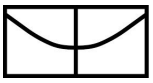

Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

Trabalho 3

Teleinformática e Redes 1

Simulador Manchesto 3.0

André Filipe da Conceição | 150005547
Gabriel Matheus da Rocha de Oliveira | 170103498
Guilherme Braga Pinto | 170162290

Brasília
2020

Introdução

O objetivo deste trabalho é, baseando-se no trabalho previamente executado relativo à simulação de Camada Física e Camada de Enlace (respectivamente Trabalhos 1 e 2 da matéria de TR1), implementar o controle de erros na comunicação entre parte transmissora e parte receptora em uma simulação de comunicação.

O funcionamento do simulador é semelhante ao do trabalho 2 porém ao invés da mensagem ser transmitida diretamente da Camada de Aplicação Transmissora para a Camada de Enlace de dados transmissora passando apenas pelos algoritmos de enquadramento, antes da mesma passar pelos algoritmos descritos ela passa pelos protocolos de controle de erro: Bit de paridade par, Bit de paridade ímpar, CRC e código de Hamming. O mesmo acontece no processo de recepção onde a mensagem passa pelo processo de controle de erro após ser transmitida da Camada física receptora antes de passar pelo processos de enquadramento.

Durante a execução do simulador, se o processo de controle de erro não identificar um erro durante a transmissão de uma mensagem o mesmo informa ao usuário por meio do terminal que nenhum erro foi detectado e o simulador continua sua execução normalmente. Caso o controle de erro identifique um erro durante a transmissão da mensagem o mesmo também informa ao usuário sobre a situação porém o simulador para seu funcionamento imediatamente.

O meio de comunicação presente na Camada Física também foi modificado para simular a geração de erros durante uma transmissão de dados. O meio apresenta três formas de geração de erros sendo eles: “Modo Caos”, “Modo Conservador” e “Modo Padrão”, que serão melhor descritos na seção de implementação. Nos três modos citados é informado no terminal para o usuário quais posições de bits foram modificados e que modificações foram realizadas para a geração dos erros.

Descrição dos protocolos de controle de erro citados:

- **Inserção de Bit de Paridade Par -** O protocolo busca identificar erros durante a transmissão de mensagens a partir da inserção de um bit adicional ao fim da mensagem que será transmitida. Durante o processo de transmissão da mensagem, o protocolo adiciona um bit 0 ou 1 ao final da mensagem de acordo com a quantidade de bits 1 presentes na mesma. O objetivo da inserção de bit de paridade par é a partir desse bit adicional garantir que a quantidade de bits 1 presentes na mensagem tenha a paridade par, dessa forma, se a mensagem possuir uma quantidade ímpar de bits 1, o bit adicionado será 1 e se a mensagem possuir uma quantidade par, o bit 0 será adicionado. No processo de recepção é verificado se a quantidade total de bits 1 presentes na mensagem manteve a paridade par, se sim, a mensagem foi transmitida com êxito, caso contrário, um ou mais bits foram modificados durante a transmissão o que caracteriza a presença de erro no envio.

- **Inserção de Bit de Paridade Ímpar -** O protocolo possui um funcionamento muito similar ao protocolo de inserção de bits par, com a única diferença que o bit adicional inserido no final da mensagem objetiva tornar a quantidade total de bits 1 presentes

na mensagem em um valor ímpar. Dessa forma, para garantir a paridade desejada, se a mensagem possuir uma quantidade ímpar de bits 1, o bit adicionado será 0 e caso o contrário, o bit adicionado será 1.

- **Erro CRC -** O código de redundância cíclica (CRC) utiliza um polinômio gerador G(x) que possui mais quadros que a mensagem a ser transmitida e o bit de mais alta e baixa ordem iguais a 1 para identificar presença de erro durante uma transmissão. A mensagem a ser transmitida é dividida pelo polinômio gerador e logo após isso, o resto da divisão é somado a mensagem e a mesma é enviada. Para identificar se houve ou não erro na transmissão o receptor divide a mensagem pelo mesmo polinômio gerador que foi utilizado na transmissão e verifica se o resultado é 0 ou não, caso seja, a mensagem foi transmitida com êxito, caso contrário, ocorreu um erro durante a transmissão.

