ('simplex')

⇒ Transmissor  
while(true)  
    DeCamadaRede(buffer)  
    s.info = buffer  
    ParaCamadaFísica(s)

⇒ Receptor  
while (true)  
    Esperar(evento)  
    DaCamadaFísica(r)  
    ParaCamadaRede(r.info)

Carlos E. Pereira - UFRGS/DELET

GCAR

## Controle de Erros no Enlace

- Procedimentos mais utilizados para controle de erro:
  - simplex pára-e-espera
    - receptor com buffer finito
    - canal ruidoso
  - bit alternado (simplex para canal ruidoso)
  - janela n com retransmissão integral
  - janela n com retransmissão seletiva

Carlos E. Pereira - UFRGS/DELET

GCAR

# Protocolo ‘pára-e-espera’

- Receptor tem **buffer finito** (informa o transmissor se está pronto ou não a receber os dados)
- Transmissor
  - EnviaQuadro
  - Aguarda
- Receptor
  - RecebeQuadro
  - Processa
  - Envia sinal para continuar

Carlos E. Pereira - UFRGS/DELET

GCAR

```
/* Protocol 2 (stop-and-wait) also provides for a one-directional flow of data from
sender to receiver. The communication channel is once again assumed to be error
free, as in protocol 1. However, this time, the receiver has only a finite buffer
capacity and a finite processing speed, so the protocol must explicitly prevent
the sender from flooding the receiver with data faster than it can be handled. */
```

```
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 */
    }
}
```

```
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 */
    }
}
```

Carlos E. Pereira - UFRGS/D

## Canal Ruidoso

- ⇒ Quadros podem chegar danificados  
**(necessidade de retransmissão)**
- ⇒ Procedimento:
  - ⇒ Transmissor envia quadro
  - ⇒ Se quadro chegou corretamente, receptor confirma para enviar outra mensagem, caso contrário é descartado sem confirmação
  - ⇒ Caso não receba confirmação, transmissor retransmite quadro (após determinado tempo de espera)

Carlos E. Pereira - UFRGS/DELET

**GCAR**

## Protocolos Elementares de Enlace de Dados

### ● protocolo simplex para canal com ruído

- somente uma confirmação por parte do receptor não é suficiente (o que fazer se a comunicação é perdida ??)
- solução: adiciona-se um número de seqüência no cabeçalho de cada quadro enviado. Receptor informa caso recepção seja OK. Número de seqüência pode ter comprimento de apenas 1 bit

Carlos E. Pereira - UFRGS/DELET

**GCAR**

```

/* Protocol 3 (par) allows unidirectional data flow over an unreliable channel. */
#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 */

    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) {
                from_network_layer(&buffer); /* get the next one to send */
                inc(next_frame_to_send); /* invert next_frame_to_send */
            }
        }
    }
}

```

Carlos E. Pereira - UFRGS/DELET

GCAR

```

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);    /* none of the fields are used */
        }
    }
}

```

Carlos E. Pereira - UFRGS/DELET

GCAR

## Algoritmo de bit alternado

- Transmissor somente envia novo quadro depois de receber o reconhecimento do quadro enviado anteriormente
- Uma vez que quadros podem ser retransmitidos, é necessário numerá-los para que o receptor possa distinguir se é retransmissão ou novo quadro
- Como transmissor somente envia quadro após receber o último, 1 bit é suficiente

Carlos E. Pereira - UFRGS/DELET

GCAR

## Algoritmo de bit alternado

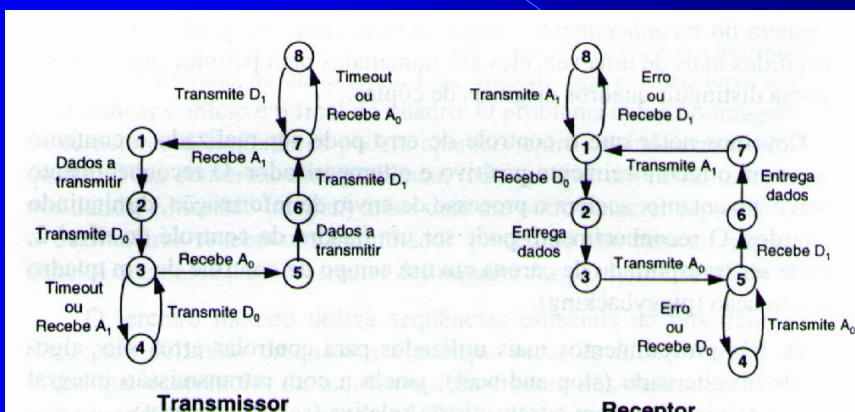


Figura 7.1: Diagrama de estados do algoritmo de bit alternado.

