Nota de aula: Sistemas de Comunicações Móveis

Capítulo 3 – Codificadores de fonte e de canal

Resumo

No contexto dos avanços tecnológicos em comunicações digitais, o Capítulo 3 aborda a importância dos codificadores de fonte e de canal, componentes essenciais para a transmissão eficiente e segura de dados através de meios como fibras ópticas e canais sem fio. Os codificadores de fonte são cruciais na compressão de dados, minimizando redundâncias sem comprometer informações essenciais, adequando-se tanto para transmissões com perda aceitável (áudio e vídeo) quanto para aquelas onde a integridade dos dados é prioritária (textos e dados médicos).

Por outro lado, os codificadores de canal adicionam redundância controlada aos dados, permitindo a detecção e correção de erros introduzidos durante a transmissão. Eles utilizam códigos de correção de erros, como os códigos de Hamming e convolucionais, para garantir que os dados cheguem ao receptor com a máxima fidelidade possível, abordando os desafios impostos pelos ruídos e interferências típicos do ambiente de transmissão.

Este capítulo também detalha a funcionalidade e aplicação de diferentes técnicas de compressão de dados, a análise de componentes de sinais discretos e a importância de balancear a taxa de bits, qualidade do sinal reconstruído, complexidade de implementação e latência em comunicações digitais. As estratégias para minimizar a latência e otimizar a qualidade dos sinais são discutidas, realçando a necessidade de algoritmos eficientes que suportem a rápida evolução das demandas de comunicação.

Adicionalmente, o capítulo apresenta uma visão técnica sobre a teoria da codificação, que se fundamenta na teoria da informação estabelecida por Claude Shannon. A teoria da codificação desempenha um papel vital na maximização da eficiência da transmissão e armazenamento de dados, sendo central para o desenvolvimento e aprimoramento contínuo das tecnologias de comunicação digital.

Palavras-chave: codec, compressão de dados, Hamming

2.1 Introdução

Na era da comunicação digital, a transmissão eficiente e confiável de dados através de diversos meios, como fibras ópticas, canais sem fio e redes de cobre, é crucial. Uma parte central dessa eficiência é atribuída aos avanços na tecnologia de codecs, que inclui tanto codificadores de fonte quanto codificadores de canal. Esses componentes são vitais para otimizar o desempenho dos sistemas de transmissão e recepção, garantindo a integridade e a eficácia da comunicação digital.

Prof. Clayton J A Silva

"Codec" é um termo derivado da contração de "codificador" e "decodificador". Em comunicações e computação, um codec é um dispositivo ou programa de software capaz de codificar ou decodificar um fluxo de dados ou sinal. No contexto de transmissores digitais, o codec desempenha funções críticas tanto na compressão quanto na descompressão de dados, permitindo uma transmissão de dados mais rápida e eficiente.

O codificador de fonte desempenha um papel fundamental na preparação dos dados para transmissão eficiente. Este componente do transmissor digital é responsável pela compressão dos dados, reduzindo sua redundância sem perder informações essenciais. Codificadores de fonte aplicam algoritmos de compressão que podem ser com perda, como aqueles usados para áudio e vídeo, onde uma certa degradação da qualidade é aceitável, ou sem perda, para aplicações onde a integridade dos dados é crítica, como em textos e dados médicos. A compressão de dados não apenas economiza largura de banda mas também ajuda a gerenciar melhor os recursos de armazenamento.

Após a compressão de dados pelo codificador de fonte, o codificador de canal entra em ação. Seu papel é crucial na adição de redundância controlada aos dados, o que é vital para a detecção e correção de erros durante a transmissão através de canais potencialmente ruidosos. Codificadores de canal utilizam códigos de correção de erros, como os códigos de Hamming ou códigos convolucionais, para garantir que qualquer erro introduzido durante a transmissão possa ser identificado e corrigido pelo decodificador no receptor. Essa capacidade de manter a integridade dos dados em face de erros é fundamental para garantir a confiabilidade e a eficácia das comunicações digitais.

A combinação de codificadores de fonte e codificadores de canal nos transmissores digitais é essencial para otimizar tanto a eficiência quanto a confiabilidade da comunicação digital. Enquanto o codificador de fonte assegura que os dados sejam compactados de forma eficiente, eliminando partes desnecessárias sem sacrificar a qualidade crítica, o codificador de canal prepara esses dados para enfrentar as adversidades inerentes ao meio de transmissão, como ruídos e interferências. Juntos, eles permitem a transmissão de uma quantidade maior de dados, mais rapidamente e com maior segurança, características indispensáveis em nossa crescente dependência de sistemas de comunicação digital em todos os aspectos da vida moderna.

Essa sinergia entre codificação de fonte e codificação de canal define a capacidade de um sistema de comunicação digital para transmitir informações de maneira eficaz, marcando sua importância indelével na engenharia de comunicações.

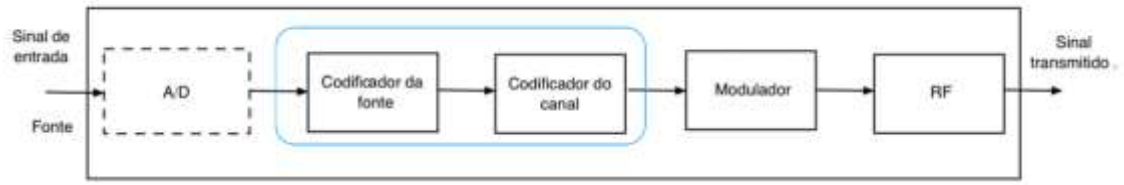

Figura 1 – Modelo de transmissor digital com destaque aos codecs (codificador da fonte e do canal)

2.2 Codificador da fonte

A codificação da fonte em transmissores de comunicações móveis digitais é uma prática essencial que visa otimizar o uso do espectro de frequência e melhorar a eficiência na transmissão de dados. O processo de codificação da fonte envolve a transformação de uma mensagem digital, que pode ser texto, áudio, vídeo ou qualquer outro tipo de dado, em uma sequência de símbolos discretos. Essa transformação é projetada para minimizar a redundância dos dados e, consequentemente, reduzir a quantidade de dados necessários para transmitir a informação original sem perda de qualidade ou significado.