- **Código de Hamming -** O protocolo adiciona bits verificadores a mensagem a ser transmitida nas posições das potências de 2 com o objetivo de não apenas detectar mas também corrigir erros de transmissão. As posições dos bits da mensagem original são decompostas na soma de potências de 2 (exemplo: posição 11 = 8 + 2 + 1). Cada bit verificador, dessa forma, será considerado a soma de informações acerca de cada respectivo termo resultante da decomposição de todas as posições da mensagem. Para cada bit de verificação o seu valor será definido como 0 ou 1 de forma que a soma dos bits 1 presentes na soma de informações que armazena seja de paridade par. No receptor, para verificar a existência ou não de erros, é calculado novamente a paridade em todos os bits verificadores, se todos os resultados forem 0 não houve erro, caso contrário, é confirmado a presença de um erro. O código de Hamming, diferente dos protocolos citados anteriormente, é capaz de corrigir erros encontrados e para isso calcula a posição do erro identificando qual bit deve ser modificado. O cálculo é bem simples e consiste em encontrar a posição a partir do cálculo já realizado para encontrar a paridade dos bits verificadores e os ler de trás para frente para formar um número binário. O Número ao ser convertido para decimal informa a posição desejada e para corrigir o erro basta inverter o bit que se encontra naquela posição.

Exemplos de resultado de execução do simulador
SEM PRESENÇA DE ERRO

```
Aplicacao Transmissora:
Digite uma mensagem:
TR1

Mensagem a ser transmitida: TR1

Camada de Aplicacao Transmissora
010101000101001000110001

Camada de Enlace de dados Transmissora
Contagem de Caracteres:

000001000101010001010010001100010100

Controle de erro: Erro CRC

000001000101010001010010001100010100

Camada Fisica Transmissora:
Codificacao Manchester:
0101010101100101011001100110010101100110010110010101101001010110011001100101

Meio de Transmissao
```

```
Meio de Transmissao

0101010101100101011001100110010101100110010110010101101001010110011001100101

Camada Fisica Receptora
Decodificacao Manchester:
000001000101010001010010001100010100

Camada de Enlade de dados receptora
Controle de Erro CRC:

Resto igual à zero: Não foi detectado nenhum erro.

000001000101010001010010001100010100

Desenquadramento da Contagem de Caracteres:

010101000101001000110001

Camada de Aplicação Receptora
010101000101001000110001

Aplicacao Receptora
A mensagem recebida foi: TR1
```

COM PRESENÇA DE ERRO

```
Aplicacao Transmissora:
Digite uma mensagem:
TR1

Mensagem a ser transmitida: TR1

Camada de Aplicacao Transmissora
010101000101001000110001

Camada de Enlace de dados Transmissora
Contagem de Caracteres:

000001000101010001010010001100010

Controle de erro: Erro CRC

000001000101010001010010001100010100

Camada Fisica Transmissora:
Codificacao Manchester:
010101010110010101100110011001010110011001011001010110100101011001100101

Meio de Transmissao
```

```
Meio de Transmissao

Erro simulado na posição 3: 1 -> 0

Erro simulado na posição 48: 0 -> 1

010001010110010101100110011001010110011001011001110110100101011001100101

Camada Fisica Receptora
Decodificacao Manchester:
010001000101010001010010001100010100

Camada de Enlade de dados receptora
Controle de Erro CRC:

Resto diferente de zero: Um erro foi detectado!
```

Implementação

Decisões relativas ao desenvolvimento

- **Técnica de Desenvolvimento**

Utilizou-se reuniões remotas para a melhor comunicação entre membros da equipe, além do uso do Github para o compartilhamento de artefatos criados no desenvolvimento do Trabalho. A modularização das funções permitiu que o trabalho fosse eventualmente executado de maneira individual e paralela, apenas quando necessário. O repositório utilizado se encontra privado no momento da entrega deste trabalho.