Carlos E. Pereira - UFRGS/DELET

GCAR

## Algoritmo de bit alternado

- Técnica simples porém ineficiente, pois canal não é usado enquanto confirmação é esperada

Carlos E. Pereira - UFRGS/DELET

**GCAR**

## Otimizações

- receptor envia confirmação de recebimento não em um quadro de controle, mas de ‘carona’ em um quadro de dados (‘piggybacking’) => melhor utilização da largura de banda do canal
- caso não tenha dados para enviar em um determinado intervalo, receptor envia confirmação como quadro de controle

Carlos E. Pereira - UFRGS/DELET

**GCAR**

## Otimizações

- Outra forma de aumentar a eficiência é permitir que transmissor envie várias mensagens mesmo sem ter recebido confirmação

Carlos E. Pereira - UFRGS/DELET

GCAR

## Protocolos de Janela Deslizante

- transmissor mantém janela de transmissão e receptor uma janela de recepção (não precisam ter o mesmo tamanho)
- janela de transmissão contém quadros enviados mas não confirmados (tamanho variável)
- janela de recepção contém quadros já recebidos e sendo processados (verificação de CRC, etc.) => tamanho constante

Carlos E. Pereira - UFRGS/DELET

GCAR

## Protocolos de Janela Deslizante

- Procedimento: Transmitir um número finito de quadros antes de parar e esperar pela confirmação: visa utilizar melhor o canal
- transmissor possui janela de tamanho variável contendo todos os quadros que pode transmitir. Cada quadro recebe uma numeração em seqüência.
- Receptor possui uma janela de tamanho fixo contendo os códigos de seqüência dos códigos que podem ser recebidos