Uma técnica comum utilizada nesse contexto é a compressão de dados, que pode ser sem perda ou com perda. A compressão sem perda permite que os dados originais sejam perfeitamente reconstruídos a partir dos dados comprimidos, o que é crucial para textos e dados onde a precisão é fundamental. Algoritmos como o Huffman e o Lempel-Ziv (LZ) são exemplos de métodos de compressão sem perda amplamente utilizados. Por outro lado, a compressão com perda, que é frequentemente aplicada a dados de áudio e vídeo, reduz o tamanho do arquivo eliminando informações menos importantes, o que pode levar a uma degradação perceptível da qualidade em níveis mais elevados de compressão.

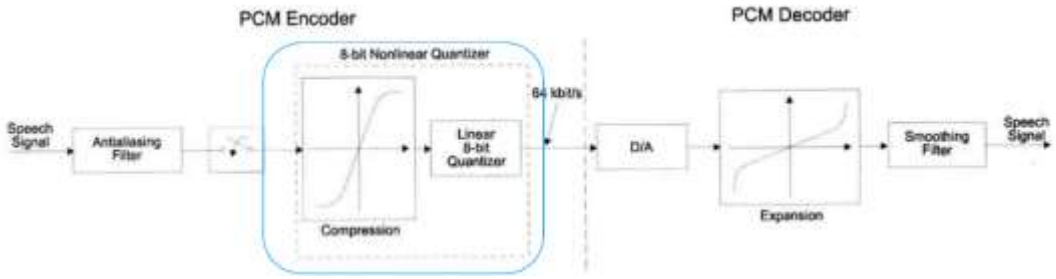

Figura 2 – Codificador PCM

Na codificação de fonte, especialmente em transmissões digitais, é fundamental identificar e manipular diferentes componentes do sinal para maximizar a eficiência da transmissão. A estrutura do sinal discreto pode ser analisada em termos de quatro componentes principais, cada um dos quais é tratado de maneira específica durante o processo de codificação:

1. Componente Relevante: Este é o núcleo da informação que precisa ser transmitida de forma intacta, pois contém os dados essenciais que definem a mensagem. Em áudio e vídeo, isso incluiria as frequências sonoras ou os elementos visuais principais que são perceptíveis e importantes para o receptor.

2. Componente Irrelevante: Durante o processo de quantização, este componente é reduzido ou eliminado. A quantização é um processo pelo qual os valores contínuos de um sinal são mapeados para valores discretos, frequentemente reduzindo a precisão mas conservando as características mais significativas do sinal. Isso ajuda a reduzir a quantidade de dados, pois os detalhes que são menos perceptíveis ao usuário final são removidos.

3. Componente Não Redundante: Este componente é crucial e deve ser transmitido sem redução, pois representa a informação única no sinal que não pode ser prevista ou inferida a partir de outras partes da mensagem.

4. Componente Redundante: Técnicas de predição e transformação são usadas para reduzir este componente. A predição utiliza modelos estatísticos para estimar partes do sinal baseadas em outras partes conhecidas, permitindo que apenas as diferenças sejam codificadas e transmitidas, o que economiza largura de banda. Transformações de sinal, como a Transformada Discreta de Cosseno (DCT) utilizada na compressão JPEG, reorganizam o sinal de maneira que os componentes de baixa energia (menos importantes) possam ser mais facilmente quantizados ou até descartados.

A Figura 3 ilustra o princípio básico da compressão de sinais.

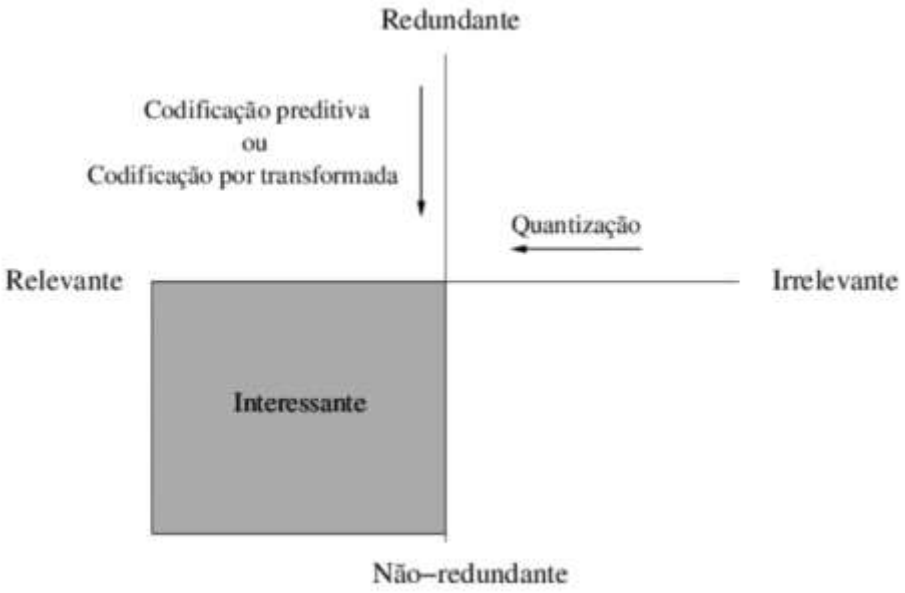

Figura 3 – Esquema de compressão de sinais

No contexto da codificação de fonte, o objetivo central da compressão é minimizar a taxa de bits usada na representação digital do sinal, ao mesmo tempo que se mantém

ou se equilibra três aspectos fundamentais: a qualidade do sinal reconstruído, a complexidade da implementação e o retardo da comunicação. Cada um desses aspectos tem implicações significativas para o desempenho e a aplicabilidade do sistema de comunicação.

1. Qualidade do Sinal Reconstruído: A qualidade do sinal após a compressão e descompressão é crucial, especialmente em aplicações onde a fidelidade do sinal é prioritária, como em transmissões de vídeo e áudio de alta definição. O desafio é manter a integridade do sinal ao mínimo custo de bits possível. Métodos de compressão sem perdas são ideais quando a integridade completa do sinal é necessária, enquanto métodos de compressão com perda podem ser utilizados onde alguma degradação do sinal é aceitável em troca de taxas de compressão significativamente maiores.

2. Complexidade da Implementação: A complexidade de implementar algoritmos de codificação e decodificação pode variar significativamente. Algoritmos complexos podem oferecer melhor compressão e qualidade, mas requerem mais poder de processamento, o que pode ser um obstáculo em dispositivos com capacidade limitada ou em aplicações em tempo real. Portanto, é essencial encontrar um equilíbrio entre a eficácia da compressão e a viabilidade da implementação em hardware ou software específico.

