3
Codificação para Canal

No projeto de um sistema robusto de comunicação digital resulta útil a inclusão de um método corretor de erro para melhorar a confiabilidade da informação recebida, a escolha deste método influi diretamente em parâmetros fundamentais do desempenho, tais como: complexidade, eficiência em potência de transmissão e largura de banda. A escolha do método corretor dependerá basicamente da aplicação já que os requisitos para um sistema de comunicação podem ser diferentes aos exigidos em outro e portanto a implementação deste deve ser analisada em um contexto próprio.

São comuns sistemas de transmissão digital sobre canais ruidosos que empregam um canal de retorno que é usado para informar quais pacotes precisam ser retransmitidos. Quando um canal de retorno esta disponível e o receptor é capaz de detectar quais pacotes são recebidos corretamente e quais são recebidos com algum erro, pode-se pedir ao transmissor que aqueles pacotes recebidos com erro lhe sejam enviados de novo. Este procedimento é um protocolo de transmissão conhecido como ARQ (*Automatic Repeat Request*) [16].

Uma outra estratégia para a transmissão confiável é o uso de códigos para controle de erro. Esta estratégia consiste em usar um método de codificação o qual geralmente consiste em criar informação que incorpora a mensagem e redundância. A uma mensagem de comprimento k bits, o processo de codificação incorpora redundância e produz uma nova informação de comprimento n. Um parâmetro conhecido como taxa de código, $R = k/n$, indica que a os k bits de informação da mensagem foram incorporados $n - k$ bits redundantes. A nova informação, agora de comprimento n, será enviada através do canal de comunicação e com um alto grau de confiabilidade espera-se recuperar a mensagem original na recepção, mesmo em presença de ruído.

O esquema de nosso interesse é do tipo Códigos FEC(*Forward Error Correction*), Onde apenas a informação recebida pelo receptor, sem retransmissões

PUC-Rio - Certificação Digital Nº 0721360/CA

será usada para recuperar a informação original.

Os códigos em bloco são tradicionalmente utilizados como códigos para controle de erros, códigos Hamming [16], códigos *RS* (Reed-Solomon) [9], códigos Tornado [15] entre outros. Estes são códigos de taxa fixa, pelo qual apresentam limitações, por exemplo, k e n são limitados a valores razoavelmente pequenos. Para o uso destes códigos deve-se estimar *a priori* a probabilidade de falha do canal para assim determinar a taxa R antes que o processo de codificação tenha início [6], portanto com estes códigos de taxa fixa se as condições do canal pioram, falhas na decodificação aumentarão e, se as condições melhoram, haverá uma desnecessária transmissão de pacotes.

Os códigos FEC aparecem também como uma alternativa para usar-se em transmissões onde não é viável usar um canal de retorno, por exemplo, numa transmissão via satélite, devido aos tempos de resposta longos. Um relativamente recente tipo de códigos FEC são os códigos fontanais, que serão usados neste trabalho. A seguir apresentamos tais códigos.

PUC-Rio - Certificação Digital Nº 0721360/CA

3.1 Códigos Fontanais

O nome fontanais surgiu da analogia com uma fonte de agua a qual produz um número "ilimitado"de gotas de água, onde as gotas neste caso, são associadas aos blocos (pacotes) de saída do codificador. Os códigos fontanais podem recuperar, com probabilidade $(1 - \delta)$, um conjunto de k símbolos de entrada, a partir de quaisquer $k + O(\sqrt{k}\ln^2(k/\delta))$ símbolos de saída (blocos de saída do codificador), necessitando para isso de uma media de $O(k\ln(k/\delta))$ operações [5]. Os códigos fontanais foram originalmente concebidos para uso em canais sujeitos a apagamento.

3.1.1 Fontana Linear Aleatória

Para uma melhor compreensão dos códigos fontanais é bom entender o conceito de Fontana linear e suas características. Como dito anteriormente, o codificador em um código fontanal se comporta metaforicamente como uma fonte (fontana digital) que emite continuamente símbolos codificados, como se fossem gotas de água emitidas por uma fonte. Num arquivo de saída de tamanho kl bits, cada gota (pacote) contém l bits codificados. Assim, o interessado em receber dita informação bota um recipiente perto da fonte e coleta as "gotas". Quando o recipiente contiver um número de

pacotes ligeiramente maior que k o "coletor"(receptor) estará em condições de recuperar o arquivo original, integralmente, sem erro [6].

