

Trabalho de camada física

TR1

Marcelo A. Marotta



Departamento de Ciência da Computação
Universidade de Brasília



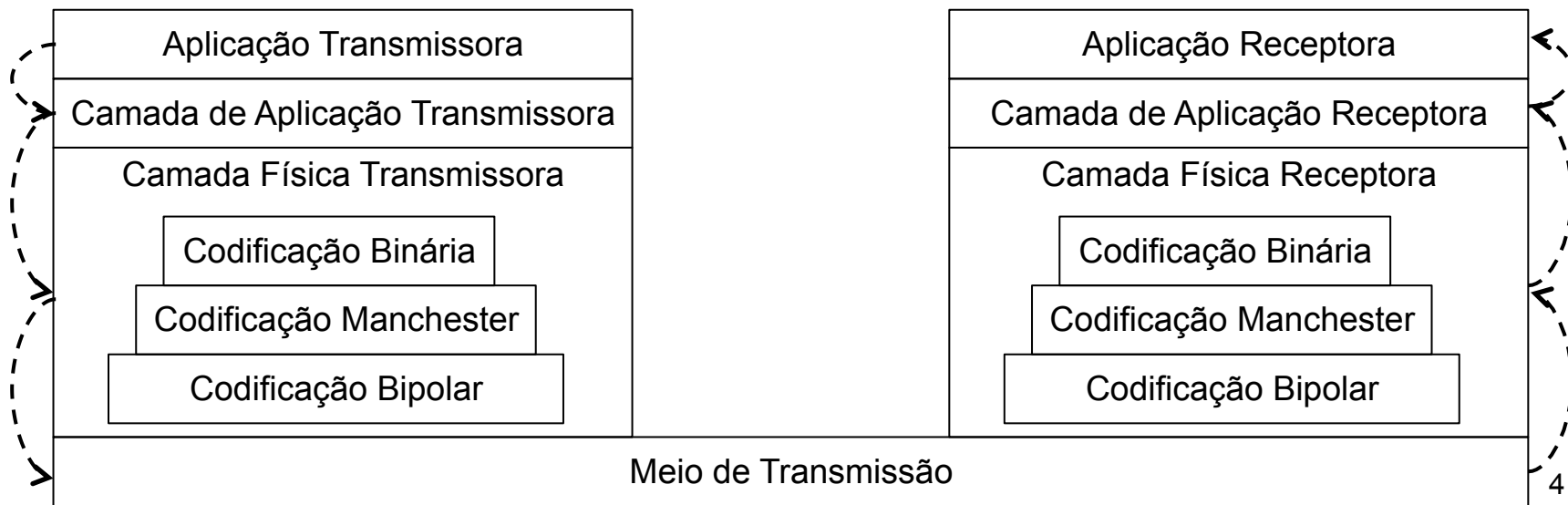
- Simular o funcionamento do enlace físico por meio da implementação das seguintes codificações:
 - Binária;
 - Manchester; e
 - Bipolar.



- Principais características:
 - Utilizar GUI e apresentar o processamento de toda informação;
 - Seguir a estrutura especificada (ver diagrama e codificação nos próximos slides); e
 - Manipular Bits (e não bytes) nos níveis mais baixos.



- Trabalho 01 há apenas camadas de aplicação e física:
 - Cada quadrado representa um protocolo; e
 - Cada protocolo será implementado por meio de uma sub-rotina



Trabalho Prático: Camada Física Transmissão



```

/*****
 * NAO ESQUECER DOS COMENTARIOS!
 *****/
void main (void) {
    AplicacaoTransmissora();
} //fim do metodo main

void AplicacaoTransmissora (void) {
    string mensagem;
    cout << "Digite uma mensagem:" << endl;
    cin >> mensagem;

    //chama a proxima camada
    CamadaDeAplicacaoTransmissora(mensagem); //em um exemplo mais
        //realistico, aqui seria dado um SEND do SOCKET
} //fim do metodo AplicacaoTransmissora

void CamadaDeAplicacaoTransmissora (string mensagem) {
    //int quadro [] = mensagem //trabalhar com bits!!!


    //chama a proxima camada
    CamadaFisicaTransmissora(quadro);
} //fim do metodo CamadaDeAplicacaoTransmissora