3. Retardo da Comunicação: Em muitas aplicações de comunicação, especialmente aquelas que são sensíveis ao tempo, como teleconferências ou jogos online, o retardo introduzido pela codificação e decodificação do sinal é um fator crítico. O retardo deve ser minimizado para garantir uma comunicação fluida e responsiva. Isso frequentemente requer um compromisso entre a qualidade do sinal e a rapidez com que ele pode ser processado e transmitido.

Análise da qualidade

Na análise da qualidade do sinal recebido, tanto medidas qualitativas quanto quantitativas são fundamentais para avaliar a eficácia de um sistema de comunicação. Essas medidas ajudam a determinar o grau de degradação do sinal devido à compressão e outras formas de processamento. Vamos explorar ambos os tipos de medidas: medidas qualitativas e medidas quantitativas. Essas medidas, tanto qualitativas quanto quantitativas, são complementares e, quando usadas juntas, oferecem uma visão abrangente da qualidade do sinal em sistemas de comunicação digital, ajudando a otimizar tanto a técnica quanto a experiência do usuário final.

As medidas qualitativas são tipicamente subjetivas e baseiam-se na percepção humana da qualidade do sinal. Elas são usadas principalmente em contextos onde a experiência do usuário é crucial, como em transmissões de vídeo e áudio. Um exemplo clássico é a realização de painéis de escuta ou visualização onde grupos de usuários avaliam a

qualidade do áudio ou vídeo em uma escala de qualidade, como o *Mean Opinion Score* (MOS). No MOS, os usuários classificam a qualidade do sinal em uma escala de 1 a 5, onde 1 representa qualidade muito ruim e 5 excelente. Este tipo de avaliação é especialmente útil porque leva em conta a percepção humana, que pode não ser completamente representada por medidas técnicas.

As medidas quantitativas fornecem uma avaliação objetiva da qualidade do sinal com base em critérios técnicos específicos. Um exemplo disso é a taxa de erro de bits (BER), que mede a proporção de bits que foram alterados (erros) após a transmissão e recepção em relação ao total de bits enviados. Outro exemplo é o Signal-to-Noise Ratio (SNR), que avalia a relação entre o sinal desejado e o ruído de fundo. Essas medidas são essenciais para avaliar a integridade técnica do sinal de uma maneira que é independente da percepção humana.

As medidas de erro são comumente usadas em processamento de sinais e estatísticas para avaliar o desempenho de algoritmos de previsão, filtragem, compressão de dados e outras aplicações relacionadas a comunicações e controle. Aqui estão as definições e as fórmulas correspondentes:

1. Erro Instantâneo $e(n)$ é a diferença entre o valor real e o valor estimado ou predito de um sinal em um determinado instante. É uma medida pontual de erro em um dado tempo. É calculado pela relação

$$e(n)=y(n)–y\hat{}(n)$$

, onde $y(n)$ é o valor real e $y\hat{}(n)$ é o valor predito ou estimado no instante *n*.

2. Erro Médio (ME) é a média aritmética dos erros instantâneos ao longo de um conjunto de N amostras, indicando a tendência central do erro. É calculado pela relação

$$ME = \frac{1}{N}\sum_{n} e(n) = \frac{1}{N}\sum_{n} y(n) - y\hat{}(n)$$

, onde $e(n)$ é o erro instantâneo em cada amostra *n*.

3. Erro Médio Quadrático (MSE) é uma medida que calcula a média dos quadrados dos erros. O MSE é muito usado porque penaliza mais fortemente os erros maiores, sendo útil quando grandes erros são particularmente indesejáveis. É calculado pela relação

$$MSE = \frac{1}{N}\sum_{n} e^2(n) = \frac{1}{N}\sum_{n} [y(n) - y^n]^2$$

, onde $e(n)$ é o erro instantâneo em cada amostra *n*.

4. Razão Sinal-Ruído de Erro (SENR) é uma medida que compara o nível do sinal desejado ao nível do erro (ruído) causado pela estimativa ou previsão. É uma variante

do tradicional *Signal-to-Noise Ratio* (SNR) focada especificamente nos erros. É calculado pela relação

$$SENR = 10log\frac{\sum_n y^2(n)}{\sum_n e^2(n)}$$

Essas medidas ajudam a quantificar a precisão de algoritmos em termos de quão bem eles reproduzem ou predizem dados em comparação com os valores reais.

Complexidade

O conceito de complexidade de algoritmos é central para entender a eficiência e a viabilidade de implementação de algoritmos em sistemas computacionais. A complexidade pode ser analisada sob duas perspectivas principais: complexidade de tempo e complexidade de espaço.

A complexidade de tempo de um algoritmo se refere ao tempo necessário para completar a execução como uma função do tamanho da entrada, geralmente expresso em termos de número de instruções ou operações. Uma métrica comum para medir a complexidade de tempo é o MIPS (Milhões de Instruções Por Segundo), que indica quantas milhões de instruções de máquina um computador pode processar por segundo. Isso é particularmente útil para entender a eficiência de um algoritmo em relação ao hardware em que é executado.

A complexidade de espaço de um algoritmo descreve a quantidade de memória necessária para o processamento do algoritmo. Isso inclui tanto a memória temporária para execução quanto o espaço permanente para armazenar o programa e os dados necessários. A complexidade de espaço é crucial em dispositivos com capacidade de armazenamento limitada ou onde o custo da memória é alto.

Além dessas métricas, ao analisar a complexidade de algoritmos, especialmente em codecs de áudio e vídeo, considera-se também:

- Tamanho Físico e Custo. O tamanho físico e o custo dos dispositivos necessários para executar o algoritmo são importantes, principalmente em dispositivos incorporados ou portáteis. Um algoritmo que requer hardware mais sofisticado pode não ser viável para todas as aplicações devido ao aumento dos custos e das dimensões físicas necessárias.

- Consumo de Potência. O consumo de energia é uma preocupação crítica em dispositivos móveis e portáteis, onde a energia é limitada. Algoritmos que consomem menos energia são preferíveis para tais dispositivos, mesmo que ofereçam menor desempenho em outros aspectos.

A complexidade de um algoritmo pode ser calculada ou estimada através de vários métodos, incluindo análise teórica, medição empírica e *benchmarking*.

Na análise teórica, estuda-se o algoritmo e conta-se o número máximo de operações (como comparações, atribuições, etc.) que ele realiza, geralmente resultando em uma notação de complexidade como O(n), O(log n), etc. Na medição empírica, executa-se o algoritmo com diferentes tamanhos de entrada e medindo o tempo de execução e o uso de memória. Essas medições podem então ser comparadas para entender como a complexidade cresce com o tamanho da entrada. Em *benchmarking*, compara-se o desempenho do algoritmo com outros algoritmos que executam a mesma tarefa. Isso é comumente feito usando suites de *benchmark* que padronizam as condições de teste para uma comparação justa.