Com o propósito de ilustrar o funcionamento da codificação fontanal , suponha que se precisa codificar a mensagem $\mathbf{s} = (s_1, s_2, ..., s_K)$, que consiste em K símbolos de informação, Em cada ciclo, o codificador gerará K bits aleatoriamente $\{G_{kj}\}$, e transmitira uma mensagem codificada t_j, $1 \leq j \leq n$, dada por:

$$t_j = \sum_{k=1}^{K} s_k G_{kj} \tag{3-1}$$

Cada conjunto de K bit gerados aleatoriamente definem uma coluna, na sempre crescente matriz geradora original, As operações para obter t_j são feitas em modulo dois. Ao longo do tempo uma matriz $\mathbf{G}$ (matriz geradora original) é gerado pelo codificador. Estamos supondo que no processo de transmissão certos pacotes serão apagados e alguns pacotes chegaram sem erro, com aqueles pacotes recebidos sem erro se pretende recuperar o arquivo original de comprimento K, este processo se mostra na Figura 3.1

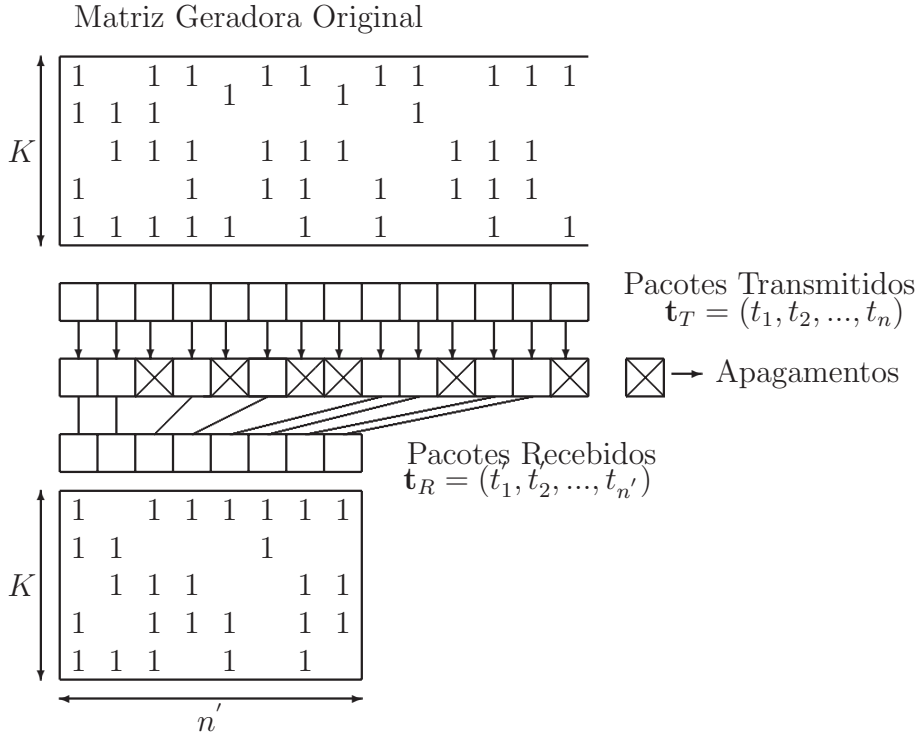

Figura 3.1: Matriz geradora de um código linear.

A partir da seqüencia recebida $\mathbf{t}_R$ e da matriz $K \times n'$ o objetivo é recuperar a mensagem original. Qual deve ser o tamanho de n' para recuperar a mensagem original de comprimento K?

PUC-Rio - Certificação Digital Nº 0721360/CA

- Para $K < n^{'}$, é impossível recuperar a informação original, todo se passa como se tivéssemos um sistema de equações lineares que não tem solução única.
- Para $K = n^{'}$, a matriz geradora é quadrada podendo-se encontrar sua inversa G^{-1}, em modulo dois, e assim obter:

$$s_k = \sum_{j=1}^{n^{'}} t_j G_{jk}^{-1} \tag{3-2}$$

  A probabilidade de encontrar (aleatoriamente) uma matriz binária G $K \times K$ inversível é o produto de k probabilidades (a probabilidade de que cada coluna seja linearmente independente das anteriores e não nula). O primeiro fator é $(1-2^{-K})$, a probabilidade de que a primeira columna de G não seja uma coluna com todos os elementos nulos. O segundo fator é $(1-2^{-(K-1)})$, probabilidade de que a segunda coluna também não seja uma coluna nula de G e não seja igual à coluna anterior. Iterando se obtém a probabilidade de encontrar uma matriz inversível é $(1-2^{-K})(1-2^{-(K-1)}) \times ... \times (1-\frac{1}{8})(1-\frac{1}{4})(1-\frac{1}{2})$. Para qualquer $K > 10$ esta probabilidade é aproximadamente 0.289.
- Para $n^{'} > K$, $n^{'} = K + E$, a probabilidade de conseguir uma solução para o sistema é pelo menos $(1-\delta)$ onde $\delta = 2^{-E}$, o qual indica a possibilidade de aproximar-se ao limite de Shannon quando o valor de K é incrementado, o problema é que a complexidade do sistema também aumenta.

