UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Instituto de Informática

INF01154 - Redes de Computadores N

Prof. Valter Roesler

Laboratório 04

Guilherme Schievelbein

João Luiz Grave Gross

Porto Alegre, 11 de outubro de 2013.

Exercícios - Parte 1

1. Fazer a lista de exercícios de debate em aula.

1.1. O que é controle de fluxo? Como é feito em TCP? E em UDP?

Controle de fluxo consiste em ajustar o transmissor a uma certa taxa de transmissão, de modo a se fazer o melhor uso da banda da rede. Procura evitar a perda de dados que ocorre quando é enviado mais dados num intervalo de tempo do que a máquina receptora é capaz de processar.

Em TCP o controle de fluxo é justo para todos os usuários. Se um usuário TCP estiver transmitindo na taxa máxima da rede e um segundo usuário vier a transmitir informações, a banda da rede é dividida para ambos de forma igualitária. Isso é feito através de “sliding windows”.

Já na abordagem UDP, todos os terminais transmitem na sua velocidade máxima, não há controle de fluxo. Nesse caso fica a cargo do roteador armazenar em buffers os excessos de dados, já que as taxas de transmissões somadas irão exceder a taxa máxima da rede. Em um dado ponto o roteador começará a descartar dados por overflow dos buffers e as camadas de nível superior terão que tratar os pacotes perdidos. O programador pode implementar seu próprio controle de fluxo nas camadas superiores.

1.2. O que é janela deslizante? Qual a consequência de uma janela grande demais? E pequena demais?

Janela deslizante é uma técnica de envio de pacotes que prima por maximizar a utilização dos recursos de banda da rede. Em uma abordagem stop-and-wait, por exemplo, o transmissor envia um pacote, fica travado esperando resposta e só destrava quando a recebe. Nesse intervalo em que o transmissor fica travado, recursos preciosos da rede são desperdiçados, por subutilização da banda disponível.

A ideia da janela deslizante é realizar diversos envios de pacotes em sequência, enquanto houver janela disponível (enquanto ainda houver “espaço” na banda para enviar novos pacotes). O receptor receberá diversos pacotes e dos pacotes recebidos poderá dizer quais vieram corretamente e quais precisam ser retransmitidos. Há diversos algoritmos de janela deslizante, como go-back n e selective repeat, cada um com suas particularidades, porém possui o mesmo objetivo, maximizar a utilização da banda da rede, o que se traduz em maior velocidade de transmissão de dados.

Porém para o uso de janela deslizante deve levar alguns itens em consideração. A janela possui um limite e janelas muito grandes ou muito pequenas podem geram problemas.

Janelas deslizantes muito grandes congestionam o tráfego na rede, e podem criar injustiças. Por exemplo, se houver dois terminais transmissores na mesma rede, e um ele tiver uma janela

deslizante muito grande, este terminal iria pegar a banda da rede praticamente só para ele, deixando pouco espaço para outros transmissores. Se uma janela for maior que a banda, perde-se pacotes devido ao congestionamento.

Já janelas deslizantes muito pequenas desperdiçam recursos da rede, subutilizando-a. A janela deslizante foi pequena o suficiente seu comportamento irá se assemelhar ao de um algoritmo de stop-and-wait.

1.3. Como relacionar tamanho da janela com banda? Como relacionar RTT com banda?

O tamanho da janela e o fluxo na rede são diretamente proporcionais, já o RTT é inversamente proporcional ao fluxo.

Com um RTT muito alto deve-se buscar uma janela grande, pois se for pequena, boa parte da banda da rede será ocupada por RTTs. Com uma janela grande o suficiente, teremos uma ocupação satisfatória da banda da rede com transmissão de dados e pouca da banda será usada para RTTs.

1.4. Com RTT (Round Trip Time) alto, é mais eficiente tamanho de janela grande ou pequena? Por quê?

Como destacado na questão 1.3., com um RRT alto é interessante ter uma janela grande para aproveitar melhor os recursos da rede. Com uma janela pequena a banda é desperdiçada enviando poucos pacotes e o RTT alto faz com que fique tempo ocioso esperando a confirmação.

1.5. O que é “tempo de timeout”? Qual o problema para determinar o tempo de timeout (retransmit timer) em redes reais? Qual a solução?

Tempo de timeout é o tempo que cada mensagem enviada tem para receber uma confirmação sem que haja uma retransmissão automática daquele dado.

