UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Códigos de Máxima Distância de Posto e suas Aplicações em Codificação de Rede

Monografia submetida ao curso de Engenharia Elétrica da Universidade Universidade Federal de Santa Catarina como parte dos requisitos para a aprovação na disciplina EEL7890 - Projeto Final

Ricardo Bohaczuk Venturelli

Orientador: Danilo Silva

Florianópolis, 11 de março de 2014.

RICARDO BOHACZUK VENTURELLI

CÓDIGOS DE MÁXIMA DISTÂNCIA DE POSTO E SUAS APLICAÇÕES EM CODIFICAÇÃO DE REDE

Monografia submetida ao curso de Engenharia Elétrica da Universidade Federal de Santa Catarina como requisito para aprovação da disciplina EEL7890 - Projeto Final.
Orientador: Prof. Danilo Silva, Ph.D.

FLORIANÓPOLIS
2013

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Venturelli, Ricardo Bohaczuk Venturelli
Códigos de Máxima Distância de Posto e suas Aplicações em Codificação de Rede / Ricardo Bohaczuk Venturelli Venturelli ; orientador, Danilo Silva - Florianópolis, SC, 2014.
80 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico. Graduação em Engenharia Elétrica.

Inclui referências

1. Engenharia Elétrica. 2. Codificação de Rede. 3. Códigos de Máxima Distância de Posto. 4. Códigos de Gabidulin. 5. Minimização do Overhead. I. Silva, Danilo. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Elétrica. III. Título.

Ricardo Bohaczuk Venturelli

CÓDIGOS DE MÁXIMA DISTÂNCIA DE POSTO E SUAS APLICAÇÕES EM CODIFICAÇÃO DE REDE

Esta Monografia foi julgada adequada no contexto da disciplina EEL7890 - Projeto Final, e aprovada em sua forma final pelo Departamento de Engenharia Elétrica da Universidade Federal de Santa Catarina.

Florianópolis, 25 de fevereiro de 2014.

Banca examinadora:

Prof. Danilo Silva, Ph.D.
Orientador
Universidade Federal de Santa Catarina

Prof. Bartolomeu Ferreira Uchôa Filho, Ph.D.
Universidade Federal de Santa Catarina

Roberto Wanderley da Nóbrega, D.Sc.
Universidade Federal de Santa Catarina

Para meus pais Ivo e Marly
e para minha querida Thais.

Agradecimentos

Desejo expressar meu reconhecimento a todos que, de uma maneira ou outra, colaboraram na realização deste trabalho.

ao Professor Danilo Silva pela sua orientação, por sempre tentar explicar as coisas ao invés de simplesmente dar a resposta e por sempre acreditar que eu possa render mais do que eu acredito ;

ao amigo Roberto Wanderley da Nóbrega por sua paciência em explicar diversas vezes a mesma coisa até que eu finalmente entendesse, por todo auxílio para utilizar a plataforma SAGE ou então o LaTeX, e finalmente "obrigado por nada";

aos professores Bartolomeu Ferreira Uchôa Filho, Carlos Aurélio Faria, Leonardo Silva Resende, Raimes Moraes, Djones Vinicius Lettnin, Eduardo Augusto Bezerra que de uma maneira ou de outra sempre estavam incentivando para o término deste trabalho;

aos amigos Tomás Grimm, Bruno Fontana da Silva, Maria Cláudia de Almeida Castro, Rodrigo Farias, André José da Silva, Paulo Ricardo Branco da Silva, Robinson Pizzio, Haroldo Corrêa Garcia Neto, Gabriel Mariano Marcelino, João Pedro dos Reis, Julián Jair Lopez Salamanca, Willian Jamir da Silva, Hiago Murilo Horstmann e demais amigos do Laboratório de Comunicações e Sistemas Embarcados pelas horas de confraternização;

aos amigos Ednei Rodrigues Júnior, Gustavo Saran, Rafael Alves de Araújo, Alair José da Silva Júnior e demais amigos da graduação pelas incontáveis horas de estudo e pelas festividades durante a graduação;

aos amigos Vitor Camargo, Luís Gustavo Aleixo, Carlos Leite, Adelmo Oliveira, Marcelo Ferrantini, Thierri de Freitas, Eduardo Medeiros Schütt e demais amigos que conheci pela internet por mostrar que para a amizade não existe fronteiras;

aos meus pais Ivo José Venturelli e Marly Bohaczuk Venturelli pelo apoio incansável, não só durante a graduação, como em toda minha vida;

a minha irmã Rafaela Bohaczuk Venturelli e seu noivo André Felipe Knop por me ajudarem com a estadia em Florianópolis, mesmo eu não contribuindo tanto para a organização da casa;

a minha querida Thaís Simões Camargo que em tão pouco tempo se tornou uma das minha maiores alegrias.

RESUMO

Codificação de rede é um novo paradigma na transmissão de pacotes em uma rede. Diferente da técnica de roteamento, na qual um nó-intermediário transmite uma réplica de um dos pacotes por ele recebido, o princípio da codificação de rede é que um nó-intermediário transmita uma função dos pacotes recebidos. Essa função pode ser, em particular, uma combinação linear entre os pacotes. Embora a codificação de rede possa melhorar as taxas de transmissão, quando uma rede está sujeita a erros e apagamentos, a codificação de rede apresenta alguns desafios. Mesmo que sejam utilizados códigos corretores de erros tradicionais, isso pode não ser suficiente para recuperar a mensagem. Devido à mistura de pacotes, o número de erros pode ficar acima da capacidade de correção do código. Entretanto, tais erros podem ser modelados como erros de posto, o que nos permite recuperar a mensagem se utilizarmos códigos com métrica de posto, como por exemplo os *códigos de Gabidulin*. Os códigos de Gabidulin foram introduzidos em 1985 e tem como uma das características a correção de erros de posto. O codificador e o decodificador para esses códigos foram implementados ao longo deste trabalho. Também foi desenvolvido um simulador de redes, que faz uso desses códigos, para criar um cenário completo de codificação de rede. Através das simulações, foi possível calcular, para qualquer topologia de rede, o *overhead* na transmissão de pacotes. Esse overhead quantifica a proporção de dados não-úteis transmitidos, incorporando tanto a parte da mensagem designada ao cabeçalho da codificação de rede assim como o número de pacotes não inovadores recebidos pelo nó-destino. Mostramos neste trabalho que utilizando códigos de Gabidulin é possível minimizar este overhead em qualquer topologia de rede.