PUC-Rio - Certificação Digital Nº 0721360/CA

3.2 Código LT

Introduzido por Michael Luby no ano 2002, o código LT (*Luby Transform*) da familia dos códigos fontanais, é uma classe de código desenvolvido procurando alcançar a capacidade de um canal BEC (*Binary Erasure Channel*). Os códigos LT têm a característica de ser um código com taxa versátil (*rateless*), que pode interpretar-se melhor como um código sem taxa fixa, este código possui o desempenho de um código Fontana Linear mas com uma complexidade mais baixa.

Como se mostrou anteriormente, as fontanas digitais são sistemas que produzem a partir de um bloco (ou arquivo)

$$\boldsymbol{s} = \left(\ s_1, \ s_2, \ \ldots \ , s_K \ \right)$$

com K símbolos de entrada gerados por uma fonte de informação, um bloco

$$\boldsymbol{t} = \Big(\, t_1, \, \ldots \, , t_n \, \Big),$$

com n símbolos codificados.

Devido que as operações nas que se baseiam os códigos LT são do tipo XOR a construção de codificadores e decodificadores resulta simples e eles podem operar em altas velocidades [11].

Cada símbolo codificado t_j é estatisticamente independente dos demais símbolos codificados e o conjunto de k símbolos de entrada originais podem ser recuperados, com probabilidade maior que $(1 - \delta)$, a partir de quaisquer $k + O(\sqrt{k}\ln^2(k/\delta))$ símbolos codificados, com uma média de $O(k\ln(k/\delta))$ operações por símbolo [5]. O número de símbolos codificados que podem ser gerados a partir dos dados de entrada é potencialmente ilimitado, pode-se ter $n \to \infty$ (e por esta razão os códigos fontanais são ditos "rateless"ou de taxa versátil).

Em um canal com apagamento, os códigos LT são "quase ótimos"no sentido de que o decodificador pode recuperar a mensagem original formada por k símbolos de entrada a partir de quaisquer $n = (1 + Overhead)k$ símbolos codificados, onde $0 < Overhead < 1$. Considera-se que um código é ótimo, se consegue recuperar a mensagem original (formada por K símbolos de entrada), a partir de quaisquer subconjunto de K símbolos de saída dentre os n,$n > K$ gerados pelo codificador [7]. Por serem quase ótimos e por sua eficiência aumentar com o crescimento do comprimento do bloco de entrada do codificador, os códigos LT são denominados universais [5]

PUC-Rio - Certificação Digital Nº 0721360/CA

3.2.1 Codificação LT

Para descrever o processo de codificação considere inicialmente uma sequência $\{D_1, \ldots, D_n\}$ de v.a.'s discretas i.i.d.'s $\forall_j$ em que $D_j \in \mathbb{Z}_k = \{1, \ldots, \text{k}\}$,$j = 1, \ldots, n$. Considere também que a função μ com

$$\mu(d) = P(D_j = d), \quad d \in \mathbb{Z}_k.$$

é a Distribuição de Probabilidades (DP) associada à v.a. D_j.

Para uma dada seqüencia de informação $\boldsymbol{s} = (\, s_1, \ldots, s_k \,)$, considerando que $\{d_1, \ldots, d_n\}$ é uma realização da sequência aleatória $\{D_1, \ldots, D_n\}$, tem-se então que cada símbolo codificado t_j é produzido de forma aleatória de acordo

com a seguinte equação

$$t_j = \bigoplus_{\ell=1}^{d_j} s_{i_{\ell j}} \tag{3-3}$$

onde $\bigoplus$ corresponde à adição módulo-2 dos termos do somatório.

É importante ressaltar que os valores dos índices $\{i_{1j}, i_{2j}, \ldots, i_{d_j j}\}$ das d_j parcelas $\{s_{i_1 j}, s_{i_2 j}, \ldots, s_{i_{d_j j}}\}$ em (3-3) são realizações da sequência de v.a.'s $I_1, \ldots, I_{d_j}$ onde $I_1 \in \mathbb{Z}_k$ é uma v.a. discreta com DP uniforme, ou seja, para $i_1 \in \mathbb{Z}_k$, tem-se

$$P(I_1 = i_1) = \frac{1}{k}.$$

A v.a. I_2 é também uma v.a. discreta com DP-condicional uniforme, definida da seguinte maneira para $(i_1, i_2) \in \mathbb{Z}_k^2$,

$$P(I_2 = i_2 \mid I_1 = i_1) = \begin{cases} \frac{1}{k-1} & i_2 \neq i_1 \\ 0 & i_2 = i_1 \end{cases}$$

