

Trabalho Prático: Camada de Enlace de Dados

Controle de Erro

⇒ Descrição

✓ Acrescentar ao código do “simulador de redes” os protocolos vistos para o **controle de erro** da informação

- Bit de paridade par
- Bit de paridade ímpar
- CRC
- Código de Hamming

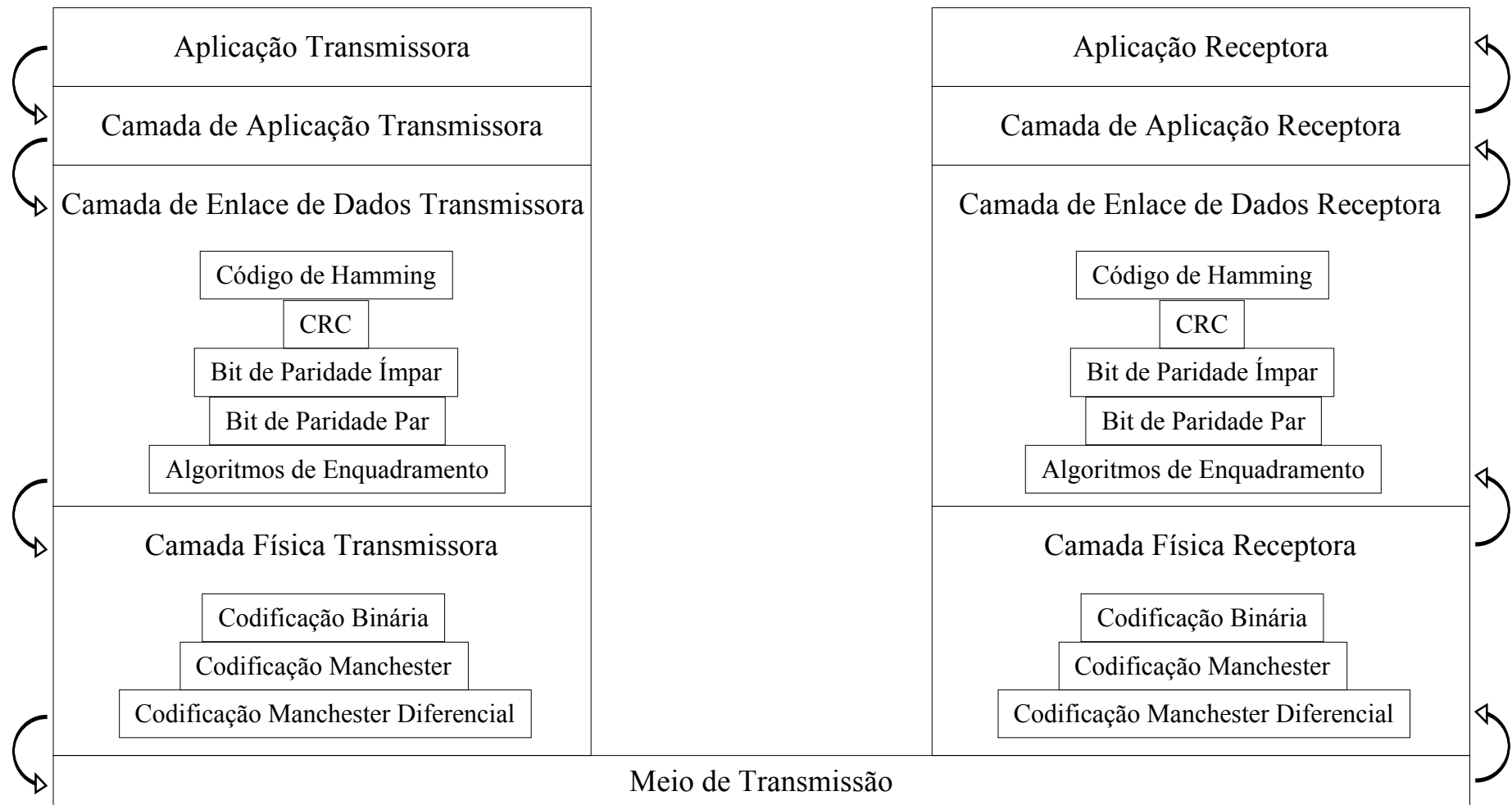
⇒ Para poder testar o controle de erros, não esquecer de alterar o código de geração de erros na camada física