Palavras-chave: Códigos de Gabidulin. Codificação de rede. Códigos corretores de erros. Minimização do overhead.

ABSTRACT

Network coding is a new paradigm in packet transmission on a network. Unlike routing, in which an intermediate node sends a copy of one of the packets that it has received, the principle of network coding is that an intermediate node sends a function of the packets received. This function can be, in particular, a linear combination of such packets. Although network coding can improve the transmission rate, it faces some challenges when the network is subject to errors and erasures. Even if a traditional error-correcting code is used, this may not be enough to recover the message. Due to packet mixing, the number of errors may be beyond the error-correcting capability of the code. However, these errors can be modeled as rank errors, which allow us to recover the message if we use rank-metric codes, such as *Gabidulin codes*. Gabidulin codes were introduced in 1985 and one of its features is rank error correction. The encoder and decoder for these codes were implemented throughout this work. A network simulator, which makes use of these codes, has also been developed to create a complete simulation environment for network coding. Throughout the simulations, it was possible to calculate, for any network topology, the *overhead* in packet transmission. This overhead quantifies the proportion of non-useful data in the transmission, incorporating both the header of network coding as well as the number of non-innovative packets received by the sink node. We show in this work that using Gabidulin codes is possible to minimize this overhead for any network topology.

Keywords: Gabidulin codes. Network coding. Error-correcting codes. Minimizing the overhead.

Sumário

1 Introdução **1**
1.1 Organização do Trabalho 3

2 Codificação de Rede **5**
2.1 Preliminares 6
2.1.1 Grafos 6
2.1.2 Fluxo de Informação 7
2.2 Codificação de Rede Linear e Aleatória 9
2.2.1 Cabeçalho de Identificação 11
2.2.2 Transmissão via Gerações 12
2.3 Erros e Deficiência de Posto 13

3 Códigos de Gabidulin **15**
3.1 Preliminares 16
3.1.1 Bases sobre Corpos de Extensão 16
3.1.2 Bases Normais 17
3.1.3 Polinômios Linearizados 18
3.2 Códigos com Métrica de Posto 19
3.3 Códigos de Gabidulin 20
3.4 Decodificação 21
3.4.1 Algoritmo de Decodifição ESP 22

3.4.2 Algoritmo de Decodicação ELP 24
3.4.3 Algoritmo de Berlekamp-Massey modificado 25
3.4.4 Algoritmo de Gabidulin 26
3.5 Decodificação Generalizada 28
3.5.1 Algoritmo de Decodifição Generalizado ESP 29
3.5.2 Algoritmo de Decodifição Generalizado ELP 30
3.6 Codificação 33
3.7 Considerações sobre a Complexidade 34
3.8 Aplicação dos Códigos de Gabidulin na Codificação de Rede 34
3.9 Implementação 38

4 Aplicação: Minimização do Overhead 41
4.1 Overhead 42
4.1.1 Overhead de Dependência Linear 42
4.1.2 Overhead de Cabeçalho 43
4.1.3 Outros Overheads 44
4.1.4 Overhead Intrínseco 44
4.2 Minimizando o Overhead Intrínseco 45
4.3 Redes mais Complexas 47
4.4 Simulador de Redes 48
4.5 Resultados 50

5 Conclusão 55
5.1 Trabalhos futuros 56

Lista de Figuras

1.1 Rede utilizada para mostrar uma das vantagens da codificação de rede. 2

2.1 Rede borboleta, exemplo padrão encontrado na literatura. O nó-fonte s deseja se comunicar com os nós-destino t_1 e t_2. 6
2.2 Redes unidifusão e multidifusão. A topologia da rede pode ser qualquer uma, o importante é o número de nós-fontes e nós-destino. 8
2.3 Rede utilizada no exemplo. O nó-fonte s deseja se comunicar com o nó-destino t. 10
2.4 Exemplo esquemático. O vetor de erro E_i é somado na aresta i. 13

4.1 Representação do overhead intríseco nos pacotes recebidos pelo nó-destino. 44
4.2 Overhead intrínseco para a rede com um nó-fonte e um nó destino. 46
4.3 Transformando os k pacotes em n pacotes usando códigos MRD. Note que o tamanho do cabeçalho também aumenta. 47
4.4 Overhead intrínseco para a rede com um nó-fonte e um nó destino, usando códigos MRD e fazendo $\mu = \mu^*$. 48

4.5 Overhead intrínseco para a rede com um nó-fonte e dois nós-destino, usando $\mu = \mu^*$ e $k = 50$. 49
4.6 Rede com um nó-fonte e dois nós-destinos que comunicam-se entre si. 51
4.7 Rede com um nó-fonte e três nós-destinos que comunicam-se entre si. 52
4.8 Rede com um nó-fonte, um nó-destino e cinco nós-intermediários que auxiliam a comunicação. 53

CAPÍTULO 1

Introdução

A transmissão de informação em redes *peer-to-peer* (par-a-par) é cada vez mais comum, como por exemplo, em transmissão via protocolo *bit Torrent* [3] e comunicação sem fio. Com isso é crescente o interesse na área, principalmente para melhorar as taxas de transmissão existentes.