Em redes reais é necessário saber o tempo do RTT, pois com base nele que se dimensiona o timeout. Isso é um problema, pois o RTT varia entre as redes.

Caso o timeout seja pequeno, próximo ao tempo de RTT, facilmente o timeout será completado, retransmitindo dados muitas vezes sem necessidade. Logo, há uma relação que se estabelece entre timeout e RTT, na qual o timeout deve ser em média 4 vezes o valor do RTT, de modo a permitir que confirmações de envio tenham tempo de chegar no transmissor, evitando retransmissões sem necessidade e uma também evitando um canal ocioso.

1.6. Qual a consequência das perdas num protocolo real, tipo TCP, em termos de controle de fluxo? Outra forma de perguntar: o que o protocolo assume quando tem perdas? Lembre que as perdas acontecem nas duas direções (na transmissão do pacote ou no ACK).

Em um protocolo TCP, caso haja muitas perdas de pacotes o protocolo realiza uma inferência de que a taxa de transmissão de seus terminais está muito elevada. O protocolo é inteligente o suficiente para identificar esse problema e realizar correções de taxa de transmissão, reduzindo o tamanho da janela até um ponto ideal, ou seja, sem perdas. As vezes o problema pode não ser o tamanho da janela, mas o que causa as perdas é um sinal/cabo ruim.

Caso muito acks sejam perdidos, o tempo de timeout é aumentado.

2. Um canal tem uma taxa de 100 Mbit/s e um retardo de propagação de 20 ms e o programador escolheu utilizar quadros de 1.250 bytes. Para que tamanho de janela o sistema proporciona uma eficiência de pelo menos 50%? Desconsidere os tempos de inserção e desinserção dos pacotes na rede, ou seja, considere que o tempo de propagação é o mesmo para qq tamanho de janela.

taxa: 100Mbit/s

tamanho de quadro: 10.000 bits

tempo de propagação: 20 ms

tempo de quadro:

100 M bits - 1s

10.000 bits - x s

x = 10.000 / 100 M = 0,0001 s = 100us

tamanho da janela: w

- utilização do canal em 100%: w >= 2a + 1
- utilização do canal em w / (2a + 1): w < 2a + 1

a = tpropagação / tquadro

a = 20 ms / 100 us = 200

w >= 2*200 + 1

w >= 401

Com tamanho de janela w = 401, temos uma utilização de 100% do canal.

3. Quadros de 1.250 bytes são enviados de A para B por um canal de 4 Mbit/s usando um satélite geoestacionário. Suponha que o tempo de propagação de A para B seja de 270 ms. Qual é a capacidade máxima do canal utilizando os seguintes mecanismos:

Tamanho de quadro: 10.000 bits

taxa: 4 M bits/s

tempo de propagação: 270 ms

4 M bits - 1s

10.000 bits - x s

x = 10.000 / 4 M = 0,0025 s = 2,5 ms

a = 270 ms / 2,5 ms = 108

w >= 2 * a + 1

a. Stop-and-wait?

No stop and wait é como se a janela tivesse tamanho 1, logo:

Utilização do canal:

w / (2 * a + 1)

1 / (2 * 108 + 1) = 0.004608 = 0,46 %

Capacidade máxima do canal: utilização do canal * taxa de transmissão

(0,46/100) * 4 M bits /s = 18.400 bits / s = 18,4 kbits / s

b. Janela deslizante de 10 pacotes?

Considerando o go-back n como algoritmo de janela deslizante temos o seguinte:

w = 10

Utilização do canal:

w / (2 * a + 1) => 10 / (2 * 108 + 1) = 0.04608 = 4,608 %

Capacidade máxima do canal: utilização do canal * taxa de transmissão

(4,6/100) * 4 M bits /s = 184.000 bits / s = 184 kbits / s

c. Janela deslizante de 250 pacotes?

Considerando o go-back n como algoritmo de janela deslizante temos o seguinte:

w = 250

Utilização do canal:

w / (2 * a + 1) => 250 / (2 * 108 + 1) = 1.152 => 100%

Capacidade máxima do canal: 4 Mbits/s (taxa máxima)

Experiência - Parte 1

1. Explore o demo Sliding Window em http://www2.rad.com/networks/2004/sliding_window/demo.html.

Eventualmente será necessário entrar via web archive (http://web.archive.org). Explique apoiado com imagens da execução da tela do computador o funcionamento do demo sliding window. No mínimo, os seguintes itens devem ser explicados:

a. Funcionamento de janela deslizante.