Cada uma dessas abordagens tem suas vantagens e limitações, e frequentemente uma combinação delas é usada para obter uma compreensão completa da complexidade de um algoritmo. Ao escolher um algoritmo para uma aplicação específica, é crucial considerar tanto a complexidade teórica quanto os resultados empíricos, equilibrando eficiência, custo e outros fatores práticos.

Retardo ou atraso no processamento de dados pelo codec

O retardo ou atraso no processamento de dados por um codec é um aspecto crítico da análise de desempenho em sistemas de comunicação, especialmente em contextos onde a interatividade ou a sincronização temporal é fundamental. Esse atraso, geralmente chamado de "latência", pode ter um impacto significativo na experiência do usuário e na eficácia da comunicação, dependendo da aplicação específica.

O retardo do codec pode ser definido como o tempo total que leva para um sinal ser codificado no transmissor, transmitido e, em seguida, decodificado no receptor. Esse atraso é composto por várias partes: tempo de codificação, ou seja, tempo necessário para o codec processar e codificar os dados de entrada; tempo de transmissão, ou seja, tempo que leva para os dados codificados serem enviados ao receptor; e tempo de decodificação, ou seja, tempo necessário para converter os dados codificados de volta à forma original no receptor.

O impacto do atraso varia consideravelmente entre diferentes aplicações, dependendo da criticidade o impacto pode ser maior ou menor.

Na comunicação em tempo real, aplicações como videoconferências, jogos online e telefonia VoIP são extremamente sensíveis ao atraso. Atrasos superiores a 150-200 milissegundos podem ser perceptíveis e prejudicar a experiência do usuário, causando interrupções no fluxo de comunicação e dificuldades na interação.

Em aplicações de *streaming* de mídia, embora serviços de streaming de vídeo e áudio possam tolerar atrasos um pouco maiores devido ao uso de *buffers*, atrasos muito longos ainda podem afetar negativamente a sincronização entre vídeo e áudio, bem como a resposta do sistema a comandos do usuário.

Por último, existem aplicações que não são sensíveis ao tempo. Por exemplo, transferências de arquivos ou atualizações de dados em segundo plano geralmente toleram atrasos mais longos, pois a interação do usuário não é em tempo real.

Diversas estratégias podem ser empregadas para minimizar o atraso do codec, incluindo a otimização do algoritmo do codec, melhorando a eficiência do algoritmo de codificação e decodificação pode reduzir o tempo de processamento, o uso de codecs de baixa latência e o uso de hardware mais poderoso para acelerar o processo de codificação e decodificação, aumentando a capacidade de processamento.

Retardo na transmissão

O retardo na transmissão, especialmente em canais móveis, é uma consideração crucial na avaliação e no desenho de sistemas de comunicação móvel. Esse retardo, comumente referido como latência, pode significativamente afetar a qualidade e a eficácia da comunicação, dependendo da natureza e das exigências da aplicação.

Os seguintes fatores contribuem para o retardo em canais móveis:

1. Tempo de Propagação. Este é o tempo que leva para um sinal viajar do transmissor ao receptor. Em sistemas móveis, esse tempo pode variar significativamente devido à mudança na localização física dos dispositivos móveis e às características variáveis do ambiente de propagação.

2. Processamento no nó de base e na rede. Os sinais em sistemas móveis frequentemente passam por várias camadas de processamento nos nós de base e em outros componentes da rede, cada um adicionando seu próprio atraso. Isso inclui tarefas como codificação e decodificação de sinal, roteamento e, em alguns casos, *bufferização*. Os nós são os pontos de uma rede através dos quais os pacotes de dados são encaminhados ou processados.

3. Condições do Canal. A qualidade do canal móvel pode flutuar devido a fatores como obstáculos físicos, interferências e *multipath fading* (quando o sinal chega ao receptor por múltiplos caminhos, cada um com seu próprio atraso). Essas condições podem exigir que os sinais sejam retransmitidos, aumentando ainda mais o retardo.

4. Protocolos de Comunicação. Os protocolos de *handshaking* (estabelecimento de conexão) e controle de erro usados em comunicações móveis podem também introduzir atrasos adicionais. Por exemplo, em condições de sinal fraco, os dispositivos podem repetidamente tentar estabelecer uma conexão estável, incrementando o tempo total de transmissão.

O impacto do retardo na transmissão é particularmente perceptível em aplicações da mesma forma tratada anteriormente. Entretanto, convém destacar as aplicações de controle de veículos e sistemas de automação. Em aplicações de controle remoto, como drones ou sistemas de transporte inteligentes, atrasos na transmissão podem

resultar em respostas tardias a comandos, potencialmente levando a erros operacionais.

Para mitigar o impacto da latência em sistemas móveis, várias estratégias podem ser implementadas, como otimização da rede, uso de tecnologias de acesso rápido, entre outras.

Melhorar o roteamento de dados e a alocação de recursos na rede pode ajudar a reduzir a latência. Técnicas como o uso de redes de acesso por rádio distribuído (D-RAN) ou redes definidas por software (SDN) estão sendo exploradas para esse fim. Utilizar tecnologias de comunicação mais rápidas, como 5G, que são projetadas para oferecer latências significativamente mais baixas em comparação com as gerações anteriores. Implementar *caching* de conteúdo e capacidades de processamento na borda da rede (*edge computing*) para reduzir a necessidade de longas transmissões de dados e processamento centralizado.

2.3 Codificador do canal

2.3.1 Visão geral

A codificação de canal é o processo através do qual o transmissor adiciona redundância controlada à informação de modo a permitir a detecção e a correção de erros. Dependendo do número de bits adicionados, os códigos de canal podem permitir a correção de erros na transmissão ou somente a detecção dos erros ocorridos. Existem duas grandes famílias de códigos detetores e corretores de erros: os códigos de bloco e os convolucionais.

Os códigos de bloco funcionam segmentando a mensagem original em blocos de dados fixos e adicionando bits de paridade a cada bloco. Esses bits de paridade são calculados com base nas informações do bloco, de modo a criar um padrão que possa ser verificado pelo receptor para detectar e corrigir erros. Um dos exemplos mais conhecidos de códigos de bloco é o código de *Hamming*, que é projetado para corrigir um único erro por bloco e detectar até dois erros.