Redes tradicionais utilizam a técnica conhecida como roteamento, no qual os dispostivos eletrônicos fazem uma cópia de um dos pacotes recebidos e então essa cópia é transmitida. Para muitos casos, o roteamento é suficiente para atingir níveis aceitáveis na taxa de transmissão.

Por exemplo, na rede apresentada na Fig. 1.1, suponha que a fonte s deseja enviar 100 pacotes para ambos os destinos t_1 e t_2. Com um esquema bem coordenado de roteamento podemos atingir taxas satisfatórias. Por exemplo, s enviaria 50 pacotes para um, e outros 50 pacotes para o outro, enquanto isso, eles poderiam trocar entre si os pacotes já recebidos. Note que, com esse esquema, cada destino receberia 2 pacotes por uso do canal (neste trabalho, entende-se por uso do canal, quando todas as arestas da rede transmitirem um pacote).

Mas o que aconteceria se caso um dos enlaces que conectam os usuários falhasse? Apenas com o roteamento, seria necessário utilizar o canal mais vezes, o que diminuiria a taxa de transmissão. A codifica-

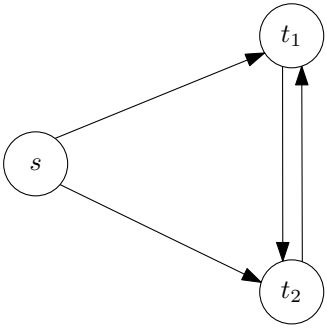


Figura 1.1: Rede utilizada para mostrar uma das vantagens da codificação de rede.

ção de rede surge como uma alternativa para esse caso [1]. A ideia da codificação de rede é que diferentes combinações de pacotes sejam enviadas aos destinos. Assim, caso um enlace falhe, ainda sim é possível atingir níveis aceitáveis de taxa.

Há diferentes maneiras de combinar pacotes. A mais simples é fazer uma soma módulo-2, equivalente a uma XOR (ou-exclusivo) bit-a-bit, dos pacotes, interpretados como sequência de bits. Entretanto, sabe-se que a soma módulo-2 nem sempre será eficiente. Como mostrado em [13, 11], considerando os pacotes como vetores sobre um corpo finito, é suficiente fazer combinações lineares desses pacotes, em que os coeficientes das combinações perteçam ao corpo finito utilizado.

Uma estratégia para deixar a codificação de rede mais eficiente é utilizando códigos corretores de erros. (Esses códigos são amplamente usados no dia-a-dia, devido ao sinal de informação que passa por um canal ruidoso.) Ao passar por um canal ruidoso, o sinal pode ser alterado, ocasionando assim, erros na mensagem transmitida, impossibilitando o receptor de identificá-la. O princípio básico dos códigos corretores de erros é a inserção de redundância, de forma que mesmo havendo trocas de bits, por exemplo, seja possível recuperar a mensagem.

Existem códigos corretores de erros desde os mais simples, como o código de repetição, até mais complexos, como os códigos Turbo e LDPC (do inglês, *Low-Density-Parity-Check*) [14]. Nesses códigos, a

mensagem pode ser vista como sendo um vetor sobre um corpo finito. Há ainda outros tipos de códigos, chamados códigos matriciais, em que a mensagem é vista como uma matriz sobre tal corpo finito.

Códigos corretores de erro e codificação de rede eram estudados de maneira desconectadas. Isto é, o estudo de um não tinha influência direta no outro. Como mostrado em [22] a união da codificação de rede com códigos matriciais corretores de erros pode se tornar bastante útil.

Dentre os principais desafios da codificação de rede, está a propagação de erros. Como são feitas combinações lineares dos pacotes, se um destes pacotes contiver erros, então todas as combinações feitas a partir desse pacote também conterá erros. Assim, o código corretor tradicional pode não ser suficiente, haja vista que o erro estará além da capacidade de correção do código.

Um outro desafio encontrado na codificação de rede é que um dispositivo eletrônico pode receber vários pacotes que não acrescentam informação dando origem ao que chamamos de *overhead*. Mesmo utilizando um corpo finito suficientemente grande, existe a possibilidade de um pacote ser linearmente dependente dos demais (como por exemplo, o pacote nulo).

Ao utilizar códigos corretores de erro com a codificação de rede, podemos minimizar (ou até mesmo eliminar) esses problemas, justificando assim o estudo desses temas.

1.1 Organização do Trabalho

A organização deste trabalho é descrita a seguir. No capítulo 2 serão apresentados os conceitos básicos da codificação de rede: como ela funciona, como podemos equacionar, e a influência dos erros. Também será apresentado um modo de recuperar a mensagem, mesmo na presença de erros.

No capítulo 3 serão introduzidos os códigos de máxima distância de posto, em particular, os códigos de Gabidulin. Tais códigos são uteis na codificação de rede, principalmente devido à sua capacidade de correção de deficiência de posto.

Já no capítulo 4 iremos unir a codificação de rede e os códigos matriciais e verificar seu impacto no *overhead* em uma rede com apagamentos. Nesse capítulo também será explicado o simulador de rede,

implementado ao longo dos estudos para esse trabalho, o qual é utilizado para comprovar experimentalmente os resultados teóricos.

Por fim, no capítulo 5 serão apresentadas algumas conclusões, assim como ideias para a continuidade do trabalho.

CAPÍTULO 2

Codificação de Rede

Em redes tradicionais a transmissão de pacotes é feita através de roteamento. Isto é, dos pacotes recebidos por um dispositivo (o qual chamaremos de nó), é feita uma cópia de um desses pacotes e enviada ao dispositivo seguinte [12].