São enviado vários pacotes, de acordo com o tamanho da janela. Quando o "sender" receber os acks desses pacotes, a janela "desliza" para enviar os novos pacotes. As janelas só podem deslizar quando receberem os pacotes/acks na posição mais a esquerda na janela.

Quando o primeiro pacote da janela recebe confirmação a janela desliza para enviar novos pacotes. Caso haja algum erro (perda de pacote ou ack), o algoritmo aguarda o timeout daquele pacote para realizar nova retransmissão. Um ack referente a um pacote só é enviado se os pacotes anteriores já foram recebidos também. Se um pacote é perdido, ele e todos os pacotes seguintes são reenviados (algoritmo GO BACK N).

b. Comente três diferenças desse simulador em relação ao TCP utilizado na WEB.

No TCP original, ao receber o terceiro ACK duplicado ocorre o fast-retransmit, onde o pacote pedido é imediatamente reenviado, ao invés de esperar pelo timeout. Outra diferença é que no simulador o receptor avisa até onde recebeu, já o TCP indica qual o pacote está pronto para receber. Além disso, no TCP a janela é ajustável conforme a disponibilidade de banda na rede, enquanto no simulador a janela é fixa.

2. Acesse o simulador em http://histrory.visualland.net/tcp_swnd.html ou

http://histrory.visualland.net/tcp_video.php?video=tcp2%20sliding%20window&protocol=TCP&title=2v.Sliding%20Window. Verifique a simulação. Explique o mecanismo de crescimento da janela deslizante, definindo o slow-start e congestion avoidance.

Com o slow-start a conexão é iniciada com uma janela de tamanho 1, e o tamanho da janela é dobrado, até que comece a ocorrer perdas. Quando ocorrer alguma perda, o congestion avoidance entra em ação. O tamanho da janela é divido por dois, e passa a ser aumentado por um. A janela é então mantida no tamanho que não ocorrem mais perdas.

Exercícios - Parte 2

1. Utilizando o polinômio CRC-CCITT (x16 + x12 + x5 + 1), gere um código CRC de 16 bits para uma mensagem formada por um bit 1 seguido de quinze bits 0.

Mensagem: M(x) = x15

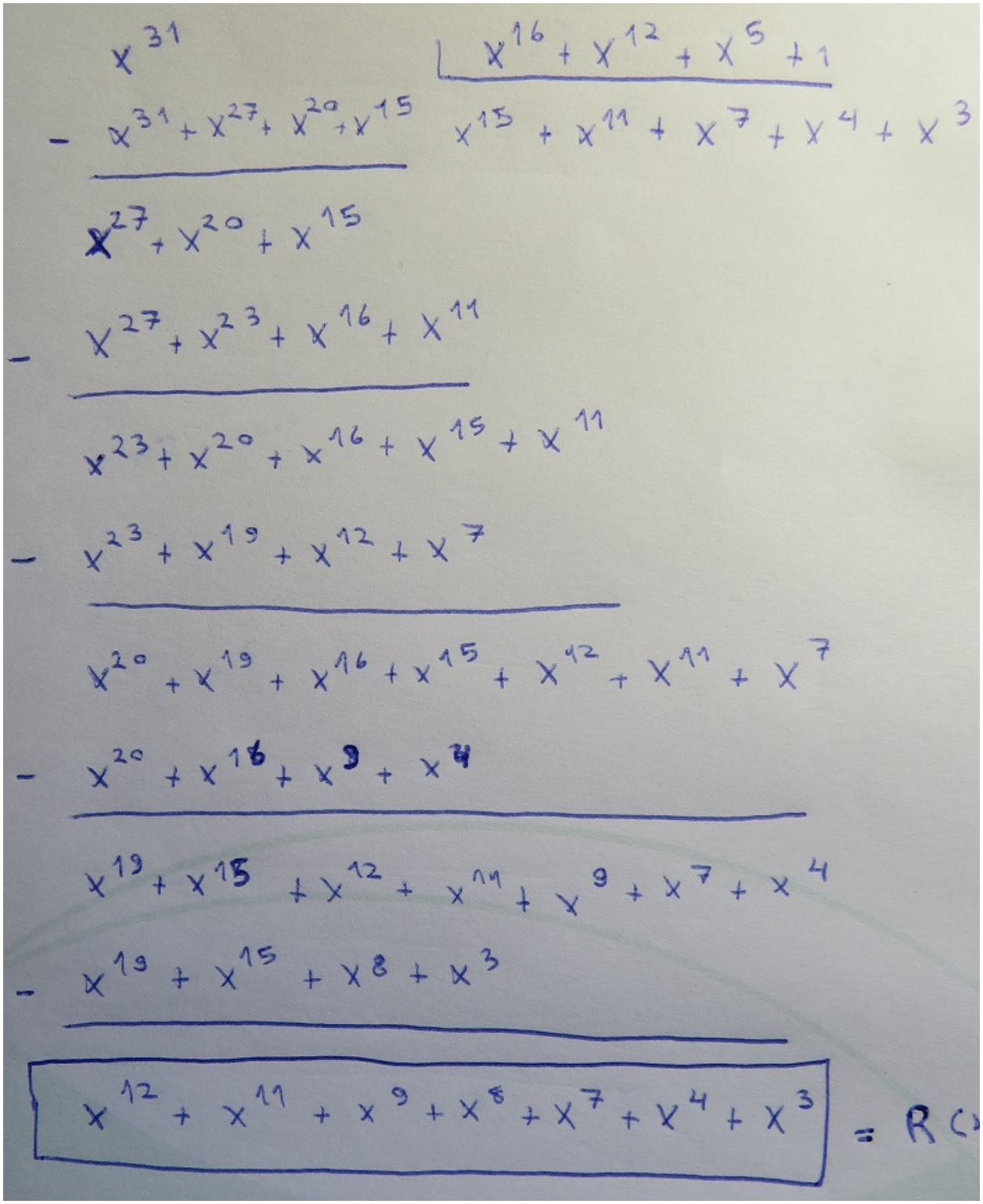
Polinômio gerador: G(x) = x16 + x12 + x5 + 1 => possui grau 16, ou seja, r = 16