A v.a. I_3 tem DP-condicional uniforme, definida para $(i_1, i_2, i_3) \in \mathbb{Z}_k^3$,

$$P\left(I_3 = i_3 \mid (I_1, I_2) = (i_1, i_2)\right) = \begin{cases} \frac{1}{k-2} & i_3 \notin \{i_1, i_2\} \\ 0 & i_3 \in \{i_1, i_2\}, \end{cases}$$

e, assim por diante, até que, para $(i_1, i_2, \ldots, i_{d_j}) \in \mathbb{Z}_k^{d_j}$, tem-se

$$P(I_{d_j} = i_{d_j} \mid (I_1, \ldots, I_{d_j - 1}) = (i_1, \ldots, i_{d_j - 1})) = \begin{cases} \frac{1}{k-(d_j-1)} & d_j \notin \{i_1, \ldots, i_{d_j-1}\} \\ 0 & d_j \in \{i_1, \ldots, i_{d_j-1}\}. \end{cases}.$$

Vale observar que $\boldsymbol{t} = (\, t_1, \ldots, t_j, \ldots, t_n\,) \in \{0,1\}^*$, onde $\{0,1\}^*$ é a notação usada para representar o conjunto de todos os blocos binários.

3.2.2
Codificador LT como um código em grafo

O codificador LT pode também ser associado com um grafo $\mathcal{G}$, bi-particionado, isto é, um grafo em que o conjunto de nós é particionado em dois conjuntos — um conjunto $\mathcal{E} = \{s_1, \ldots, s_i, \ldots, s_k\}$ de *nós de mensagem* ou *nós de entrada* e um conjunto $\mathcal{S} = \{t_1, \ldots, t_j, \ldots, t_n\}$ de *nós de verificação* ou *nós de saída*. A cada nó de entrada associa-se um símbolo de entrada s_i (que será identificado, indistintamente, como nó s_i), e a cada nó de saída corresponderá um símbolo codificado t_j (que será identificado como nó t_j).

PUC-Rio - Certificação Digital Nº 0721360/CA

Note que $\mathcal{G}$ é um grafo aleatório em que cada nó de saída t_j está conectado a d_j nós de entrada $\{s_{i_1}, \ldots, s_{i_{d_j}}\}$, onde $d_j \in \mathbb{Z}_k$ é escolhido aleatoriamente de acordo a uma DP $\mu(d)$. Tem-se assim que $\mathcal{V}_j = \{s_{i_1}, \ldots, s_{i_{d_j}}\}$ é o conjunto de nós de entrada vizinhos do nó t_j. O número de vizinhos d_j de um nó de saída t_j será referido como o *grau* do nó t_j e, a DP $\mu(d)$ será designada por *distribuição de graus* do código LT. Então, o codificador LT é descrito a seguir:

O processo de codificação de um código LT é o seguinte:

- Para cada símbolo codificado t_j, escolhe-se aleatoriamente e segundo uma distribuição de probabilidade $\mu(d)$, um número d_n definido como grau.
- Escolhe-se aleatoriamente, um conjunto $\{s_{i_1}, \ldots, s_{i_{d_n}}\}$, de d_n símbolos de entrada distintos (em posições $\{i_1, \ldots, i_{d_n}\}$ escolhidas de acordo com uma distribuição uniforme), como símbolos de entrada vizinhos do símbolo codificado t_j.
- O valor do símbolo codificado é

$$t_j = s_{i_1} \oplus \ldots \oplus s_{i_{d_j}},$$

onde $\oplus$ representa a adição bit a bit, modulo-2, desses d_n símbolos.

Na Tabela (3.1) pode-se ver para uma mensagem de entrada $s_1 s_2 s_3 = 101$, o grau e o respectivo valor do símbolo codificado. O grafo $\mathcal{G}$ associado a este processo é mostrado na Figura 3.2

Tabela 3.1: Processo de codificação.

símbolo	**grau** (d_j)	**vizinhos**	**valor**
t_1	1	s_1	$s_1 = 1$
t_2	3	$s_1 s_2 s_3$	$s_1 \oplus s_2 \oplus s_3 = 0$
t_3	2	$s_2 s_3$	$s_2 \oplus s_3 = 1$
t_4	2	$s_1 s_2$	$s_1 \oplus s_2 = 1$

3.2.3 Decodificação

A tarefa do decodificador é recuperar $\mathbf{s}$ de $\mathbf{t}_R = \mathbf{sG}$, onde $\mathbf{G}$ é a Matriz associada como o grafo $\mathcal{G}$, Aqui a partir dos símbolos recebidos se pretende recuperar a mensagem original. Para recuperar os k símbolos de entrada da mensagem original, supõe-se que o decodificador conhece o grafo $\mathcal{G}$, conforme a ilustração 3.2, isto é, o grau e o conjunto de vizinhos de cada símbolo codificado. Esta informação se encontra na matriz geradora. Neste processo