Já em *codificação de rede*, ao invés de um simples roteamento, o pacote transmitido é uma função dos pacotes recebidos [1]. Para melhor demonstrar utilizaremos como exemplo a rede "borboleta" mostrada na Fig. 2.1. Nela, a fonte s deseja transmitir pacotes para os destinos t_1 e t_2. A rede é caracterizada pelo "gargalo" no nó b.

Suponha que a fonte deseja transmitir os pacotes x, y e z. No primeiro uso do canal a fonte transmite os pacotes x e y. O nó b escolhe transmitir o pacote x, por exemplo. Assim, ao final do primeiro uso do canal, o destino t_1 tem o pacote x, enquanto o destino t_2 tem os pacotes x e y. No segundo uso do canal, a fonte transmite os pacotes y e z. Dessa vez o nó b escolhe transmitir o pacote z. Assim ao final do segundo uso do canal, todos os destinos teriam os três pacotes. A taxa de transmissão é, portanto, $1,5$ pacotes por uso de canal.

Para a solução usando codificação de rede, suponha que a fonte deseja transmitir os pacotes x e y. No primeiro uso do canal a fonte

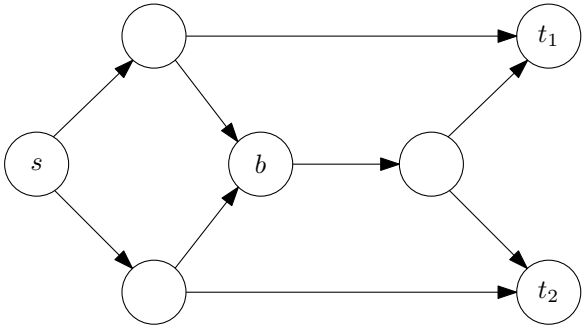


Figura 2.1: Rede borboleta, exemplo padrão encontrado na literatura. O nó-fonte s deseja se comunicar com os nós-destino t_1 e t_2.

transmite ambos os pacotes. Porém, o nó b, ao invés de escolher um desses pacotes, transmite a soma módulo 2 dos pacotes, equivalente ao ou-exclusivo (denotado por $\oplus$) bit-a-bit. O destino t_1 receberia os pacotes x e $x \oplus y$, porém recuperaria o pacote y fazendo $x \oplus (x \oplus y)$. Analogamente, acontece o mesmo no nó t_2. Assim ambos os destinos teriam os dois pacotes com um único uso do canal. Assim, a taxa de transmissão é de 2 pacotes por uso do canal.

A seguir, iremos recapitular alguns conceitos básicos de grafos e fluxo de informação. Então será apresentado o conceito de codificação de rede linear aleatória. No final desse capítulo iremos tratar do caso em que a rede pode apresentar erros e/ou apagamentos, e um método para recuperar a mensagem nesse caso.

2.1 Preliminares

2.1.1 Grafos

Como visto no exemplo anterior, é comum utilizar grafos direcionados para representar uma rede. Seja $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ um grafo direcionado, em que $\mathcal{V}$ representa o conjunto de nós (vértices) e $\mathcal{E}$ representa o conjunto de arestas. Uma aresta que ligue o nó $u \in \mathcal{V}$ ao nó $v \in \mathcal{V}$ é denotada por $(u, v) \in \mathcal{E}$. Note que essa aresta não liga o nó v ao nó u, e portanto, o grafo é dito ser *direcionado* [26].

Cada aresta pode transmitir um pacote, sem erros, por uso do canal.

Dizemos, então, que cada aresta possui *capacidade* unitária. Todavia, entre dois nós, é permitido haver arestas paralelas (que podem ser utilizadas para representar enlaces com maiores velocidades).

Para um determinado nó $v \in \mathcal{V}$ dizemos que $I(v)$ é o conjunto de arestas através das quais v recebe pacotes e $O(v)$ é o conjunto de arestas pelas quais v envia os pacotes.

Em uma rede é necessário especificar um conjunto $S \subseteq \mathcal{V}$, que é o conjuntos de nós-fonte. Os nós desse conjunto contêm os pacotes com a mensagem a ser transmitida. Outro conjunto necessário especificar é o conjunto dos nós-destino $T \subseteq \mathcal{V}$. Tais nós são aqueles interessados em receber a mensagem de pelo menos um nó-fonte. Note que um nó-fonte, por exemplo, pode estar interessado nos pacotes de um outro nó-fonte, nesse caso ele pode ser, simultaneamente, um nó-fonte e um nó-destino. Se um nó não é nó-fonte nem nó-destino então chamamo-no de nó-intermediário [12].

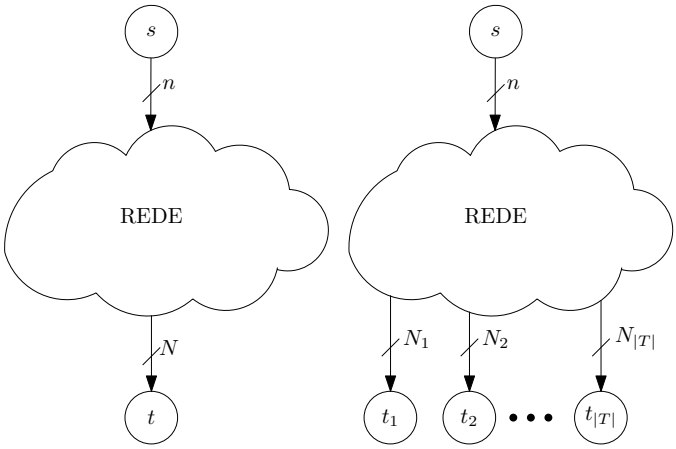
Um *caminho* entre os nós u e $v \in \mathcal{V}$ é um conjunto de arestas tais que $(u, v_1), (v_1, v_2), \ldots, (v_n, v)$. Dizemos que o nó u se comunica (ou pode se comunicar) com o nó v se existir um caminho entre eles.