Cálculo do CRC:

CRC é resto da divisão Xr * M(x) / G(x).

(x16 * x15) / (x16 + x12 + x5 + 1)

(x31) / (x16 + x12 + x5 + 1)



O resto R(x) é igual a x12 + x11 + x9 + x8 + x7 + x4 + x3. Logo, o código CRC de 16 bits é 0001

1011 1001 1000

2. Considere o código Reed Solomon utilizado em TV Digital, com o identificador (204,188), com 16 bytes de redundância. Para cada bloco, é possível corrigir 8 bytes errados. Descreva com detalhes uma forma de aumentar o número de bytes corrigidos para 80 bytes errados num pacote, sem aumentar a redundância de cada quadro individual.

Um forma seria utilizar o CIRC (Cross Interleaved Reed-Solomon Coding), pois ela intercala bits de diferentes blocos (interleave), assim um erro em rajada que afeta vários bits de um mesmo bloco é distribuído por vários blocos, sendo que cada bloco irá tratar o erro, que agora é pequeno para cada bloco. Depois de tratados os erros ocorre o desentrelaçamento dos blocos (deinterleave), na qual os blocos voltam a ficam com a sua configuração inicial.

Experiência - Parte 2

1. O simulador ReedSolomon-Test permite verificar o funcionamento da correção de erros com o ReedSolomon. Está em alemão, mas o significado é:

Bit Län ge – Número de bits por símbolo (fixo)

Fehler Korrigierbar – Número de símbolos máximo que o código corrige

Botão Daten – Gera dados aleatórios se campo ao lado possuir valores negativos.

Eingabe Daten – Dados de entrada

Beschädigte Daten – Dados corrompidos

Check – Diferença entre dados de entrada e dados corrompidos

Reparierte Daten – Dados Recuperados

Campos que não estão circulados podem ser editados

Ao editar um campo pressione a tecla Enter para atualizar os dados

Campo de dados recuperados em verde indica recuperação bem sucedida.

Baixe o simulador em:

http://runtimebasic.net/_media/Projekt:ReedSolomon.zip?id=Projekt%3ADownload&cache=nocache e verifique seu funcionamento editando o campo de dados corrompidos, sua paridade e o valor do campo Fehler Korrigierbar. Questões:

a) O que determina o número no campo Fehler Korrigierbar?