```

Trabalho Prático: Camada Física Transmissão



```
void CamadaFisicaTransmissora (int quadro[]) {  
    int tipoDeCodificacao = 0; //alterar de acordo o teste  
    int fluxoBrutoDeBits []; //ATENÇÃO: trabalhar com BITS!!!  
  
    switch (tipoDeCodificacao) {  
        case 0 : //codificacao binaria  
            fluxoBrutoDeBits = CamadaFisicaTransmissoraCodificacaoBinaria(quadro);  
            break;  
        case 1 : //codificacao manchester  
            fluxoBrutoDeBits = CamadaFisicaTransmissoraCodificacaoManchester(quadro);  
            break;  
        case 2 : //codificacao bipolar fluxoBrutoDeBits =  
            CamadaFisicaTransmissoraCodificacaoBipolar(quadro);  
            break;  
    } //fim do switch/case  
  
    MeioDeComunicacao(fluxoBrutoDeBits);  
} //fim do metodo CamadaFisicaTransmissora
```



Trabalho Prático: Camada Física Transmissão



```
int[] CamadaFisicaTransmissoraCodificacaoBinaria (int quadro []) {  
    //implementacao do algoritmo  
} //fim do metodo CamadaFisicaTransmissoraCodificacaoBinaria  
  
int[] CamadaFisicaTransmissoraCodificacaoManchester (int quadro []) {  
    //implementacao do algoritmo  
} //fim do metodo CamadaFisicaTransmissoraCodificacaoManchester  
  
int[] CamadaFisicaTransmissoraCodificacaoBipolar(int quadro []){  
    //implementacao do algoritmo  
} //fim do CamadaFisicaTransmissoraCodificacaoBipolar
```

Trabalho Prático: Camada Física Meio de Comunicação




```
/* Este metodo simula a transmissao da informacao no meio de
 * comunicacao, passando de um pontoA (transmissor) para um
 * ponto B (receptor)
 */
void MeioDeComunicacao (int fluxoBrutoDeBits []) {
    //OBS IMPORTANTE: trabalhar com BITS e nao com BYTES!!!
    int fluxoBrutoDeBitsPontoA[], fluxoBrutoDeBitsPontoB[];

    fluxoBrutoDeBitsPontoA = fluxoBrutoDeBits;

    while (fluxoBrutoDeBitsPontoB.lenght!= fluxoBrutoDeBitsPontoA) {
        fluxoBrutoBitsPontoB += fluxoBrutoBitsPontoA; //BITS! Sendo transferidos
    }//fim do while


    //chama proxima camada
    CamadaFisicaReceptora(fluxoBrutoDeBitsPontoB);
} //fim do metodo MeioDeTransmissao
```



Trabalho Prático: Camada Física Recepção



```
void CamadaFisicaReceptora (int quadro[]) {  
    int tipoDeDecodificacao = 0; //alterar de acordo o teste  
    int fluxoBrutoDeBits []; //ATENÇÃO: trabalhar com BITS!!!  
  
    switch (tipoDeDecodificacao) {  
        case 0 : //codificacao binaria  
            fluxoBrutoDeBits = CamadaFisicaReceptoraDecodificacaoBinaria(quadro);  
            break;  
        case 1 : //codificacao manchester  
            fluxoBrutoDeBits = CamadaFisicaReceptoraDecodificacaoManchester(quadro);  
            break;  
        case 2 : //codificacao bipolar fluxoBrutoDeBits =  
            CamadaFisicaReceptoraDecodificacaoBipolar(quadro);  
            break;  
    }//fim do switch/case  
  
    //chama proxima camada  
    CamadaDeAplicacaoReceptora(fluxoBrutoDeBits);  
}//fim do metodo CamadaFisicaTransmissora
```



Trabalho Prático: Camada Física Recepção



```
int[] CamadaFisicaReceptoraCodificacaoBinaria (int quadro []) {  
    //implementacao do algoritmo para DECODIFICAR  
}  
//fim do metodo CamadaFisicaReceptoraDecodificacaoBinaria  
  
int[] CamadaFisicaReceptoraCodificacaoManchester (int quadro []) {  
    //implementacao do algoritmo para DECODIFICAR  
}  
//fim do metodo CamadaFisicaReceptoraDecodificacaoManchester  
  
int[] CamadaFisicaReceptoraCodificacaoBipolar(int quadro []){  
    //implementacao do algoritmo para DECODIFICAR  
}  
//fim do CamadaFisicaReceptoraDecodificacaoBipolar
```

Trabalho Prático: Camada Física Recepção



```
void CamadaDeAplicacaoReceptora (int quadro []) {  
    //string mensagem = quadro []; //estava trabalhando com bits  
  
    //chama proxima camada  
    AplicacaoReceptora(mensagem);  
}  
//fim do metodo CamadaDeAplicacaoReceptora  
  
void AplicacaoReceptora (string mensagem) {  
    cout << "A mensagem recebida foi:" << mensagem << endl;  
}  
//fim do metodo AplicacaoReceptora
```