- **Implementação do Erro CRC**

A divisão por meio de operações de XOR ocorre após o vetor do que se está adicionando o controle de erro é movido à esquerda com a adição de 3 números zeros. No final, o resto é adicionado neste espaço criado e repassado, mantendo assim a carga útil da mensagem intacta. Caso o resto fosse adicionado na carga útil, a mensagem alterada seria perdida de qualquer forma por parte do receptor, que não saberia que quantidade subtrair o fluxo de bits recebido para obter a mensagem original. O receptor então deve, executar a divisão e retirar da mensagem a ser repassada o espaço de bits usado para a transmissão do resto. A parte com a transmissão do resto é útil unicamente para fins de checagem de controle de erros.

- **Inserção de Bit de Paridade**

A fim de se satisfazer a implementação feita previamente relativa à Camada de Enlace e à Camada Física, a inserção de bit de paridade (seja par ou ímpar) resultou em um erro caso a carga adicionada ao pacote após passar pela Camada Física e pela Camada de Enlace fosse de exatamente 1 bit. Isso ocorre devido à natureza da implementação feita que está preparada para receber caracteres ASCII na forma de bits. Uma maneira de contornar este problema e todavia satisfazer a teoria de Inserção de Bit de Paridade Par ou Ímpar foi a inserção de um conjunto de 4 bits, simulando a inserção de um “1” ou “0” assim sendo “0000” ou “0001”. Na contagem da verificação de Paridade por parte do receptor, o cálculo se mantém o mesmo: determina-se a quantidade de bits “1” no programa para então definir se este número é par ou ímpar.

Outro aspecto da implementação de Inserção de Bits de Paridade é ressaltar que neste protocolo erros são apenas identificados e não tratados, diferente do Código de Hamming.

- **Convenção adotada relativa à vetores**

Neste trabalho vetores são usados e descartados com elevada frequência devido à agregação de headers e bits afins, então ao longo de todo trabalho o limite de um vetor é sempre dado pelo número "2" em seu final, se diferenciando do fluxo de bits onde há a ocorrência de apenas os números "0" e "1".

- **Update da Camada Física**

Identificou-se a necessidade de se atualizar detalhes de implementação relativos à Camada Física, que não estava pronta para receber e repassar erros devido à forma que as operações do tipo "XOR" são realizadas, por meio de comparações entre inputs e números. Se adicionou pequenas atualizações que resultaram que a Camada Física agora pode repassar erros, porém, eventualmente o protocolo usado não permite a recuperação da mensagem, logo por exemplo, no caso do Bit de Paridade Par.

- **Simulação de Erros de Transmissão**

Com a finalidade de se obter testes relativos à detecção de erros, a parte do programa que tem por objetivo simular o envio de pacotes entre uma parte Receptora e uma parte Transmissora foi alterada para se incluir a inversão de bits em momentos aleatórios, com um determinado grau de chance. Porém decidiu-se implementar três maneiras de se implementar um erro de pré-determinada chance: o “modo caos”, o “modo conservador” e o “modo padrão”. O “modo caos” gera um número aleatório entre 0 e 100 para cada bit sendo transmitido, e se o número gerado for menor do que o “fator de probabilidade” escolhido, o bit em questão é invertido. Por exemplo, se o “fator de probabilidade” escolhido foi de “10”, logo há 10 chances em 100 de que o número gerado aleatoriamente seja menor do que 10, resultando em 10% de chance para cada bit. Notou-se que praticamente sempre dessa forma um erro é gerado, assim o modo “conservador” foi implementado. Neste modo, a comparação do “fator de aleatoriedade” com o número gerado pelo programa ocorre apenas uma única vez por transmissão. Se a comparação for satisfeita e o erro é gerado, apenas 1 bit transmitido irá mudar. O local onde o bit é gerado é gerado também de maneira aleatória, assim o local do erro pode ser um valor entre 0 e o valor do tamanho do vetor sendo transmitido. O “modo padrão” é semelhante ao “modo caos” porém, com o objetivo de reduzir a quantidade de erro gerado, para ocorrer um erro é necessário que o valor aleatório gerado seja igual a chance de erro e não menor igual ao do modo caos. A geração de números aleatórios foi implementada por meio da função “rand()” do C++. Para melhores erros, o “modo padrão” é ***recomendado***.