✓ Complementar o método **MeioDeComunicacao**

Trabalho Prático: Camada de Enlace de Dados

Controle de Erro

⇒ Diagrama



Trabalho Prático: Camada de Enlace de Dados

Controle de Erro - Transmissão

```
void CamadaEnlaceDadosTransmissora (int quadro []) {  
    //algum codigo aqui  
} //fim do metodo CamadaEnlaceDadosTransmissora
```

```
void CamadaEnlaceDadosTransmissoraEnquadramento (int quadro []) {  
    //algum codigo aqui  
} //fim do metodo CamadaEnlaceDadosTransmissoraEnquadramentos
```

```
void CamadaEnlaceDadosTransmissoraControleDeErro (int quadro []) {  
    //algum codigo aqui  
} //fim do metodo CamadaEnlaceDadosTransmissoraControleDeErro
```

Trabalho Prático: Camada de Enlace de Dados

Controle de Erro - Transmissão

```
void CamadaEnlaceDadosTransmissora (int quadro []) {  
  
    CamadaEnlaceDadosTransmissoraEnquadramento(quadro)  
    CamadaEnlaceDadosTransmissoraControleDeErro(quadro);  
  
    //chama proxima camada  
    CamadaFisicaTransmissora(quadro);  
  
} //fim do metodo CamadaEnlaceDadosTransmissora
```

Trabalho Prático: Camada de Enlace de Dados


Controle de Erro - Transmissão

```
void CamadaEnlaceDadosTransmissoraControleDeErro (int quadro []) {  
    int tipoDeControleDeErro = 0; //alterar de acordo com o teste  
    switch (tipoDeControleDeErro) {  
        case 0 : //bit de paridade par  
            //codigo  
            break;  
        case 1 : //bit de paridade impar  
            //codigo  
            break;  
        case 2 : //CRC  
            //codigo  
        case 3 : //codigo de Hamming  
            //codigo  
            break;  
    } //fim do switch/case  
} //fim do metodo CamadaEnlaceDadosTransmissoraControleDeErro
```

Trabalho Prático: Camada de Enlace de Dados

Controle de Erro - Transmissão

```
void CamadaEnlaceDadosTransmissoraControleDeErroBitParidadePar (int quadro []) {  
    //implementacao do algoritmo  
} //fim do metodo CamadaEnlaceDadosTransmissoraControleDeErroBitParidadePar  
  
void CamadaEnlaceDadosTransmissoraControleDeErroBitParidadeImpar (int quadro [])  
{  
    //implementacao do algoritmo  
} //fim do metodo CamadaEnlaceDadosTransmissoraControleDeErroBitParidadeImpar  
  
void CamadaEnlaceDadosTransmissoraControleDeErroCRC (int quadro []) {  
    //implementacao do algoritmo  
    //usar polinomio CRC-32(IEEE 802)  
} //fim do metodo CamadaEnlaceDadosTransmissoraControleDeErroCRC  
  
void CamadaEnlaceDadosTransmissoraControleDeErroCodigoDeHamming (int quadro []) {  
    //implementacao do algoritmo  
} //fim do metodo CamadaEnlaceDadosTransmissoraControleDeErroCodigoDeHamming
```



Trabalho Prático: Camada de Enlace de Dados

Meio de Comunicação

PROVOCAR O ERRO!