Os códigos de bloco são geralmente mais simples de implementar e são eficazes em situações onde os erros ocorrem de forma aleatória e isolada. Eles são amplamente utilizados em sistemas de armazenamento de dados e comunicações onde a latência de transmissão não é um fator crítico.

Diferentemente dos códigos de bloco, os códigos convolucionais não segmentam a mensagem em blocos. Em vez disso, eles aplicam operações matemáticas aos dados de entrada usando um processo que "convoluciona" a mensagem com uma função geradora, resultando em um fluxo contínuo de saída que inclui tanto a informação quanto a redundância. Esses códigos são frequentemente descritos em termos de sua

"memória", que se refere ao número de bits anteriores que afetam a geração dos bits de saída.

Os códigos convolucionais são especialmente úteis em ambientes onde os erros tendem a ocorrer em rajadas. Eles são comumente usados em comunicações via satélite e comunicação móvel, onde a qualidade do sinal pode variar dramaticamente, tornando os erros mais propensos a se agrupar.

Ambos os tipos de códigos são fundamentais em muitas aplicações tecnológicas modernas. Eles são usados desde as comunicações espaciais até as redes móveis 5G, onde a integridade e a confiabilidade da informação são cruciais. A escolha entre códigos de bloco e convolucionais depende das características específicas do canal de comunicação e dos requisitos do sistema.

Na comunicação de dados, a presença de erros durante a transmissão de um sinal pode ser categorizada em dois tipos principais: erros aleatórios e erros de rajada. Cada tipo tem características distintas e implica diferentes estratégias para detecção e correção.

Os erros aleatórios, também conhecidos como erros isolados, ocorrem esporadicamente e sem um padrão previsível ao longo da transmissão de dados. Eles são geralmente causados por ruídos e interferências aleatórias no canal de comunicação, como ruído térmico, ruído elétrico, e interferência eletromagnética. Esses erros são tipicamente distribuídos de maneira uniforme ao longo do tempo e são menos dependentes das características do sinal transmitido. Para combater os erros aleatórios, os códigos de bloco, como os códigos de Hamming, são particularmente eficazes porque podem corrigir erros isolados em blocos de dados, onde cada bloco é tratado de forma independente.

Diferentemente dos erros aleatórios, os erros de rajada ocorrem em sequências ou grupos. Um erro de rajada não implica que todos os bits em uma sequência estejam errados, mas que a presença de um erro aumenta a probabilidade de erros nos bits vizinhos. Este tipo de erro é comum em canais onde as condições podem degradar-se rapidamente, como em sistemas de comunicação via satélite, links de rádio sujeitos a desvanecimento, e em meios de armazenamento físico onde defeitos físicos podem afetar áreas contíguas. Os códigos convolucionais são frequentemente usados para corrigir erros de rajada, dado que o método de convolução (processamento contínuo dos dados de entrada através de uma função geradora) permite que o código abranja várias posições de bit, proporcionando proteção contra grupos de erros.

Em muitas aplicações práticas, técnicas híbridas são utilizadas para fornecer proteção robusta contra ambos os tipos de erros. Um exemplo comum é o uso de um código convolucional seguido por um código de bloco em um arranjo conhecido como codificação concatenada. Isso permite que o sistema aproveite as forças de ambos os

tipos de códigos - os convolucionais para lidar com erros de rajada e os de bloco para corrigir erros isolados que podem permanecer.

2.3.2 Teoria da codificação

A teoria da codificação é um ramo da matemática e da ciência da computação que estuda as propriedades dos códigos e sua aptidão para uma aplicação específica. Os códigos são usados para compressão de dados, criptografia, correção de erros, transmissão de dados e armazenamento de dados de forma eficiente e segura. As pesquisas em teoria da codificação têm sido fundamentais para o desenvolvimento das comunicações digitais e do armazenamento de dados.

A teoria da codificação se baseia fortemente em conceitos da teoria da informação, uma disciplina estabelecida por Claude Shannon na década de 1940. A pesquisa seminal de Shannon, "A Mathematical Theory of Communication" (1948), lançou as bases para a teoria da codificação ao introduzir conceitos fundamentais como: entropia, informação mútua e capacidade do canal.

2.3.3 Entropia

No contexto da teoria da informação, entropia é uma medida da incerteza ou da imprevisibilidade de uma fonte de informação. Shannon definiu a entropia como uma medida da quantidade média de informação produzida por uma fonte de informação. Para uma variável aleatória *X* com vários possíveis resultados xi, cada qual com probabilidade $p(xi)$, a entropia *H(X)* é dada por:

$$H(X)=-\sum p(xi)\log_2 p(xi)$$

A entropia é geralmente medida em bits (se o logaritmo base 2 é usado) e fornece um limite inferior para o número médio de bits necessários para codificar os símbolos produzidos pela fonte sem perda de informação.

Onde $p(xi)$ representa a probabilidade de ocorrência de cada possível valor x_i da variável aleatória X.

Apesar da presença do sinal negativo na fórmula, o valor da entropia $H(X)$ sempre será não-negativo. Isso se deve ao fato de que as probabilidades $p(xi)$ são sempre valores não-negativos entre 0 e 1, e o logaritmo de uma probabilidade $p(xi)$ será um número negativo (pois $0<p(xi)\leq 1$). Portanto, o produto $p(xi)\log 2p(xi)$ será não-positivo (ou seja, negativo ou zero), e ao somar todos esses termos e multiplicar por -1, o resultado final será um valor não-negativo.

A entropia zero ocorre quando a variável aleatória tem um valor com probabilidade 1 (certeza total), e os valores mais altos de entropia ocorrem quando há mais incerteza sobre o resultado da variável aleatória, ou seja, quando a distribuição de probabilidade

é mais uniforme entre os possíveis valores de X. Assim, a entropia mede a quantidade de incerteza ou "surpresa" associada ao resultado de uma variável aleatória.

Taxa de Informação

A taxa de informação de uma fonte é a média ponderada da informação associada a cada símbolo, também medida em bits. Esta taxa é essencialmente a entropia da fonte, indicando a eficiência máxima com que os dados podem ser codificados. A taxa de informação também pode ser vista como a capacidade de um canal de comunicação, que é a taxa máxima na qual a informação pode ser transmitida com uma probabilidade arbitrariamente baixa de erro sob condições ideais.