Chamamos de *corte* entre os nós u e v o conjunto de arestas que se forem removidas impedem a comunicação entre esses nós, isto é, não há mais caminhos entre u e v. A *capacidade do corte* é definida como sendo a soma das capacidades de cada aresta (virtualmente) removida. Como assumimos que cada aresta tem capacidade unitária, a capacidade do corte será igual ao número de arestas do corte que devem ser removidas para impedir a comunicação entre os nós u e v [26].

O *corte mínimo* entre dois nós u e $v \in \mathcal{V}$ é o corte que apresenta a menor capacidade. Denotamos a capacidade de corte mínimo por $\text{mincut}(u, v)$.

2.1.2 Fluxo de Informação

Quando a rede apresenta apenas um nó-fonte, denotado por $s \in \mathcal{V}$, e um nó-destino, denotado por $t \in \mathcal{V}$, dizemos que estamos lidando com o caso de *unidifusão* (do inglês, *unicast*), ver Fig. 2.2a. O problema de fluxo de informação é semelhante ao problema de fluxo de mercadorias. Portanto podemos utilizar o teorema que diz que o fluxo máximo é



(a) Rede unidifusão, apenas um nó-fonte (com n arestas) e um nó-destino (com N arestas)

(b) Rede multidifusão, um nó-fonte (com n arestas) e vários nós-destino.

Figura 2.2: Redes unidifusão e multidifusão. A topologia da rede pode ser qualquer uma, o importante é o número de nós-fontes e nós-destino.

igual ao corte mínimo [1]. Assim a máxima taxa entre s e t é

$$R(s,t) = \text{mincut}(s,t).$$

Já para o caso em que há mais de um nó-destino (o conjunto de nós-destino é denotado por $T \subseteq \mathcal{V}$) e um nó-fonte, denotado por $s \in \mathcal{V}$, o qual chamamos de *multidifusão* (do inglês, *multicast*), ver Fig. 2.2b, podemos estender esse resultado. A máxima taxa de transmissão entre o nó-fonte e os nós-destino é calculada como

$$R(s,T) = \min_{t \in T} \text{mincut}(s,t). \tag{2.1}$$

O principal teorema da codificação de rede diz que podemos alcançar a máxima taxa de informação em uma rede multidifusão utilizando codificação de rede [1].

2.2 Codificação de Rede Linear e Aleatória

Ao longo desse trabalho denotaremos por $\mathbb{F}_q$ o corpo finito com q elementos, $\mathbb{F}_q^{n\times m}$ o conjunto de matrizes $n \times m$ em $\mathbb{F}_q$. Note que $\mathbb{F}_q^{1\times m}$ denota o conjunto de vetores-linha, com dimensão m, em $\mathbb{F}_q$, enquanto $\mathbb{F}_q^m$ denota o conjunto de vetores-coluna, também com dimensão m, em $\mathbb{F}_q$.

Como vimos, a codificação de rede se baseia na ideia de que nós-intermediários enviem uma função dos pacotes recebidos. Todavia a função escolhida não deve ser de alta complexidade (o que diminuiria o desempenho na transmissão de pacotes) mas também, com essa função, deseja-se alcançar a máxima taxa de transmissão.

Em [13, 11] é demonstrado que podemos utilizar combinações lineares para alcançar a máxima taxa de transmissão. Nesse caso, os pacotes são vistos como vetores ℓ-dimensionais sobre $\mathbb{F}_q$, em que q é uma potência de um número primo. Cada pacote tem, portanto, $\ell \cdot \log_2 q$ bits.

Para um nó $v \in \mathcal{V}$, definimos como $P_{I(v)}$ a matriz em que cada linha representa um pacote recebido por v e $P_{O(v)}$ a matriz em que cada linha representa um pacote enviado pelo nó v. Para o nó-fonte s, dizemos que $X \in \mathbb{F}_q^{n\times\ell}$ é a matriz que contém os pacotes que o nó s deseja transmitir, isto é, $X = \begin{bmatrix} x_1^T & x_2^T & \cdots & x_n^T \end{bmatrix}^T$, em que $x_i \in \mathbb{F}_q^{1\times\ell}$, $i = 1, \ldots, n$. Para um nó-destino $t \in T$, dizemos que $Y_t \in \mathbb{F}_q^{N\times\ell}$ é a matriz que contém os pacotes recebidos por t, isto é, $Y_t = \begin{bmatrix} y_{t,1}^T & y_{t,2}^T & \cdots & y_{t,N}^T \end{bmatrix}^T$, em que $y_{t,i} \in \mathbb{F}_q^{1\times\ell}$, $i = 1, \ldots, N$. Se o nó-destino é previamente especificado, então o subscrito t é omitido.

Em um nó $v \in \mathcal{V}$ qualquer, podemos relacionar $P_{I(v)}$ com $P_{O(v)}$ atraves da equação linear

$$P_{O(v)} = L_v P_{I(v)},$$

em que L_v é chamada matriz de transferência local.

