Redes de computadores 1- Trabalho 2

Leonardo Becker de Oliveira – 20211779

Introdução

Em comunicações, os dados a serem transmitidos, estão constantemente sendo desafiados para durante o transporte do seu destino a sua origem, ou então, do cliente ao seu servidor, por diversos fatores. Com isso, se faz necessário a aplicação de técnicas, que tenham por finalidade, reduzir, corrigir ou eliminar, todos os erros de transmissões. Neste trabalho, será apresentado a forma detecção de erro “Cyclic redundancy check” em tradução livre “Verificação de redundância cíclica”, e a forma de correção dos erros “Multidimensional parity-check” em tradução livre “Verificação de paridade multidimensional”.

Detecção de erros

Existem diversas formas para se detectar um erro, em que a grande maioria funciona de forma semelhante, onde adiciona-se bits adicionais a sequência de dados, a fim de essa informação adicionada indicar se os dados recebidos foram mesmo corretamente recebidos ou não.

-Verificação de redundância cíclica:

Seu funcionamento se dá através de uma divisão de polinômios (utilizado o WCDMA, 0x9B em hexadecimal), então para cada sequência de bytes, é realizado sucessivas divisões binárias entre os dados e o polinômio, utilizando o resto da divisão para verificação de integridade. Portanto, para cada pacote, além dos n bytes de dados, é necessário um byte adicional no final, para inserção do CRC (apenas um byte pois o polinômio tem tamanho 8, o que gera então no máximo 7 bits de resto).

Foram desenvolvido em linguagem C duas funções, uma geradora do CRC, e outra para testar o CRC. As duas funções funcionam de maneira muito semelhante, em que dois laços de repetição aninhados percorrem todos os bits de um byte, por toda a extensão da mensagem (vetor de unsigned char).

Para cada repetição do loop, a sequência é sempre rotacionada em um bit à esquerda, percorrendo então todos os demais bits do vetor que vem a sua direita. Caso o bit mais significativo (MSB), for 1, realiza-se a operação de xor entre a sequência e o respectivo polinômio (visto como a operação de divisão anteriormente descrita). Chegando no fim do vetor, ou então, assim que zerar todos os bits do último byte de dados, obtém-se a sequência de no máximo 7 bits (sempre rotacionado para ficar com 8), que é o CRC inicialmente buscado.

A função testa CRC executa os mesmos passos da função geradora, tirando que para o último byte, ao invés da rotação retornar 0, ele passa a retornar os bits do valor CRC. Ao final do processo, se todos os bits estiverem zerados, os dados foram transmitidos da maneira correta, caso contrário é necessário solicitar o reenvio (retorno NACK) pois algo deu errado.

Retorna CRC:

```
for (i = 0; i < TAM; i++)
{
for (j = 0; j < CRC; j++)
{
if (((seq >> 7) & (0x01)) == 1)
{
            seq = seq ^ POL;
}
if (conta_bit >= (TAM * CRC - CRC)) {
            seq_crua = (seq >> j);
            if ((seq_crua == 0)) {
            return seq << (CRC - j);
} }
if (i == (TAM - 1))
            proximo_bit = 0;
else
            proximo_bit=(dados[i+1]>>((CRC-1)-j)) & (0x01);
seq = seq << 1;
if (proximo_bit)
            seq += 1;
conta_bit++; }}
```

Testa CRC:

```
for (i = 0; i < TAM; i++)
{
for (j = 0; j < CRC; j++)
{
if (((seq >> 7) & (0x01)) == 1)
{
            seq = seq ^ POL;
}
proximo_bit=(dados[i+1]>>((CRC-1)-j)) & (0x01);
seq = seq << 1;
if (proximo_bit)
            seq += 1;
conta_bit++; } }
if (seq == 0)
            return 1;
else
            return 0;
```

Correção de erros

A correção de erros, entra em ação após a detecção de erros, pois os bits adicionais para detecção de erro, anteriormente descritos, podem servir também para indicar o bit exato em que aconteceu, deixando de ser necessário o reenvio do pacote uma vez que pode ser facilmente corrigido. A forma abordada e desenvolvida nesse projeto foi a verificação de paridade multidimensional.

