Detecção e correção de erros na transmissão de dados

Ian A. Escobar
Thiago P. Medina
Mateus A. B. G.
Lucca Dornelles

[1]Escola Politécnica – Pontifícia Universidade Católica (PUC-RS)

Resumo. *Este trabalho, feito para a Disciplina Integradora 1 do curso de Ciência da Computação da PUCRS, apresenta um estudo teórico e prático a cerca do problema da correção e detecção de erros na transmissão de dados. Será feita uma descrição do problema e da classe de problemas algorítmicos associado a ele, bem como, apresentaremos a sua relevância. Além disso, serão mostradas possíveis implementações que resolvam esse problema.*

1. Introdução

A correção e a detecção de erros na transmissão de dados é de grande importância para a computação moderna. Sua necessidade se dá ao pensarmos, por exemplo, na comunicação de sistemas em que erros não são toleráveis. Nesses sistemas, os erros que ocorrem durante a transmissão dos dados devem ser detectados e corrigidos, de forma que a informação que saia de uma das partes, chegue totalmente íntegra do outro lado.

1.1. Detecção de erros

A detecção de erros pode ser realizada utilizando por exemplo o algoritmo checksum. Nele, é gerado um valor numérico que é, basicamente, uma compressão irreversível do dado recebido. NO algoritmo de checksum, há várias operações computacionais(AND, XOR, Rot, adição binária, OR, NOT, shift, etc) seguidas em vários blocos de dados, é então aplicado ao dado recebido e o resultado é comparado com o checksum recebido. Mudanças mínimas no dado causam um grande impacto no checksum resultante, permitindo que sejam usados tanto para pequenos erros quanto grandes rajadas de erros.
Outros algoritmos bastante utilizados são o MD5 e o SHA-1. Ambos são considerados inseguros, devido a possibilidade de códigos modificados intencionalmente poderem gerar o mesmo resultado, porém ainda são úteis para detectar erros não intencionais. Os algoritmos novos SHA-2 e SHA-3 são considerados seguros contra certos tipos de ataques, porém são mais computacionalmente intensivos que os outros.

1.2. Correção de Erros

Os algoritmos de correção de erros trabalham com fragmentos redundantes dos dados que deverão então serem usados para detectar e corrigir erros nos dados.

Há basicamente 2 tipos de erros que podem ocorrer na transmissão dos dados: erros simples, aonde há a mudança de 1 bit na informação transmitida e rajada de erros, aonde múltiplos bits são transmitidos com erros. Existem algoritmos tanto para a detecção destes erros, como para a correção deles. Serão abordados algoritmos destes 2 tipos, porém daremos ênfase na correção do erro em si.

As próximas seções serão dividades da seguinte forma: na seção 2 será mostrada uma primeira visão sobre o problema computacional, apresentado também os tipos de erros que podem ocorrer na transmissão de dados. Na seção 3 explicaremos em que classe de problemas algorítmicos a questão da detecção e correção de erros está inserida. Na seção 4 serão mostradas situações onde a detecção e correção de erros é importante. Na seção 5 serão mostradas as estruturas que servem de pilar para o problema da detecção e correção de dados. Por fim, na seção 6 apresentaremos uma possível implementação para resolvermos o problema abordado neste trabalho.

2. Problema Computacional

O problema abordado neste trabalho é: quando ocorre uma transmissão de dados computacionais, há a possibilidade da perda de dados e alterações feitas nas informações, portanto, temos como objetivo principal o desenvolvimento de um algoritmo que detecta e corrige esses eventuais problemas.
Sabe-se que há dois tipos de erros a serem detectados, ambos realizam troca de valores binários na transmissão:

- Erro Simples: É a troca de um único valor binário isolado na transmissão de dados,ou seja, o valor binário originalmente enviado é recebido com a troca de um dos bits originais,possuindo,agora, um valor totalmente diferente do que o desejado.
- Rajada de erros: É uma sequência de trocas de valores binários dentro de um intervalo de tamanho n, ou seja, o valor originalmente enviado é recebido com um valor relevantemente alterado, não sendo o valor original.