Por linearidade, $P_{I(v)}$ é uma combinação linear de X, para todo $v \in \mathcal{V}$. Assim para o nó-destino t, podemos relacionar Y com X através da equação

$$Y = AX, \tag{2.2}$$

em que $A \in \mathbb{F}_q^{N\times n}$ é a chamada matriz de transferência global. Note

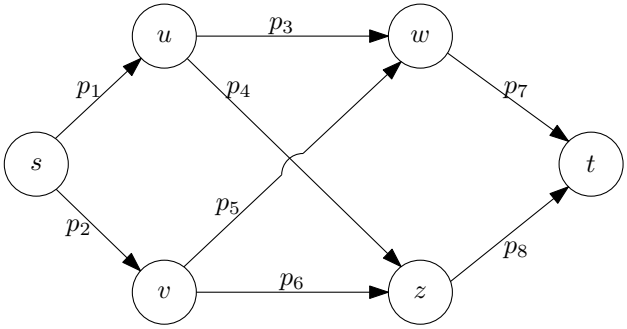


Figura 2.3: Rede utilizada no exemplo. O nó-fonte s deseja se comunicar com o nó-destino t.

que agora temos um sistema linear e, portanto, podemos recuperar X a partir de Y desde que posto $A \geq n$.

Exemplo. Suponha a rede apresentada na Fig. 2.3, em que o nó-fonte s deseja transmitir os pacotes x_1 e x_2 para o nó-destino t.

Por convêniencia, iremos supor que $P_{I(s)} = \begin{bmatrix} x_1^T & x_2^T \end{bmatrix}^T$, isto é, iremos imaginar que o nó s receba uma combinação de pacotes de tal forma que o resultado dessa combinação seja os pacotes x_1 e x_2. Assim podemos dizer que o nó fonte enviará

$$\underbrace{\begin{bmatrix} p_1 \\ p_2 \end{bmatrix}}_{P_{O(s)}} = \underbrace{\begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix}}_{L_s} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{P_{I(s)}},$$

e os demais nós enviarão

$$\underbrace{\begin{bmatrix} p_3 \\ p_4 \end{bmatrix}}_{P_{O(u)}} = \underbrace{\begin{bmatrix} a_3 \\ a_4 \end{bmatrix}}_{L_u} \underbrace{\begin{bmatrix} p_1 \end{bmatrix}}_{P_{I(u)}} \qquad \underbrace{\begin{bmatrix} p_5 \\ p_6 \end{bmatrix}}_{P_{O(v)}} = \underbrace{\begin{bmatrix} a_5 \\ a_6 \end{bmatrix}}_{L_v} \underbrace{\begin{bmatrix} p_2 \end{bmatrix}}_{P_{I(v)}}$$

$$\underbrace{\begin{bmatrix} p_7 \end{bmatrix}}_{P_{O(w)}} = \underbrace{\begin{bmatrix} a_7 & a_8 \end{bmatrix}}_{L_w} \underbrace{\begin{bmatrix} p_3 \\ p_5 \end{bmatrix}}_{P_{I(w)}} \qquad \underbrace{\begin{bmatrix} p_8 \end{bmatrix}}_{P_{O(z)}} = \underbrace{\begin{bmatrix} a_9 & a_{10} \end{bmatrix}}_{L_z} \underbrace{\begin{bmatrix} p_4 \\ p_6 \end{bmatrix}}_{P_{I(z)}}$$

Note que $p_7 = y_1$ e $p_8 = y_2$ são os pacotes recebidos pelo nó-destino

t. Resolvendo recursivamente obteremos

$$Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} a_7a_3a_1 & a_8a_5a_2 \\ a_9a_4a_1 & a_{10}a_6a_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = AX.$$

Considerando que o nó-destino conheça A e que posto $A = 2$ então o nó-destino pode recuperar a mensagem enviada pelo nó-fonte. □

É natural se questionar como os coeficientes da combinação linear devem ser escolhidos. Em [10] é demonstrado um algoritmo que requer tempo polinomial. Todavia [9] mostra que podemos escolher os coeficientes de forma aleatória e uniforme sobre o corpo finito $\mathbb{F}_q$. A probabilidade de um pacote recebido ser linearmente dependente aos demais (o que não alteraria o posto A) é diminuida conforme aumentamos o tamanho do corpo finito.

2.2.1 Cabeçalho de Identificação

Um problema encontrado em redes que utilizam codificação de rede linear aleatória é que o nó-destino não tem conhecimento de A. Isso porque, os coeficientes de A dependem das combinações lineares feitas nos nós-intermediários e por sua vez, cada coeficiente da combinação linear é escolhido de forma aleatória. Assim, os coeficientes da matriz A também são aleatórios.

Uma ideia bem simples para resolver esse problema é a inserção de um cabeçalho de identificação em cada pacote enviado pelo nó-fonte [2]. Desse modo, o pacote x_i será a concatenação de um vetor n-dimensional cuja i-ésima entrada é 1 e as demais são 0, com o vetor que realmente contém a mensagem a ser transmitida (chamada agora de x_i'). Em outras palavras, teremos que $X = \begin{bmatrix} I_n & X' \end{bmatrix} \in \mathbb{F}_q^{n \times \ell}$, em que I_n é a matriz identidade $n \times n$ e $X' \in \mathbb{F}_q^{n \times \ell - n}$.

Portanto, para o nó-destino $t \in T$, teremos que

$$Y = AX = A \begin{bmatrix} I_n & X' \end{bmatrix} = \begin{bmatrix} A & AX' \end{bmatrix}, \tag{2.3}$$

ou seja, as n primeiras colunas de Y correspondem à matriz de transferência global. E, é possível recuperar a mensagem, desde que posto $A = n$.

A codificação de rede linear aleatória com cabeçalho de identificação é útil no sentido que a topologia da rede pode ser desconhecida, o que a torna mais geral. No entanto, ainda sim, é um esquema simples, que pode alcançar a máxima taxa de transmissão desde que o corpo finito utilizado tenha tamanho suficientemente grande [9].