Carlos E. Pereira - UFRGS/DELET

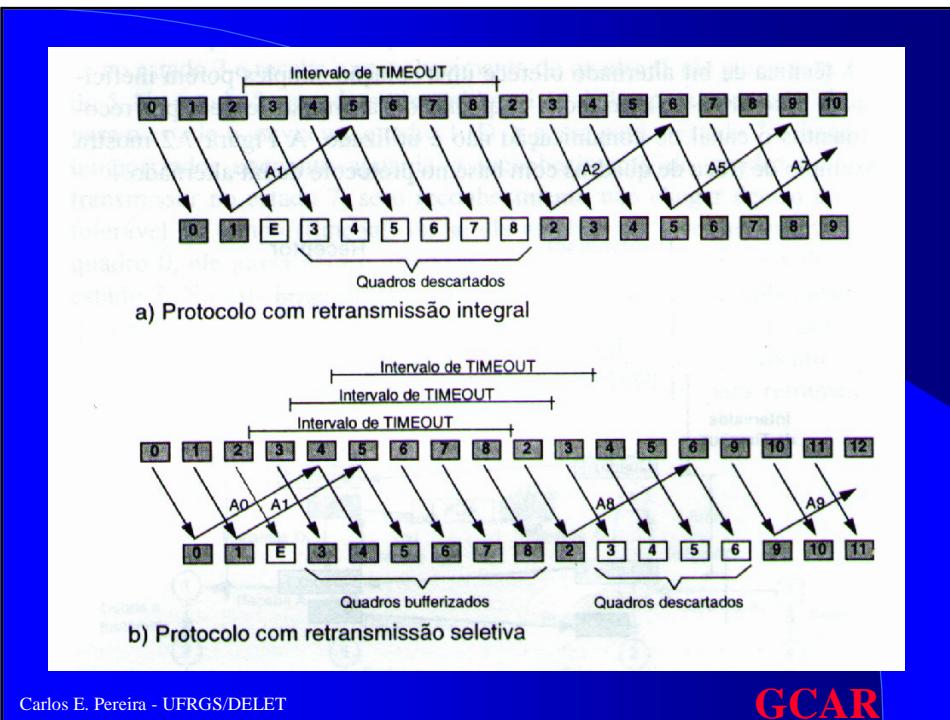
GCAR

## Como funciona em caso de erro de transmissão ?

- ‘Go back n’: ignora todos os quadros recebidos depois do quadro com erro até que o quadro originalmente errado seja recebido corretamente
- Repetição seletiva: os quadros recebidos corretamente após um quadro errado são bufferizados pela camada de enlace. Quando o quadro errado for recebido corretamente, todo o conjunto de quadros bufferizados é passado para a camada de rede

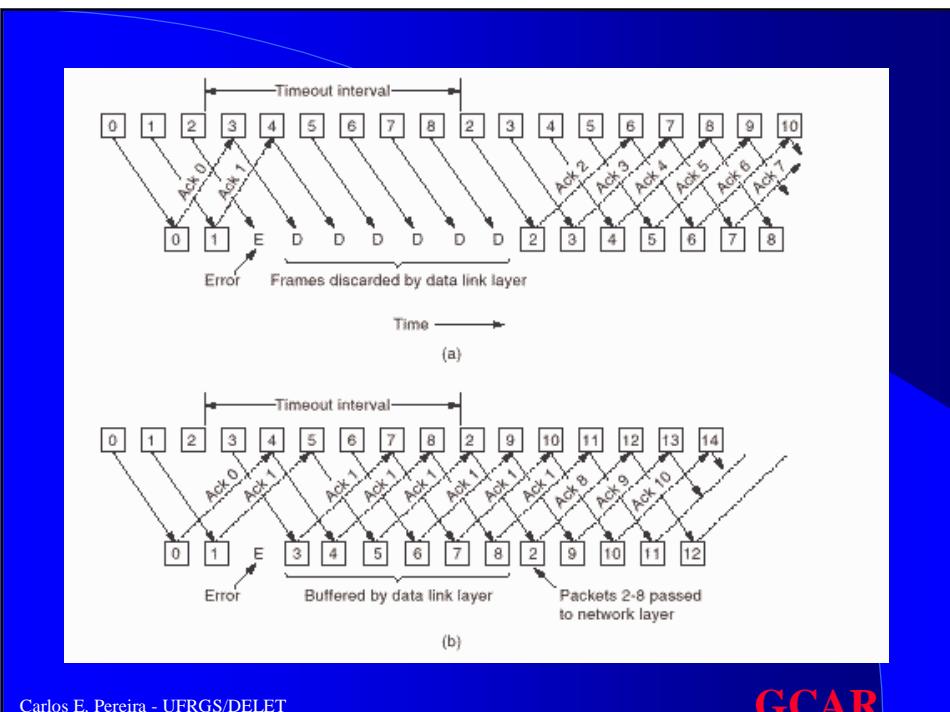
Carlos E. Pereira - UFRGS/DELET

GCAR



Carlos E. Pereira - UFRGS/DELET

GCAR



Carlos E. Pereira - UFRGS/DELET

GCAR

# Protocolos Elementares de Enlace de Dados

## ● Go back n:

- confirmação de um quadro n confirma automaticamente todos os quadros de seqüência menor que n

Carlos E. Pereira - UFRGS/DELET

GCAR

# Controle de Fluxo

- Transmissor rápido (rodando em máquina rápida ou não sobrecarregada) quer enviar dados para receptor lento (rodando em máquina lenta ou sobrecarregada)
  - Problema pode ser contornado através do uso do protocolo de janelas deslizantes