PUC-Rio - Certificação Digital Nº 0721360/CA

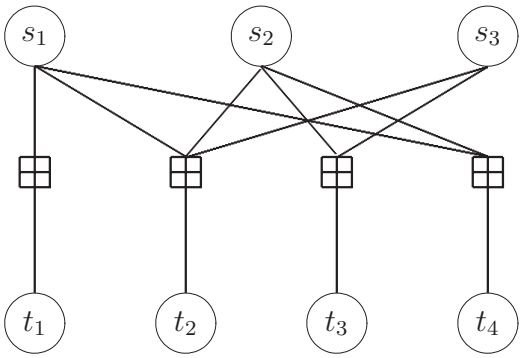

Figura 3.2: Grafo resultante da codificação.

de decodificação, utiliza-se a mesma notação: os símbolos codificados (t_j) formarão os nós de saída e os símbolos de entrada (s_k) formarão os nós de entrada. O algoritmo de decodificação de códigos LT é o seguinte:

1. Encontrar um nó de saída t_j que esteja conectado a apenas um símbolo de entrada s_k, caso não exista tal nó de saída, o processo de decodificação falha, sendo necessário receber mais símbolos codificados antes de fazer uma nova tentativa de decodificação.

   - Fazer $s_k = t_j$,
   - Somar em modulo-2, o valor de s_k a todos os nós de saída restantes t'_j que estejam conectados a s_k,
   - Remover todas as conexões que chegam ao símbolo de entrada s_k. Como resultado, o grau de tais nós de saída é reduzido em um.

2. Repetir (1) até que todos os s_k símbolos de entrada sejam determinados.

Na Figura 3.3 se mostra o processo de decodificação para a mensagem codificada anteriormente na Figura 3.2.

3.2.4 Distribuição de graus

A distribuição de graus é fundamental para o bom desempenho do código LT, ela deve ser projetada com o propósito de garantir que todos os símbolos de entrada estejam conectados, que a complexidade seja mínima garantindo que as conexões no grafo $\mathcal{G}$ em número sejam poucas, que o processo possa iniciar e continuar até conseguir recuperar a totalidade da mensagem original, que o conjunto de símbolos liberados mais ainda não processados

PUC-Rio - Certificação Digital Nº 0721360/CA

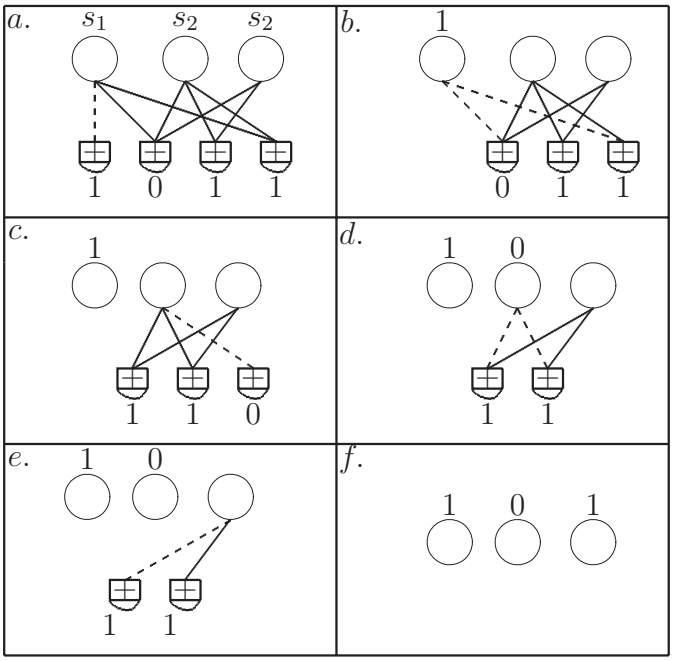

Figura 3.3: Processo de decodificação.