2.2.2 Transmissão via Gerações

Para que o nó-destino recupere a mensagem enviada pelo nó-fonte, considerando que posto $A = n$, é necessário, basicamente, resolver um sistema linear usando eliminação Gauss-Jordan, por exemplo. Todavia, sabemos que a complexidade da eliminação Gauss-Jordan cresce com n^3, o que poderia tornar a recuperação da mensagem muito complexa. Além disso, o tamanho do cabeçalho também cresce conforme aumentamos o número de pacotes.

Visando diminuir tanto a complexidade para recuperar a mensagem quanto o tamanho do cabeçalho dividimos a mensagem original em L gerações cada uma com k pacotes (de modo que $n = Lk$) [2]. Iremos assumir que as gerações são disjuntas, isto é, cada pacote faz parte de apenas uma geração. Também assumiremos que pacotes de diferentes gerações não podem ser usados para gerar um novo pacote durante a codificação.

Quando a matriz de transferência de uma dada geração tem posto k, dizemos que tal geração está completa. O nó-destino pode recuperar os pacotes enviados pelo nó-fonte tão logo todas as gerações estejam completas. Note que a complexidade será Lk^3 operações em $\mathbb{F}_q$ que é menor do que $n^3 = L^3k^3$ operações.

Entretanto, alguns desafios surgem ao usar transmissão via geração. Uma geração, que já esteja completa, pode continuar recebendo pacotes. Note que com isso, o nó-destino receberá pacotes redundantes em uma geração, enquanto outra geração não está completa. Assim, será necessário que o nó-destino receba mais pacotes para que todas as gerações fiquem completas. Uma estratégia para minimizar isso é mostrada em [24]. Nesse trabalho iremos trabalhar com geração única, então não será necessário lidar com esse desafio.

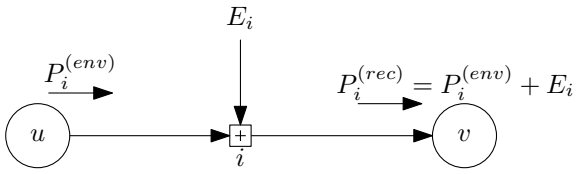


Figura 2.4: Exemplo esquemático. O vetor de erro E_i é somado na aresta i.

2.3 Erros e Deficiência de Posto

Até então, consideramos o caso em que todos os pacotes transmitidos por um nó são recebidos pelo nó subsequente, sem erros. Todavia, na prática, sabemos que nem sempre isso acontece.

Suponha que o nó-destino receba a mesma quantidade de pacotes enviados pelo nó-fonte, isto é, $N = n$. Note que esse é o menor valor de N para que seja possível que a matriz de trasferência global A tenha posto n. Isso significa que é o menor número de pacotes que o nó-destino deve receber para poder realizar a decodificação. Em uma rede qualquer, um pacote enviado pode não ser recebido pelo nó subsequente, o que chamamos de *apagamento*. Isso equivale ao nó receber um pacote nulo (todos os bits 0). As combinações seguintes não poderão utilizar este pacote. Isso pode fazer com que algumas linhas da matriz A sejam linearmente dependente. Dizemos então que a matriz A apresenta um *deficiência de posto* [22]. A deficiência de posto (ρ) é definida como sendo $\rho \triangleq n - \text{posto}\, A$.

Uma maneira simples de se lidar com a deficiência de posto seria ficar esperando o nó-destino receber mais pacotes até que $\text{posto}\, A = n$.

Há ainda o caso em que pode haver *erros* na transmissão de um pacote, isto é, um determinado nó envia um pacote, porém o nó subsequente recebe outro pacote. Mais precisamente, suponha que um nó envie pela aresta i o pacote $P_i^{(env)}$, e o nó subsequente receba o pacote $P_i^{(rec)}$. Se $P_i^{(rec)} = P_i^{(env)} + E_i$, em que $E_i \neq 0$, então dizemos que ocorreu um *erro* [20], ver Fig. 2.4. Note que se $E_i = -P_i^{(env)}$ então ocorreu um apagamento. Obviamente, se $E_i = 0$, então não há erros na aresta i.

Seja $|\mathcal{E}|$ o número de arestas na rede, as quais são indexadas por $1, \ldots, |\mathcal{E}|$. Seja $E \in \mathbb{F}_q^{|\mathcal{E}| \times \ell}$ a matriz em que, a i-ésima linha contém o vetor de erro, E_i, somado na aresta i, $i = 1, \ldots, |\mathcal{E}|$. O número de

linhas não-nulas de E corresponde ao número de pacotes com erros inseridos na rede. A equação que caracteriza uma rede, que pode haver erros, é

$$Y = AX + FE,$$

em que Y, X e A são definidas como anteriormente e $F \in \mathbb{F}_q^{N \times |\mathcal{E}|}$ é matriz correspondente as combinações lineares aplicadas nas linhas de E.

Suponha que t arestas tenham erros inseridos, ou seja, a matriz E apresenta t linhas não-nulas. (Note que, t deve ser menor que posto A, pois, caso contrário, poderíamos cair no cenário trivial de comunicação em que Y é matriz nula.) Seja $Z = F'E' \in \mathbb{F}_q^{N \times \ell}$, em que $E' \in \mathbb{F}_q^{t \times \ell}$ é a matriz que contém as linhas não-nulas de E e $F' \in \mathbb{F}_q^{N \times t}$ é uma sub-matriz de F, tal que posto $Z = t$. Desse modo, podemos reescrever a equação que caracteriza a rede como

$$Y = AX + Z. \tag{2.4}$$

Os erros podem acontecer por diversos motivos. Por exemplo, pode haver uma falha na decodificação da camada física, ou ainda a rede pode apresentar um nó malicioso, que quer prejudicar a comunicação. Além disso, nessas redes ocorre a chamada *propagação de erros*, ou seja, se um determinado pacote contiver erros, todos os pacotes formados a partir desse pacote também irão conter erros. O uso de códigos corretores de erro que utilizam como métrica a distância de Hamming se tornam ineficazes nesse caso, uma vez que, com a propagação de erros, o número de pacotes com erros estaria além da capacidade de correção do código.