Além de Claude Shannon, muitos outros contribuíram significativamente para a teoria da codificação e a teoria da informação. Richard Hamming, conhecido pelo desenvolvimento dos códigos de *Hamming* na década de 1950, que são códigos de correção de erros que permitem que o sistema detecte e corrija erros automaticamente. Robert G. Gallager desenvolveu códigos de baixa densidade de verificação de paridade (LDPC) que são amplamente utilizados em modernos sistemas de comunicação para transmitir dados de maneira confiável sobre canais ruidosos. Andrew Viterbi inventou o algoritmo de Viterbi para decodificação de códigos convolucionais, que é amplamente utilizado em sistemas de comunicação celular e TV digital.

Esses conceitos e contribuições formam a espinha dorsal da moderna comunicação digital e do armazenamento de dados, permitindo o desenvolvimento de tecnologias que vão desde a Internet e telefonia móvel até satélites e transmissões de TV. A teoria da codificação continua sendo um campo ativo de pesquisa, com novas teorias e aplicações sendo exploradas para enfrentar desafios emergentes nas comunicações e no processamento de dados.

Os códigos corretores de erro, fundamentais na teoria da codificação, podem de fato reduzir a probabilidade de erro na decodificação a valores extremamente pequenos. A capacidade desses códigos de minimizar erros segue uma relação fundamental entre o comprimento do código (denotado por *n*), a taxa de codificação, e a capacidade do canal de comunicação, conforme estabelecido pela Teoria da Informação de Shannon.

Quando *n*, o comprimento do código, é aumentado, a probabilidade de erro na decodificação de fato decresce exponencialmente, desde que o código seja adequadamente projetado para o canal em questão. Esse decréscimo exponencial ocorre porque o aumento no comprimento do código permite uma maior redundância e, consequentemente, uma maior capacidade de detectar e corrigir erros.

No entanto, o aumento de *n* também implica um aumento na complexidade do sistema. Isso inclui não apenas a complexidade computacional associada à codificação e decodificação dos dados, mas também considerações sobre o uso eficiente da largura

de banda e do armazenamento. Métodos de codificação e decodificação mais complexos podem exigir mais poder de processamento e mais memória, o que pode ser um desafio em sistemas com recursos limitados.

Com base nesses princípios, os objetivos da teoria da codificação podem ser resumidos como:

1. Encontrar Códigos Longos e Eficientes. Desenvolver códigos que, mesmo longos, sejam capazes de atingir taxas de erro muito baixas com uma eficiência de uso da banda próxima à capacidade do canal. Isto implica códigos que podem operar eficazmente perto dos limites estabelecidos pela teoria da capacidade de canal de Shannon.

2. Encontrar Métodos Práticos de Codificação/Decodificação Eficientes. Desenvolver algoritmos de codificação e decodificação que não só minimizem a probabilidade de erro, mas que também sejam computacionalmente viáveis. Isso inclui a criação de algoritmos que possam ser implementados com eficiência em hardware ou software e que operem dentro dos limites de tempo e memória aceitáveis para aplicações práticas.

2.3.4 Códigos de bloco *(n , k, d)*

Os códigos de bloco, particularmente aqueles designados pela notação *(n, k, d)*, são uma forma fundamental de códigos corretores de erros na teoria da codificação. Eles são amplamente utilizados devido à sua estrutura simples e eficácia na detecção e correção de erros.

Conceitualmente, o código de bloco *(n, k, d)* é definido por:

- (*n*), que representa o comprimento total do bloco de código, ou seja, o número total de bits em cada palavra codificada;

- (*k*), que indica o número de bits de dados ou bits de informação em cada bloco. Estes são os bits que realmente carregam a informação que está sendo transmitida;

- (*d*), que denota a distância de *Hamming* entre quaisquer duas palavras do código distintas dentro do conjunto de palavras de código. A distância de Hamming é o número de posições nas quais dois blocos de código de mesma dimensão diferem.

Para um código de bloco *(n, k, d)*, o número total de palavras de código possíveis é 2^k. Isso reflete a variedade de combinações diferentes que os *k* bits de informação podem formar. Os (*n* - *k*) bits restantes são os bits de paridade ou redundância, adicionados ao código para permitir a detecção e correção de erros. Esses bits de paridade são calculados com base nos bits de informação, garantindo que qualquer erro introduzido durante a transmissão possa ser detectado e, dependendo do código, corrigido.

A eficiência de um código de bloco é uma medida de quão bem o espaço dos códigos é utilizado e é dada pela razão (*k/n*), que reflete a proporção de bits de informação em relação ao total de bits transmitidos. Um maior valor de eficiência indica uma maior proporção do bloco de código está sendo usada para transportar informações úteis, ao invés de redundância.

A distância de *Hamming* (d) é um indicador crítico da capacidade de um código de bloco de corrigir erros. Um código com uma distância de *Hamming* maior pode identificar e corrigir mais erros.

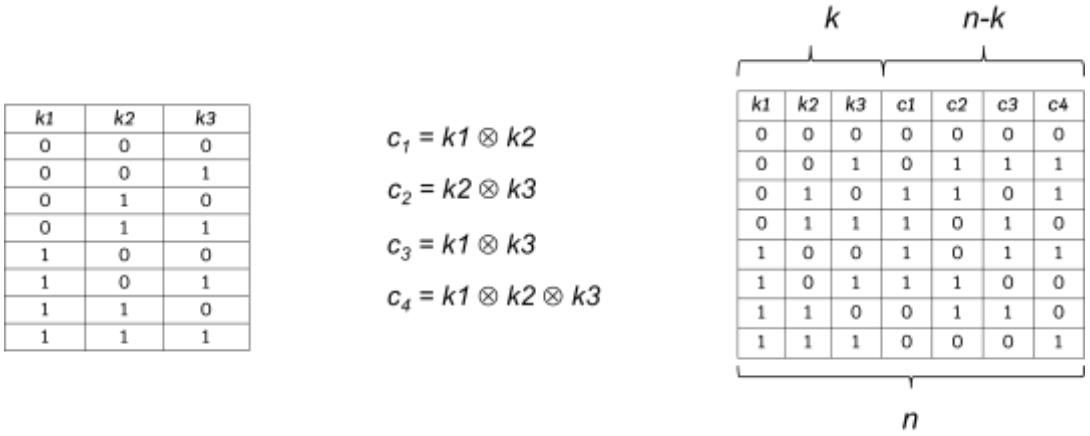

Figura 4 – Código de bloco (7,3,4)

Em um código com distância mínima d, o menor número de mudanças necessárias para converter uma palavra do código em outra é pelo menos d, logo é possível detectar o seguinte número de erros

$$d-1$$

Em um código com distância mínima d, supondo palavras com mesma probabilidade, é possível identificar e corrigir até t erros e decidir com acerto na detecção, se

$$2.t+1 \leq d$$

2.3.4.1 Códigos de repetição

Os códigos de repetição formam uma das mais simples e intuitivas classes de códigos de correção de erros utilizados na teoria da codificação. Como o próprio nome sugere, esses códigos operam repetindo cada bit de dados várias vezes para formar uma palavra de código. Apesar de sua simplicidade, os códigos de repetição ilustram os princípios básicos da codificação de erro: redundância, detecção e correção de erros.