PUC-Rio - Certificação Digital Nº 0721360/CA

(*ripple*) seja mantido em um tamanho ideal. Deve-se procurar demandar em media, a recepção de menor quantidade de símbolos de saída possível com o fim de garantir sucesso da codificação e utilizar o menor número possível de operações para gerar um símbolo de saída. Isto se consegui mantendo-se um número baixo em o grau médio dos símbolos de saída. O ripple deve ser o menor possível ao longo do processo de codificação para evitar uma repetição muito grande no referido processo, pois se um símbolo liberado já se encontra nó ripple, nenhuma informação adicional vai ser obtida a partir da decodificação deste símbolo desperdiçando esforço computacional. Assim quanto menor o ripple menor a probabilidade de haver liberação de símbolos os quais já se encontrem no mesmo. O conjunto de símbolos liberados e não processados deve ser mantido em um tamanho grande o suficiente tal que não fique vazio antes da conclusão bem sucedida da decodificação gerando o fim da mesma sem que todos os símbolos de entrada sejam recuperados. O ideal numa distribuição seria aquela que liberasse exatamente um símbolo a cada iteração, pois assim o ripple seria mantido nó menor tamanho possível (um símbolo). Visto que o símbolo processado a cada iteração seria imediatamente substituído por outro mantendo o ripple com apenas um símbolo ao longo de tudo o processo de decodificação, tendo aquele uma alta probabilidade de nunca desaparecer antes da codificação ser completada com sucesso.

3.2.5
Distribuição de graus Sóliton Robusta

Satisfazendo as condições anteriormente mencionadas Luby propõe um a distribuição Soliton ideal $\rho(d)$ para um tamanho de mensagem k [5], definida como:

$$\rho(d) = \begin{cases} \frac{1}{k}, & \text{para } d = 1 \\ \frac{1}{d(d-1)}, & \text{para } d = 2, 3, ..., k. \end{cases} \tag{3-4}$$

O grau esperado usando esta distribuição é aproximadamente $\log_e k$. Na prática esta distribuição não tem um bom desempenho, já que em muitos casos não se consegui sempre nós com grau um em algum ponto do processo fazendo que este termine sem sucesso, mais com a introdução de dois novos parâmetros c y δ se consegui uma distribuição que garanta que o número esperado de nós de grau um seja aproximado de $S \equiv c\ \log_e(k/\delta)\sqrt{k}$. O parâmetro δ e um limitante da probabilidade de falha do processo de decodificação depois de que um certo número de pacotes sejam recebidos. O parâmetro c é uma constante de ordem 1, um estudo mais detalhado destes parâmetros se encontra-se em [3]. Usando estes novos parâmetros Define-se uma função positiva da seguinte forma:

$$\tau(d) = \begin{cases} \frac{S}{dk}, & \text{para } d = 1, 2, 3...(\frac{k}{S} - 1) \\ \frac{S}{k} \log\left(\frac{S}{\delta}\right), & \text{para } d = \frac{k}{S}, \\ 0, & \text{para } d > \frac{k}{S} \end{cases} \tag{3-5}$$

Somando as distribuições e normalizando se obtém:

$$\mu(d) \ = \ \frac{\rho(d) + \tau(d)}{\beta}, \quad \text{para } 1 \leq d \leq k \tag{3-6}$$

$$\text{onde } \ \beta \ = \ \sum_{d=1}^{k} \left(\rho(d) + \tau(d)\right) \ \text{ é uma constante de normalização.}$$

(3-6)se conhece como distribuição Sóliton Robusta.

A Figura 3.4 mostra a distribuição de grau ótima Sóliton Robusta para $k = 10^4$, $c = 0.2$ e $\delta = 0.1$. Este valor de c foi escolhido para uma melhor visualização do gráfico.

3.2.6

PUC-Rio - Certificação Digital Nº 0721360/CA

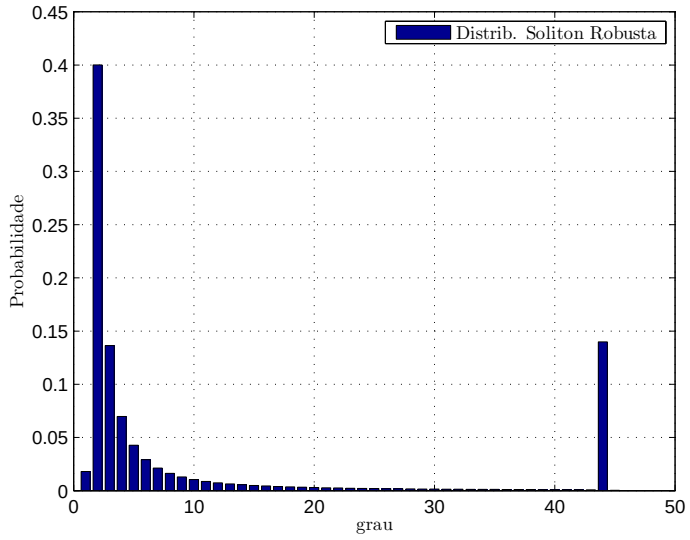

Figura 3.4: A distribuição Sóliton Robusta para o caso $k = 10^4$, $c = 0.2$ e $\delta = 0.1$.