- Verificação de paridade multidimensional:

Seu funcionamento se da através da através da adição de bits de paridade na horizontal e na vertical, formando uma espécie de matriz byte a byte em que os extremos são os bits de paridade. Ou seja, para uma sequência de 8 bytes, é necessário uma matriz 9 por 9, uma vez que todos os 9 bits da 9º coluna são reservados a paridade horizontal, e o 9º byte é responsável por guardar a paridade de todos os outros 8 bytes anteriores na vertical.

Desenvolvido em linguagem C, o código tem por objetivo inserir esse bits de verificação, e através de uma falha de transmissão forjada (inserida via código), o programa tem por objetivo identificar e possivelmente corrigir o erro. Importante destacar que esse método não apresenta resultados muito satisfatório, uma vez que ao identificar o erro em um byte, somente a identificação da linha não é o suficiente para dizer exatamente qual dos 8 bits estão errados. Além disso, caso múltiplos de dois números na linha e coluna, venham a ser transmitido errado, o erro não chega nem a ser detectado, uma vez que a paridade permanece correta.

Para adicionar a paridade horizontal, percorre-se todos os 8 bits do byte, por todos os bytes da sequência, somando os valores do bit, no final é feita a verificação da soma, resultados ímpares indicam que o bit de paridade deve ser 1 e resultados pares resultam em paridades 0.

De forma semelhante, na paridade vertical, é percorrido todas as linhas coluna por coluna, e a verificação de par ou ímpar, resulta no valor que deveria ser a paridade, multiplicado pelo valor binário da coluna, gerando no final apenas uma soma, sendo o nono byte de verificação. Ao transformar essa soma em binário, todos os seus bits variam entre 0 e 1 de acordo com a paridade vertical dos 8 bytes anteriores.

Para encontrar os erros, é necessário percorrer todas as linhas novamente, idêntico a forma de gerar os bits de paridade, mas agora ao invés de alterar o valor de paridade, é comparado o valor recalculado (recalcular e comparar fica a cargo do recebedor da transmissão) com o anteriormente inserido. Feito isso, se as paridades vierem a divergir significa que há um erro naquela linha, ou coluna. Quando identificado a linha e a coluna do erro, é possível alterar o valor do bit deixando de acordo com o enviado, mas caso haja a identificação só da linha, ou só da coluna, o erro acaba passando direto, não sendo possível tratar como anteriormente descrito.

Código gerador da paridade vertical:

```
for (j = 0; j < 8; j++)
{
counter_bit = 0;
for (i = 0; i < TAM; i++)
{
        bit = (matrizMensagem[i] >> (7 - j)) & (0x01);
        if (bit)
                counter_bit++;
        }
        if ((counter_bit % 2) == 0)
                soma = soma;
        else
                soma += pow(2, (7 - j));
}
elemento[TAM].caractere = soma;
}
```

Código para correção de erros:

```
for (i = 0; i < contaErro; i++){
        linha = erros[i][0];
        coluna = erros[i][1];
        soma = elemento[linha].caractere;
        bit = ((elemento[linha].caractere) >> (7 - coluna))
& (0x01);
        if (bit)
                soma = soma - pow(2, (7 - coluna));
        else
                soma = soma + pow(2, (7 – coluna));
        elemento[linha].caractere = soma;
}
```

Código para verificação de erros:

```
for (j = 0; j < 8; j++){
counter_bit = 0;
for (i = 0; i < TAM; i++)   {
        bit = ((elemento[i].caractere) >> (7 - j)) & (0x01);
        if (bit)
                counter_bit++;
        }
        if ((counter_bit % 2) == 0){
                if  ((((elemento[TAM].caractere)  >>  (7 -
j)) & (0x01)) != 0){
                erroHorizontal(elemento,    j,    erros,
&contaErro);
        } }
        else{
                if  ((((elemento[TAM].caractere)  >>  (7 -
j)) & (0x01)) != 1){
                erroHorizontal(elemento,    j,    erros,
&contaErro); } }
}
```

Conclusão

Conclui-se então que é extremamente importante a existência de uma forma de verificação e correção de dados numa comunicação, pois sem isso, nunca seria possível deduzir se o outro lado de fato recebeu, e se recebeu da forma correta, podendo existir transmissões completamente ambíguas acarretando em outros problemas. Além disso a correção de dados agiliza a troca de informações, uma vez que deixa de ser necessário reenviar toda a mensagem, visto que foi possível corrigir a mensagem ao receber.