Em um código de repetição, cada bit da mensagem original é repetido *n* vezes, onde *n* é geralmente um número ímpar para facilitar a decisão de maioria na decodificação. Por exemplo, se o bit a ser transmitido é '1' e *n = 3*, o bit transmitido será '111'. Similarmente,

um '0' seria transmitido como '000'. Este processo amplia a robustez do sinal contra a interferência e ruído, especialmente em canais com alta probabilidade de erro de bit.

Vantagens dos Códigos de Repetição:

1. Simplicidade. A maior vantagem dos códigos de repetição é sua simplicidade de implementação. Não requerem algoritmos complexos para codificação ou decodificação, o que os torna ideais para hardware com recursos limitados ou para aplicações onde a complexidade computacional deve ser minimizada.

2. Robustez em Ambientes Ruidosos. Em ambientes onde o ruído é significativo, repetir os bits ajuda a garantir que a informação seja recebida corretamente, pois a decodificação pode ser feita simplesmente por uma votação de maioria (*decoding by majority voting*). Por exemplo, se o código transmitido é '111' e o receptor obtém '101', pode inferir que o bit transmitido era provavelmente '1'.

Desvantagens dos Códigos de Repetição:

1. Baixa Eficiência de Taxa de Transmissão. Códigos de repetição têm uma eficiência de taxa muito baixa, definida como k/n, onde k é o número de bits de informação e n é o número total de bits transmitidos. Isso significa que uma grande parte da largura de banda é consumida pela redundância, o que os torna impraticáveis para a maioria das aplicações de comunicação modernas onde a largura de banda é um recurso valioso.

2. Capacidade de Correção Limitada. Embora os códigos de repetição possam ser eficazes para corrigir um pequeno número de erros, sua capacidade de correção não é adequada para situações em que erros de múltiplos bits são comuns, a menos que o número de repetições seja aumentado significativamente, o que reduz ainda mais a eficiência de taxa.

Devido às suas limitações, os códigos de repetição são frequentemente usados em circunstâncias muito específicas, como em comunicações simples de baixa velocidade ou em sistemas onde a integridade dos dados é crítica, mas a quantidade de dados é pequena. Eles também são utilizados como um componente em esquemas de codificação mais complexos, combinados com outros tipos de códigos para melhorar a confiabilidade geral do sistema de comunicação.

Resumindo, em códigos de repetição, os parâmetros são:

$$k=1,\ d \geq 1 \text{ e } n-1 \text{ bits redundantes.}$$

Como k=1, o código tem apenas duas palavras, uma delas é uma sequência de n 0s; a outra uma sequência de n 1s.

Os dígitos de paridade são todos uma repetição do dígito c_i de informação $n-1$ vezes.

A distância de Hamming do código é igual a n e a eficiência é

1/n.

Embora os códigos de repetição não sejam adequados para todos os tipos de aplicações devido à sua baixa eficiência de taxa e capacidade de correção limitada, eles oferecem uma introdução útil aos conceitos de codificação de erro e são um exemplo ilustrativo de como a redundância pode ser usada para combater erros em transmissões de dados.

2.3.4.2 Códigos com 1 bit de paridade

Os códigos com 1 bit de paridade são uma forma básica de códigos de detecção de erros usados em sistemas de comunicação digital para verificar a integridade dos dados transmitidos. Esses códigos funcionam adicionando um único bit extra, chamado bit de paridade, a um grupo de bits de dados, com o objetivo de fazer a soma total de bits '1' em uma palavra de código ser par ou ímpar, dependendo do esquema de paridade escolhido: paridade par ou paridade ímpar.

No esquema de paridade par, o bit de paridade é definido de modo que o número total de bits '1' na palavra de código completa (incluindo o bit de paridade) seja par. Por exemplo, se uma palavra de dados é '1101' (que tem um número ímpar de bits '1'), o bit de paridade seria definido como '1' para fazer o total de bits '1' par. No esquema de paridade ímpar, o bit de paridade é escolhido de forma que o número total de bits '1' seja ímpar. Continuando com o exemplo anterior, se uma palavra de dados é '1101', o bit de paridade seria '0', pois a palavra já contém um número ímpar de bits '1'.

Vantagens dos Códigos de Paridade:

1. Simplicidade. Um dos principais benefícios dos códigos de paridade é sua simplicidade. Eles são fáceis de implementar em hardware ou software e requerem poucos recursos computacionais.

2. Baixo Custo. Devido à sua simplicidade, eles são uma solução de baixo custo para a detecção de erros em aplicações onde os requisitos de integridade de dados não são extremamente rigorosos.

Desvantagens dos Códigos de Paridade:

1. Capacidade de Detecção Limitada. Os códigos de paridade podem detectar um número ímpar de erros, como 1, 3, 5, etc., erros em uma palavra de código. No entanto, eles não conseguem detectar um número par de erros, como 2, 4, 6, etc., o que limita sua utilidade em ambientes mais ruidosos ou em aplicações críticas.

2. Sem Capacidade de Correção. Esses códigos não fornecem nenhuma capacidade de correção de erros. Eles podem apenas indicar que um erro ocorreu, mas não podem determinar qual bit está errado ou corrigir o erro sem retransmissão.

Os códigos de paridade são frequentemente usados em situações em que a detecção de erros simples é suficiente e onde a eficiência dos dados é uma preocupação menor. Eles

são comuns em interfaces de comunicação serial, como nas transmissões UART, e em alguns tipos de memória de computador onde a detecção de erros de baixo nível é necessária.

Em resumo, em códigos com 1 bit de paridade, os parâmetros são:

$$k \geq 1,\ d=2 \text{ e } n=k+1$$

O código tem apenas um bit c redundante, que é definido na transmissão para tornar o número de bits 1 par (ou impar). A regra de decodificação é contar o número de bits recebidos. Se a paridade não estiver correta significa que houve um erro na transmissão.