PUC-Rio - Certificação Digital Nº 0721360/CA

Distribuição de graus Sóliton Robusta Melhorada

Na distribuição Soliton robusta Tee, Nguyen *et al.* observaram, em [14], a existência de alguns graus d_i, os quais têm uma probabilidade $\mu(d_i)$ tão baixa, que o número de símbolos dado pelo produto de $\mu(d_i)$ e k pode ser menor que 1, indicando a ausência de símbolos codificados com estes graus. Em alguns casos, a distribuição Sóliton Robusta, pode levar a um prematuro fracasso na decodificação e uma conseqüente perda de símbolos durante o processo de decodificação, a menos que o número de símbolos redundantes seja elevado. Quando o processo de decodificação é interrompido, devido à ausência de símbolos codificados de grau 1, precisamos receber mais símbolos para a recuperação de todos os símbolos originais.

A proposta de Tee, Nguyen *et al.*, portanto, é melhorar o comportamento da distribuição Sóliton Robusta no caso em que $\mu(d_i)\cdot \mathrm{k} < 1$.

Para maximizar a freqüência relativa de ter símbolos de grau 1, ou seja, todos aqueles graus d_i, onde d_i representa o termo de grau-i da distribuição, que satisfazem (3-7) têm redefinidas as suas probabilidades de distribuição a zero e a soma de todas essas probabilidades de distribuição são adicionadas à distribuição de probabilidade de grau 1. A distribuição Sóliton Robusta Melhorada é mostrada na Figura 3.5.

$$\begin{cases} \left(\frac{1}{d(d-1)} + \frac{S}{kd}\right) \cdot \frac{k}{\beta} < 1 & \text{para } 2 \leq d \leq (\frac{k}{s} - 1), \\ \left(\frac{1}{d(d-1)}\right) \cdot \frac{k}{\beta} < 1 & \text{para } (\frac{k}{s} + 1) \leq d \leq k. \end{cases} \tag{3-7}$$

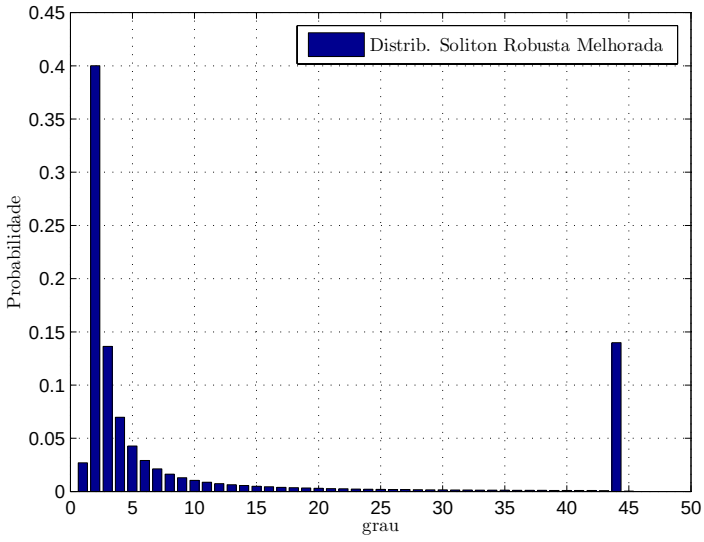

Figura 3.5: A distribuição Sóliton Robusta Melhorada para o caso $k = 10000$, $c = 0.2$ e $\delta = 0.1$

A continuação, nas Figuras 3.6 e 3.7 se apresentam os histogramas que mostram a quantidade de símbolos requeridos para conseguir uma codificação com sucesso, em um canal ideal, para uma mensagem de comprimento $k = 1000$ usando as distribuições Soliton Robusta e Soliton Robusta Melhorada. Se fizeram 1000 simulações em cada caso.

No histograma obtido para a distribuição Sóliton robusta, Figura 3.6, encontra-se que a variação na quantidade de símbolos requeridos para recuperar a mensagem original e maior respeito à Sóliton robusta melhorada, Figura 3.7. A Figura 3.8 apresenta-se a CDF (*Cumulative Distribution Function*) baseada nos histogramas anteriores, aqui pode se notar uma probabilidade sempre maior o igual de recuperar a mensagem original para um certa quantidade de símbolos codificados quando é usada a distribuição Sóliton robusta Melhorada, mostrando assim uma superioridade no desempenho respeito à Sóliton robusta, por este fato neste trabalho foi usada esta distribuição com o propósito de obter melhores resultados.