3. Classe de Problemas Algorítmicos

Os algoritmos de detecção e correção de dados tem sua base na codificação e decodificação de dados, bem como, é comum a utilização de codificação linear nestes algoritmos. Levando em conta estes fatos, foi provado no artigo **On the inherent intractability of certain coding problems** [Berlekamp et al. 1978] que a decodificação de códigos lineares é um problema NP-completo, isto é, ele é verificável em tempo polinomial

4. Aplicabilidade do Problema

A detecção e correção de erros é importante em diversos contextos, principalmente aonde há criticidade na transmissão das informações entre sistemas. Porém, mesmo em sistemas não-críticos, erros em dados transmitidos podem gerar diversos problemas e defeitos nas aplicações. Separamos as aplicabilidades da detecção e correção de problemas na transmissão de dados em 3 grandes grupos: Sistemas que interpretam dados e tomam decisões, Telecomunicações e Exemplos gerais.

4.1. Sistemas que interpretam dados e tomam decisões

O sensor de airbag de um veículo é um bom exemplo da relevância do problema que estamos trabalhando. O sensor está recebendo dados a todo instante e é evidente que estes dados não podem apresentar erros, senão há a chance de, quando necessitarmos do acionamento automático do airbag, por exemplo, este falhe.

Um outro exemplo são os processos de decisão markovianos, aonde há sistemas que tomam decisões baseadas em análises de dados por meio de cadeias de markov. A importância das cadeias de markov, é mostrada em diversos estudos, como por exemplo em **"Cadeias de markov: Aplicações no cotidiano"** [Almeira 2015].

4.2. Telecomunicações

Citamos como bom exemplo de sistemas de telecomunicações que exigem tratamento dos dados enviados os satélites e sondas espaciais. Estes estão constantemente enviando dados a nós, e estes dados, até por causa de toda a distância envolvida nessa operação, podem apresentar falhas durante o caminho deles. Assim, a detecção e correção de problemas é bastante necessária nesse tipo de comunicação de sistemas.

Em telefonia/TV, também é ideal que os dados transmitidos estejam o mais íntegros o possível, com o objetivo de haver a maior qualidade o possível no oferecimento dos serviços.

4.3. Outros exemplos

Em transferências bancárias, a integridade das informações enviadas e recebidas deve ser total, de forma que, ao por exemplo, fazermos uma transferência de R$ 100,00 para outra pessoa, não haja a possibilidade de ser descontado um outro valor, a não ser aquele esperado.

Em Download/Upload de arquivos, também é verificada a relevância da detecção e correção de dados, pensando no fato de que um arquivo corrompido, seja por qualquer ruído que tenha acontecido na comunicação das partes envolvidas corrompa este arquivo e inutilize-o;

5. Definição de Estruturas

Há diversos algoritmos que funcionam para resolver o problema da detecção e correção de erros na transmissão de dados, por exemplo, os códigos de Reed–Solomon procuram agrupar os fragmentos de um dado em "símbolos".
Originalmente, em 1960, o objetivo do código era criar um polinômio P e uma sequencia A. A sequencia poderia ser qualquer sequencia numérica linear ou geométrica. O polinômio é então aplicado a cada elemento da sequência, sendo cada resultado equivalente a um símbolo.
O polinômio pode então ser utilizado com a sequência para recalcular os valores. O decodificador de Berlekamp Welch, por exemplo, utiliza eliminação gaussiana para recalcular o polinômio e o dado recebido, e então aplica o polinômio a sequência, para recalcular os dados, dado T símbolos, corrigindo até o piso de T/2 erros e detectando até T erros.

O algoritmo de Hamming, o qual será o utilizado neste trabalho, foi criado em 1950. Ele trabalha com bits de paridade, um checksum de um bit. Isso permite que qualquer erro de 1 bit possa ser corrigido e todos erros de um ou dois bits sejam detectados, sendo relativamente simples e não requerendo um grande poder de processamento, porém gera um grande custo de espaço, aumentando o tamanho do dado em até 75% , e um aumento de vulnerabilidade a rajadas de erros.

O código de Hamming (7,4) é um dos mais famosos, sendo que nele a informação transmitida apresenta 7 bits, sendo 4 de dados e 3 de paridade. Os bits de paridade são

adicionados na primeira, segunda e quarta posições da informação a ser transmitida, isto é, nas posições que são potências de 2. O restante dos bits da informação a ser transmitida são os bits de dados.