A distância de *Hamming* do código é igual a 2 e a eficiência é

$$k/(k+1).$$

O código permite a detecção de 1 erro, mas não corrige. Pode ser eficaz quando se dispõe de canal de retorno para retransmissão da palavra.

Embora limitados em sua capacidade de detecção e correção de erros, os códigos de 1 bit de paridade oferecem uma solução de detecção de erros extremamente eficiente em termos de custo e complexidade para muitas aplicações eletrônicas e de comunicação digital. Eles são um exemplo básico de como a redundância pode ser utilizada para aumentar a confiabilidade dos sistemas de comunicação, servindo como uma introdução prática à conceitos mais avançados de codificação de erro.

2.3.4.3 Códigos de Hamming

Os códigos de *Hamming*, nomeados em homenagem ao seu inventor, Richard Hamming, são uma família de códigos de correção de erros lineares que se destacam pela sua capacidade de corrigir erros de um único bit e detectar erros de até dois bits. Introduzidos pela primeira vez nos anos 1950, os códigos de Hamming são amplamente utilizados em diversas aplicações digitais, incluindo memória de computador e comunicação de dados, devido à sua eficácia e eficiência na correção de erros comuns.

Os códigos de *Hamming* são códigos de bloco que usam uma combinação de bits de dados e bits de paridade. A ideia é posicionar os bits de paridade em pontos estratégicos dentro da palavra de código para que eles cubram diferentes combinações de bits de dados. Comumente, os bits de paridade são posicionados nas potências de dois (1, 2, 4, 8, etc.).

Cada bit de paridade é responsável por verificar a paridade (normalmente paridade par) de um grupo específico de bits. Por exemplo, o bit de paridade na primeira posição verifica todos os bits cujas posições têm o bit menos significativo em binário definido como 1 (posições 1, 3, 5, 7, etc.). O bit de paridade na segunda posição verifica todos os bits cuja segunda posição em binário é 1 (posições 2, 3, 6, 7, etc.), e assim por diante.

Essa estratégia permite que, quando ocorre um erro, a combinação de bits de paridade errados possa apontar diretamente para a posição do erro no bloco de código.

Características dos Códigos de Hamming:

- Capacidade de Correção. Os códigos de *Hamming* são capazes de corrigir erros de um único bit em cada palavra de código. Se um erro é detectado, os bits de paridade errados indicam a posição exata do erro, que pode ser corrigido invertendo o bit.

- Capacidade de Detecção. Além de corrigir um único erro, os códigos de *Hamming* podem detectar a presença de dois erros em uma palavra de código. No entanto, não podem corrigir dois erros simultaneamente.

- Distância de *Hamming*: A distância de Hamming mínima em códigos de Hamming é 3, o que significa que é necessário alterar pelo menos três bits para transformar uma palavra de código válida em outra.

Embora os códigos de *Hamming* sejam eficientes em termos de correção de um único erro, eles requerem uma quantidade relativamente alta de bits de paridade por bit de dados em comparação com outros códigos mais modernos, como códigos LDPC ou Turbo Codes. A quantidade de bits de paridade necessária aumenta com o tamanho da palavra de código para manter a capacidade de correção de erros, o que pode reduzir a taxa de transmissão de dados úteis.

Os códigos de Hamming são utilizados em uma variedade de aplicações onde erros de transmissão são relativamente raros e geralmente ocorrem de forma isolada. Eles são comumente encontrados em sistemas de memória de computadores, onde corrigem erros aleatórios causados por interferências ou falhas de hardware, e em sistemas de comunicação para garantir a integridade dos dados transmitidos.

Algoritmo de *Hamming*

Nos códigos de Hamming, o número de bits de paridade pode ser dado pela quantidade de divisões inteiras sucessivas de n por 2 até quociente igual a 0. Por exemplo, se n=7 bits, 7 div 2 = 3, 3 div 2 = 1, 1 div 2 = 0, logo d=3, k=n-3=7-3=4 Ou seja, em códigos de *Hamming*, os parâmetros são:

$$n \leq 2^d - 1$$

1. Os bits c de redundância e os bits k de informação são organizados em uma sequência de n bits a partir da posição 1...

   $$b1\ b2\ b3\ b4\ ...\ bn$$

2. ..., de modo que os bits de redundância ocuparão as posições que são potência de 2, logo

c1 c2 k3 c4 ...

3. Cada bit de redundância (bit de paridade) ajusta a paridade par (ou impar) dos bits específicos de informação...
4. Cada bit de informação *kj*, ou seja, o bit de informação que ocupa a posição *j*, é verificado pelos bits de redundância cuja soma das suas posições seja j. Por exemplo, o bit k3 é verificado pelos bits c1 e c2. O bit k5 é verificado pelos bits c1 e c4. O bit k6 é verificado pelos bits c2 e c4.
5. Cada bit de paridade será definido pelo codificador como 0 ou 1 de modo que o número de bits verificados seja par (ou impar).
6. No receptor, a determinação do bit incorreto pode ser obtida pela soma das posições de todos os bits de paridade incorretos. Por exemplo, se na recepção os bits c1 e c2 estão incorretos, significa que o erro identificado foi no bit k3.

Os códigos de *Hamming* são um componente vital da teoria da codificação e oferecem um bom equilíbrio entre complexidade e capacidade de correção de erros. Eles servem como uma solução robusta e confiável para muitos sistemas que exigem correção automática de erros, apesar de existirem alternativas que podem oferecer melhor desempenho em termos de eficiência e capacidade de correção em ambientes mais desafiadores.

2.4 REFERÊNCIAS

- Alencar, Marcelo S.; TELEFONIA CELULAR DIGITAL, 3a ed; érica Saraiva; 2014.
- HAYKIN, Simon. "Sistemas de Comunicação". Editora Bookman, 5ª Edição, 2008.
- PROAKIS, John G.; SALEHI, Masoud. "Comunicações Digitais". Editora McGraw-Hill, 5ª Edição, 2008.
- RAPPAPORT, Theodore S. "Telecomunicações sem Fio: Princípios e Práticas". Editora Prentice Hall, 2ª Edição, 2002.
- Stüber, Gordon L.; Principles of Mobile Communication; KLUWER ACADEMIC PUBLISHERS; 2002.
- Wesolowski, Krzysztof; Mobile Communications Systems; JOHN WILEY & SONS; 2002.