PUC-Rio - Certificação Digital Nº 0721360/CA

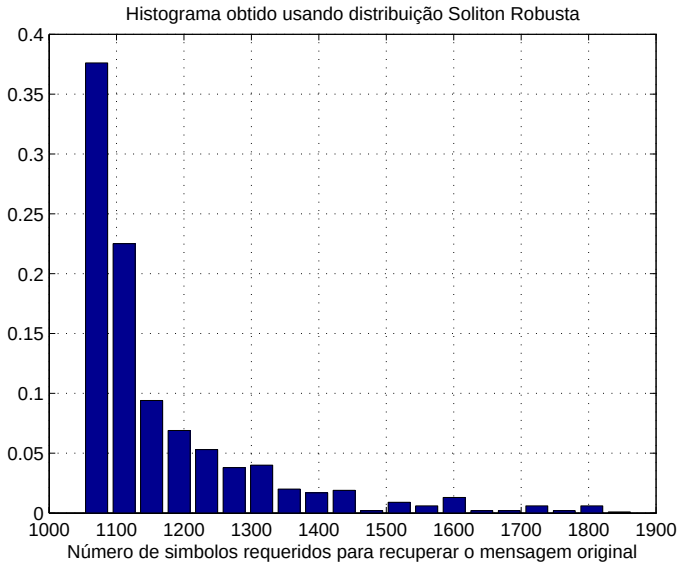

Figura 3.6: Histograma normalizado de símbolos requeridos para uma decodificação com sucesso para Sóliton Robusta para o caso $k = 1000$, $c = 0.2$ e $\delta = 0.1$

PUC-Rio - Certificação Digital Nº 0721360/CA

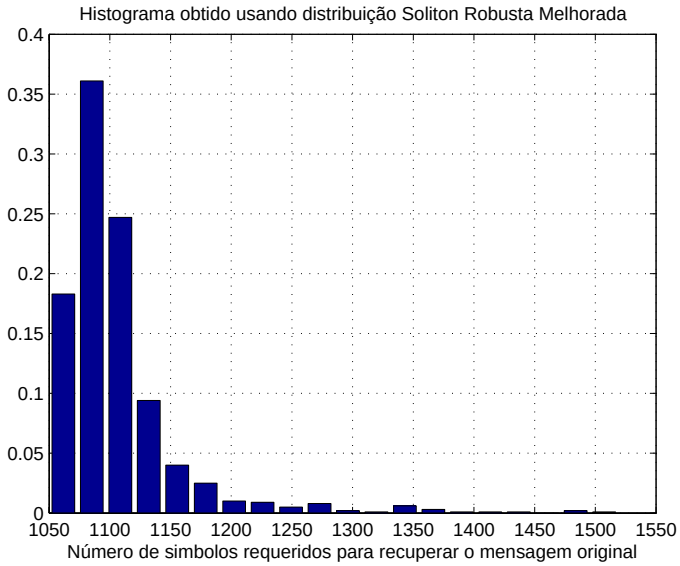

Figura 3.7: Histograma normalizado de símbolos requeridos para uma decodificação com sucesso para Sóliton Robusta Melhorada para o caso $k = 1000$, $c = 0.2$ e $\delta = 0.1$

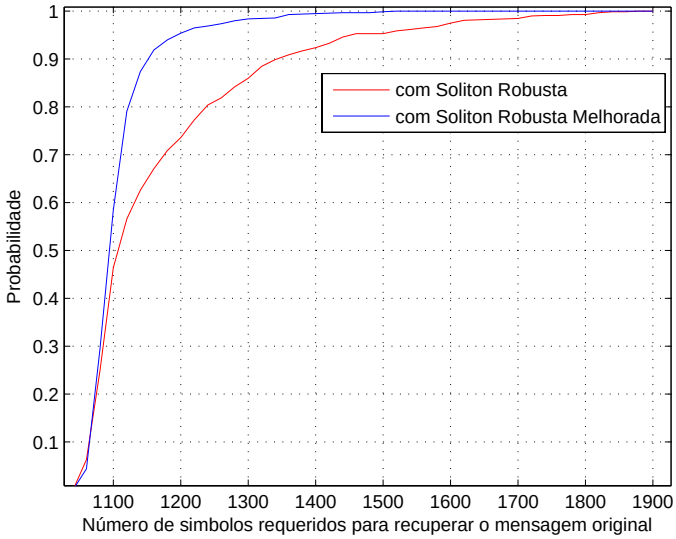

Figura 3.8: CDF baseada nos histogramas das Figuras 3.6 e 3.7

PUC-Rio - Certificação Digital Nº 0721360/CA

3.3 Resumo

Neste Capítulo foi apresentado o conceito de codificação para canal e sua importância para garantir confiabilidade na transmissão de dados, foram introduzidos os códigos fontanais e foi apresentado o código LT, fazendo uma descrição de seus processos de codificação e decodificação e a relevância da distribuição de graus para o desempenho em termos do número de símbolos necessários para uma codificação com sucesso. Mostrou-se através de simulação a superioridade da distribuição Soliton Robusta Melhorada respeito a distribuição Soliton Robusta, resultado útil para posteriores implementações.