5.1. Exemplo de funcionamento do código de Hamming (7,4)

Para a transmissão dos dados 1001, devem ser adicionados 3 bits de paridade, como visto anteriormente, nas posições 1, 2 e 4 da informação de 7 bits que será transmitida.

Tabela 1. Transmissão 1001

Posição:	1	2	3	4	5	6	7
Bits:	P1	P2	1	P3	0	0	1

Os P's são os bits de paridade adicionados à palavra. O próximo passo agora é entender que cada bit de paridade fica responsável por um conjunto de bits da palavra transmitida. Cada um dos bits de paridade, está em uma posição da palavra a ser transmitida que apresenta apenas 1 bit ligado. O bit de paridade P1, está na posição 1 da palavra, ou ainda, na posição 001, se considerarmos as posições em números binários, logo, por apresentar o 3º bit ligado, ele estará responsável pelos outros bits que apresentam o 3º bit ligado. Assim, o bit P1 fica responsável pelas posições 1, 3, 5 e 7 da palavra que será transmitida. Os bits P2 e P3 seguem a mesma lógica, sendo o P2 responsável pelas posições 2, 3, 6, 7 e P3 pelas posições 4, 5, 6, 7.

Tabela 2. Posições em binário

Posição:	001	010	011	100	101	110	111
Bits:	P1	P2	1	P3	0	0	1

O próximo passo é fazer o cálculo das paridades, ou seja, descobrir se devemos inserir 0 ou inserir 1 em cada um dos bits de paridade. Como explicado anteriormente, cada um dos bits de paridade ficou responsável por um conjunto de bits, o que devemos fazer agora é somar o valor desses bits como se fossem números inteiros, e ao chegarmos a soma de cada um deles, verificamos se o resultado deu um número par, em caso positivo, substituímos esse bit de paridade por 0, em caso negativo, substituímos por 1. É importante lembrar que as posições aonde estão os bits de paridade tem valor 0.

Tabela 3. Cálculo dos bits de paridade

Bits de paridade	P1	P2	P3
Soma dos bits:	0 + 1 + 0 + 1	0 + 1 + 0 + 1	0 + 0 + 0 + 1
Resultado final	2 ->Par ->0	2 ->Par ->0	1 ->Ímpar ->1

Descobrimos que os bits de paridade P1, P2 e P3 tem os valores 0, 0 e 1 respectivamente. Dessa maneira, concluímos que a palavra a ser enviada deve ser 0011001.

O receptor, ao receber a palavra 0011001, deverá fazer o caminho inverso e verificar se os bits de paridade estão corretos.

Caso a palavra fosse modificada no caminho e fosse transmitida com erro, por exemplo, ao invés de ser transmitido 0011001 fosse transmitido 0001001, ao calcularmos as paridades dos bits de paridade, encontraríamos erros, como é facilmente observado nas tabelas abaixo.

Tabela 4. Recepção de 0001001

Posição:	1	2	3	4	5	6	7
Bits:	0(P1)	0(P2)	0	1(P3)	0	0	1

Tabela 5. Cálculo dos bits de paridade com erro

Bits de paridade	P1	P2	P3
Soma dos bits:	0 + 0 + 0 + 1	0 + 0 + 0 + 1	0 + 0 + 0 + 1
Resultado final	1 ->Ímpar ->1	1 ->Ímpar ->1	1 ->Ímpar ->1

Assim, é detectado erro pois, a informação recebida da palavra 0001001 apresenta como bits de paridade os valores 0, 0 e 1, porém, após os cálculos, chegamos aos valores de 1, 1 e 1 para os bits de paridade, apresentando uma contradição. Essa contradição indica que há erro na transmissão dos dados.

6. Implementação

Ainda não há nada para ser posto aqui

Referências

Almeira, M. A. (2015). Cadeias de markov: Aplicações no cotidiano. *Caderno de Graduação - Ciências Exatas e Tecnológicas - UNIT*.

Berlekamp, E., McEliece, R., and van Tilborg, H. (1978). On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*.

Forouzan, B. A. (2013). *Data Communications and Networking*. McGraw-Hill.