Determina o número máximo de símbolos que o código corrige.

b) Qual é o número máximo de bits (eficiência do algoritmo) que podem ser corrigidos com Fehler Korrigierbar = 3? a) no melhor caso de erros na linha; b) no pior caso de erros. A ideia da questão é reforçar que o Reed Solomon trabalha com símbolos, e não com bits.

Melhor caso: apenas um bit errado por símbolo. Como Fehler Korrigierbar está em 3, poderemos corrigir 3 bits (1 bit/símbolo * 3 (Fehler Korrigierbar)), ou seja o tamanho do bloco.

Pior caso: no pior caso de erros em linha, se todos os bits forem errados por símbolo e Fehler Korrigierbar = 3, poderemos corrigir 12 bits (4 bits/símbolo * 3 (Fehler Korrigierbar)).

c) No caso dos CDs, erros em rajadas maiores que a capacidade de correção de cada bloco (frame) são comuns. Que técnica poderíamos utilizar para evitar a perda de dados

com rajadas grandes sem ter que aumentar o número de símbolos de redundância da codificação? Explique.

Como explicado na questão 2 dos exercícios, parte 2, pode-se usar a técnica do CIRC, para distribuir o erro entre diferentes blocos a partir de um entrelaçamento dos bits de diferentes blocos. Desse modo o erro afeta poucos bits em vários blocos, o que é corrigível, ao invés de vários bits em único bloco.

2. Baixe o simulador da página da disciplina (só funciona em 32 bits), leia o help e efetue os testes indicados, utilizando os arquivos como base (eles estão localizados no diretório de instalação do programa). Execute passo a passo os comandos mostrados no help. Edite o arquivo manualmente e veja se ele consegue corrigir o número de erros prometido. O relatório deve conter o resultado obtido passo a passo, aumentando o número de símbolos errados até o sistema não recuperar mais os erros. Justifique. Os passos a serem efetuados são:

a) RS Input: configurar os parâmetros do Reed Solomon (default – RS (255,249) símbolos de 8 bits).

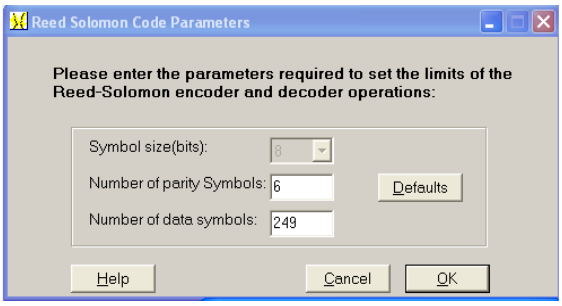
b) Encode: Entrar o arquivo de dados para codificação

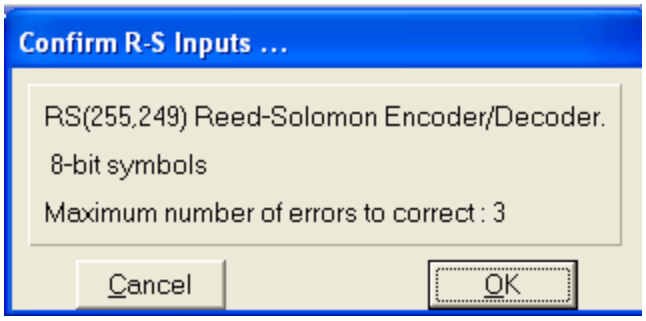
c) Insert Errors: Inserir erros no arquivo, de preferência de forma manual (editando o arquivo codificado e gerando um ".err" com algum editor de texto).

d) Decode: Selecionar o arquivo codificado (já com os erros – ".err") para decodificação.

e) Compare: Comparar o arquivo original e o decodificado, vendo se os erros foram corrigidos

INPUT:

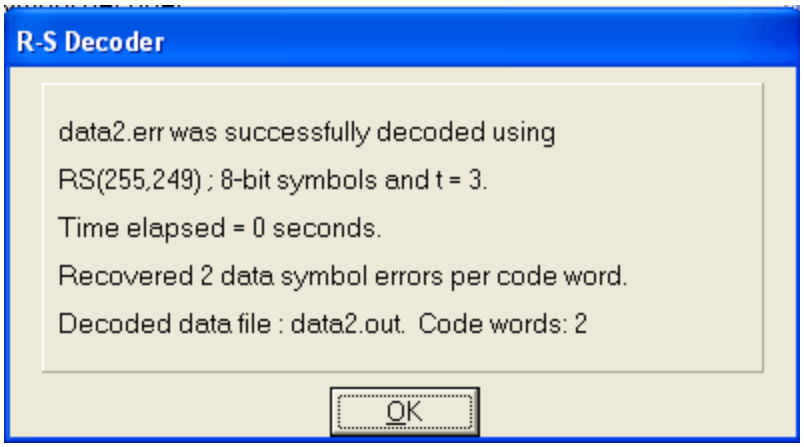




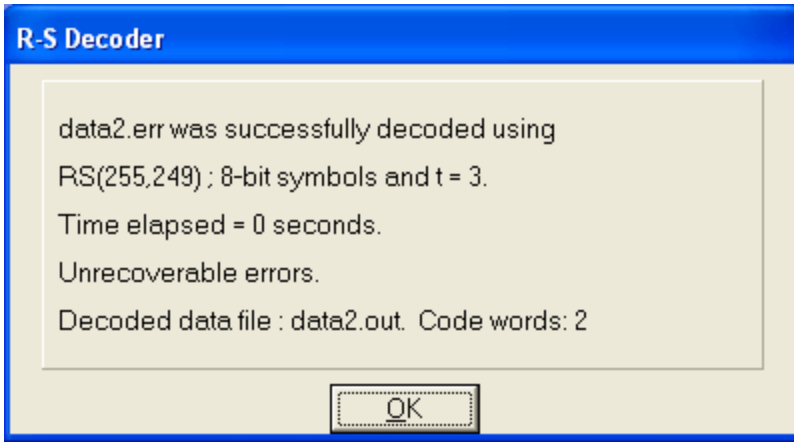
Será possível corrigir até 3 símbolos errados.

DECODE:

Até 3 erros (recuperável):

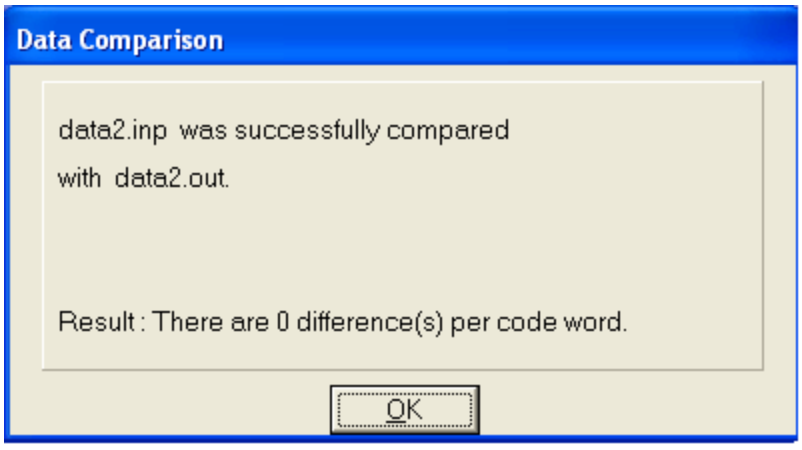


4 erros (não recuperável):

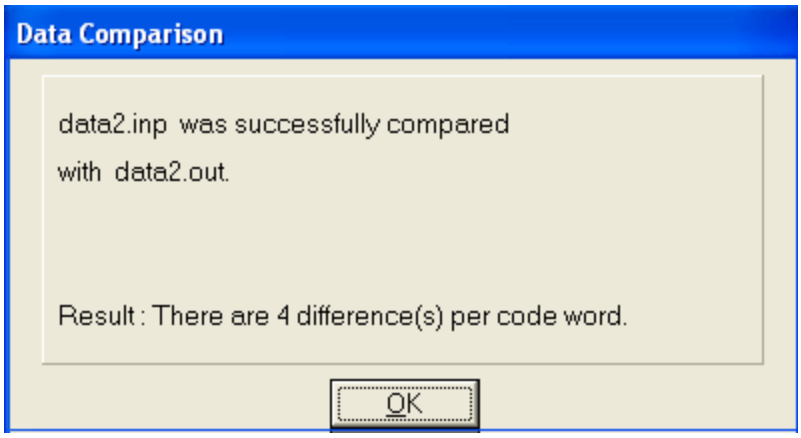


COMPARE:

Até 3 erros são corrigidos:



Com 4 erros não é corrígivel, mas é detectável:


Data Comparison
data2.inp was successfully compared
with data2.out.
Result : There are 4 difference(s) per code word.
OK