- **Modos de Funcionamento**

A seleção do tipo de tratamento de erro a ser escolhido é controlado por meio de um “#define” no arquivo “CamadaEnlace.h”, e segue a seguinte convenção para a sua seleção, sendo independente de outros modos de operação:

Controle de Erros	**Código atrelado**
Paridade Par	0
Paridade Ímpar	1
Erro CRC	2
Código de Hamming	3

Por se tratar de um programa com controle de erros, erros podem ser simulados de duas maneiras, sendo elas personalizáveis por meio de um “#define” também no arquivo “CamadaEnlace.h”, seguindo a seguinte convenção:

Modo de criação de Erros	**Código atrelado**
Modo Caos	0
Modo Conservador	1
Modo Padrão	2

O funcionamento da seleção de cada tipo de enquadramento e desenquadramento pode ser controlado por meio de um “#define” no arquivo de header CamadaEnlace.h. É adotada a seguinte convenção para o funcionamento da Camada de Enlace:

Enquadramento/Desenquadramento	**Código atrelado**
Contagem de Caracteres	0
Inserção de Bytes	1
Inserção de Bits	2

O funcionamento para a configuração da Camada Física é análogo e independente, podendo ser alterado por meio do #define no arquivo CamadaFisica.h sem alterar o funcionamento da Camada de Enlace. Cada codificação e decodificação pode ser usada com cada enquadramento e desenquadramento fazendo as devidas combinações. É adotada a seguinte convenção:

Codificação/Decodificação	**Código atrelado**
Binária	0
Manchester	1
Manchester Diferencial	2

Membros

- **André Filipe:** implementação prática da correção e detecção de erros; revisão do trabalho.

- **Gabriel Matheus:** implementação prática da correção e detecção de erros; modularização e codificação da interface (terminal), desenvolvimento do código e documentação.

- **Guilherme Braga Pinto:** implementação prática da correção e detecção de erros; decisões de desenvolvimento; desenvolvimento do código; elaboração de comentários e documentação.

Conclusão

A implementação, por mais que incompleta, do simulador permitiu que o grupo obtivesse conhecimentos novos acerca do processo de detecção e correção de erros a partir do entendimento teórico e prático dos protocolos de Inserção de Bit de paridade par, Inserção de Bit de paridade ímpar, CRC e Código de Hamming. A adição desses novos protocolos na camada de Enlace de dados implementada no Trabalho 2 de TR1 permitiu que o simulador iniciado no Trabalho 1 se tornasse ainda mais próximo de um processo real de transmissão de dados porém, ainda simplificado em uma simulação para o melhor entendimento da disciplina.

Durante o desenvolvimento do programa a equipe se encontrou com falta de tempo e não foi possível implementar o simulador em sua totalidade. Dessa forma, de todos os protocolos descritos não foi possível implementar o código de Hamming, porém, o resultado obtido a partir do que foi implementado foi satisfatório dado as condições desfavoráveis da falta de tempo dos membros da equipe.

Em suma, devido a presença de obstáculos e problemas encontrados pela equipe o simulador proposto pelo Trabalho 3 não pode ser codificado por completo, mas a implementação do que foi possível e o estudo realizado para o entendimento de todos os protocolos em sua totalidade se tornaram experiências válidas para uma melhor compreensão dos tópicos abordados na matéria de Teleinformática e Redes 1.

O link para o repositório usado neste trabalho se encontra no seguinte link. Este repositório se encontra privado no momento da entrega deste trabalho.