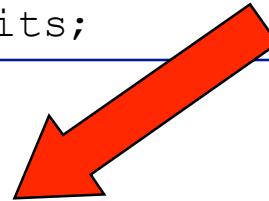
```
void MeioDeComunicacao (int fluxoBrutoDeBits []) {  
    //OBS: trabalhar com BITS e nao com BYTES!!!  
    int erro, porcentagemDeErros;  
    int fluxoBrutoDeBitsPontoA [], fluxoBrutoDeBitsPontoB [];
```

```
    porcentagemDeErros = 0; //10%, 20%, 30%, 40%, ..., 100%  
    fluxoBrutoDeBitsPontoA = fluxoBrutoDeBits;
```

```
    while (fluxoBrutoDeBitsPontoB.lenght!=  
           fluxoBrutoDeBitsPontoA) {
```

```
        if ((rand()%100)== ... ) //fazer a probabilidade do erro  
            fluxoBrutoBitsPontoB += fluxoBrutoBitsPontoA; //BITS!!!  
        else //ERRO! INVERTER (usa condicao ternaria)  
            fluxoBrutoBitsPontoB==0) ?  
            fluxoBrutoBitsPontoA=fluxoBrutoBitsPontoB++ :  
            fluxoBrutoBitsPontoA=fluxoBrutoBitsPontoB--;
```

```
    } //fim do while  
} //fim do metodo MeioDeTransmissao
```



Trabalho Prático: Camada de Enlace de Dados

Controle de Erro - Recepção

```
void CamadaEnlaceDadosReceptora (int quadro []) {  
    //algum codigo aqui  
} //fim do metodo CamadaEnlaceReceptora
```

```
void CamadaEnlaceDadosReceptoraEnquadramento (int quadro []) {  
    //algum codigo aqui  
} //fim do metodo CamadaEnlaceDadosReceptoraEnquadramento
```

```
void CamadaEnlaceDadosReceptoraControleDeErro (int quadro []) {  
    //algum codigo aqui  
} //fim do metodo CamadaEnlaceDadosReceptoraControleDeErro
```


Trabalho Prático: Camada de Enlace de Dados

Controle de Erro - Recepção

```
void CamadaEnlaceDadosReceptora (int quadro []) {  
  
    CamadaDeEnlaceTransmissoraEnquadramento(quadro)  
    CamadaDeEnlaceTransmissoraControleDeErro(quadro);  
  
    //chama proxima camada  
    CamadaDeAplicacaoReceptora(quadro);  
  
} //fim do metodo CamadaEnlaceDadosReceptora
```

Trabalho Prático: Camada de Enlace de Dados

Controle de Erro - Recepção

```
void CamadaEnlaceDadosReceptoraControleDeErro (int quadro []) {  
    int tipoDeControleDeErro = 0; //alterar de acordo com o teste  
    switch (tipoDeControleDeErro) {  
        case 0 : //bit de paridade par  
            //codigo  
            break;  
        case 1 : //bit de paridade impar  
            //codigo  
            break;  
        case 2 : //CRC  
            //codigo  
        case 3 : //codigo de hamming  
            //codigo  
            break;  
    } //fim do switch/case  
} //fim do metodo CamadaEnlaceDadosReceptoraControleDeErro
```

Trabalho Prático: Camada de Enlace de Dados

Controle de Erro - Recepção

```
void CamadaEnlaceDadosReceptoraControleDeErroBitDeParidadePar (int quadro []) {  
    //implementacao do algoritmo para VERIFICAR SE HOUVE ERRO  
} //fim do metodo CamadaEnlaceDadosReceptoraControleDeErroBitDeParidadePar  
  
void CamadaEnlaceDadosReceptoraControleDeErroBitDeParidadeImpar (int quadro []) {  
    //implementacao do algoritmo para VERIFICAR SE HOUVE ERRO  
} //fim do metodo CamadaEnlaceDadosReceptoraControleDeErroBitDeParidadeImpar  
  
void CamadaEnlaceDadosReceptoraControleDeErroCRC (int quadro []) {  
    //implementacao do algoritmo para VERIFICAR SE HOUVE ERRO  
    //usar polinomio CRC-32 (IEEE 802)  
} //fim do metodo CamadaEnlaceDadosReceptoraControleDeErroCRC  
  
void CamadaEnlaceDadosReceptoraControleDeErroCodigoDeHamming (int quadro []) {  
    //implementacao do algoritmo para VERIFICAR SE HOUVE ERRO  
} //fim do metodo CamadaEnlaceDadosReceptoraControleDeErroCodigoDeHamming
```