Carlos E. Pereira - UFRGS/DELET

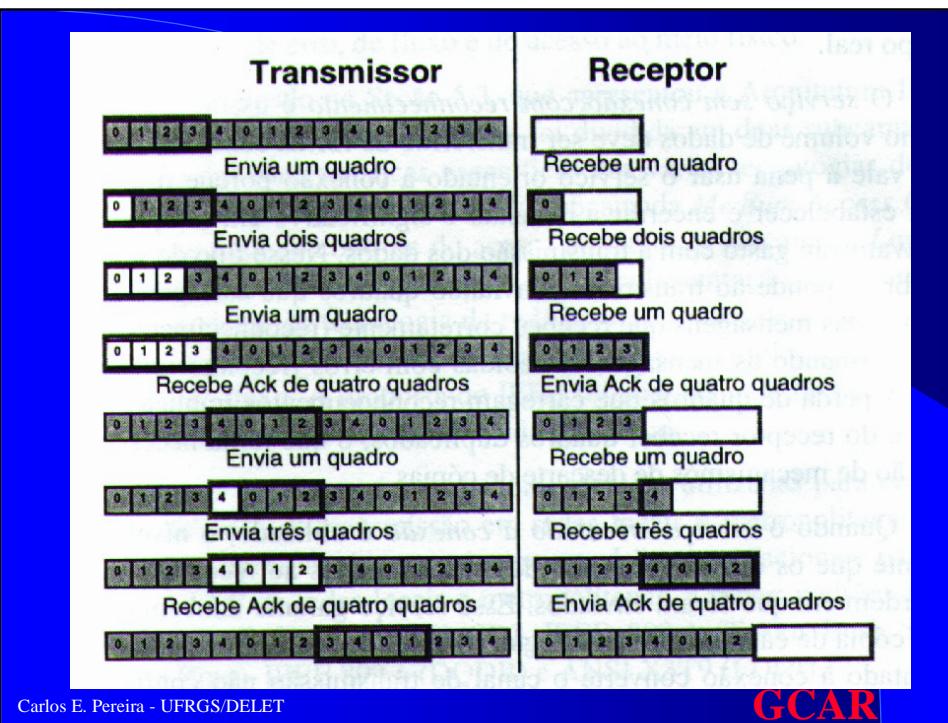
GCAR

## Protocolo de janela deslizante

- Janela de Transmissão com largura T
- após enviar T quadros, transmissor suspende envio e fica aguardando confirmação
- quadros numerados de 0 a T-1

Carlos E. Pereira - UFRGS/DELET

GCAR



Carlos E. Pereira - UFRGS/DELET

GCAR

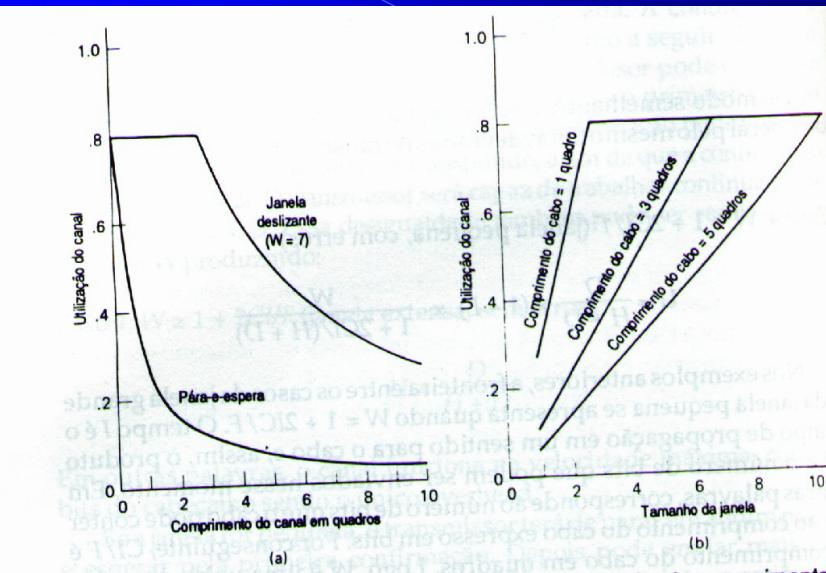
# Desempenhos dos Protocolos

- Dedução em Tannembaum

Carlos E. Pereira - UFRGS/DELET

GCAR

## Desempenho



Carlos E. Pereira - UFRGS/DELET

GCAR