CAPÍTULO 3

Códigos de Gabidulin

No capítulo anterior vimos uma estratégia para lidar com erros e apagamentos. Essa estratégia consiste em utilizar códigos matriciais corretores de erros. Nesse capítulo iremos estudar tais códigos, com maior destaque para uma família de códigos conhecida como *códigos de Gabidulin*. Os códigos de Gabidulin apresentam a melhor relação entre a capacidade de correção e a redundância inserida, o que os torna interessante em aplicações práticas.

Ao longo desse capítulo serão assumidos alguns conceitos básicos de corpos finitos e corpos de extensão. Para maiores detalhes, sugerimos a leitura de [7].

A seguir iremos rever alguns conceitos básicos que serão utilizados ao longo do capítulo. Então, serão introduzidos os códigos com métrica de posto, em especial os códigos de Gabidulin. Será explicado como é feito o processo de codificação e decodificação (tanto o caso mais simples, quanto a decodificação generalizada) dos códigos de Gabidulin. No final desse capítulo teremos algumas considerações sobre a complexidade dos métodos usados para a decodificação.

3.1 Preliminares

3.1.1 Bases sobre Corpos de Extensão

Seja $\mathbb{F}_q$ um corpo finito com q elementos, em que q é uma potência de primo. Seja $\mathbb{F}_{q^m}$ sua m-ésima extensão. Sabemos que todo corpo de extensão é um espaço vetorial sobre seu corpo base [15], então podemos pensar em uma base $\mathcal{A} = \{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$ de $\mathbb{F}_{q^m}$ sobre $\mathbb{F}_q$, em que $\alpha_i \in \mathbb{F}_{q^m}$, $i = 0, 1, \ldots, m-1$. Por simplicidade denotaremos o vetor coluna que forma base por $[[\alpha]]$, i.e, $[[\alpha]] = \begin{bmatrix} \alpha_0 & \alpha_1 & \cdots & \alpha_{m-1} \end{bmatrix}^T$.

Seja $a \in \mathbb{F}_{q^m}$. Podemos representá-lo, de forma única, como $a = \sum_{i=0}^{m-1} a_i \alpha_i = \underline{a} \cdot [[\alpha]]$, em que, $\underline{a} = \begin{bmatrix} a_0 & a_1 & \cdots & a_{m-1} \end{bmatrix} \in \mathbb{F}_q^{1\times m}$, $a_i \in \mathbb{F}_q$, $i = 0, 1, \ldots, m-1$. Por sua vez, um vetor $m-$dimensional com coeficientes em $\mathbb{F}_q$ pode ser representado, também de forma única, por um elemento de $\mathbb{F}_{q^m}$. Com isso temos uma bijeção entre $\mathbb{F}_{q^m}$ e $\mathbb{F}_q^{1\times m}$. Representaremos essa bijeção por $\varphi : \mathbb{F}_{q^m} \to \mathbb{F}_q^{1\times m}$, em que, para todo $a \in \mathbb{F}_{q^m}$, $\varphi(a) = \underline{a}$ se, e somente se $a = \underline{a} \cdot [[\alpha]]$. A função inversa $\varphi^{-1} : \mathbb{F}_q^{1\times m} \to \mathbb{F}_{q^m}$ é tal que $\varphi^{-1}(\underline{a}) = a$.

Podemos extender essa bijeção para o caso de vetores em $\mathbb{F}_{q^m}^n$ e matrizes em $\mathbb{F}_q^{n\times m}$. Seja $a = \begin{bmatrix} a_0 & a_1 & \cdots & a_{n-1} \end{bmatrix}^T$ um vetor n-dimensional, em que $a_i \in \mathbb{F}_{q^m}$, $i = 0, \ldots, n-1$. Então dizemos que

$$\varphi(a) = \begin{bmatrix} \varphi(a_0) \\ \varphi(a_1) \\ \vdots \\ \varphi(a_{n-1}) \end{bmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,m-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,m-1} \end{bmatrix}$$

se e somente se $a_i = \sum_{j=0}^{m-1} a_{i,j}\alpha_j$, $i = 0, \ldots, n-1$, em que $a_{i,j} \in \mathbb{F}_q$, $i = 0, \ldots, n-1$ e $j = 0, \ldots, m-1$.

Exemplo. Seja $q = 2$, $m = 6$ e $n = 5$. Seja α um elemento primitivo de $\mathbb{F}_{2^6}$ e seja[1] $\mathcal{A} = \{1, 2, 4, 8, 16, 32\}$ uma base de $\mathbb{F}_{2^6}$ sobre $\mathbb{F}_2$.

[1] Nesse capítulo, exemplos que utilizam corpos de extensão usarão a representação "inteira" de um elemento, ou seja, polinômios em α são representados por números inteiros em ordem crescente de potência. Por exemplo, o elemento $\alpha^3 + \alpha + 1$ será representado por 11.

Suponha que $a = \begin{bmatrix} 8 & 50 & 37 & 56 & 7 \end{bmatrix}^T$ então

$$\varphi(a) = \begin{bmatrix} \varphi(8) \\ \varphi(50) \\ \varphi(37) \\ \varphi(56) \\ \varphi(7) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix},$$

note que, por exemplo, $37 = 1 \cdot 2^0 + 1 \cdot 2^2 + 1 \cdot 2^5$ e $7 = 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2$. □

3.1.2 Bases Normais

Seja $\mathcal{A}$ uma base de $\mathbb{F}_{q^m}$ sobre $\mathbb{F}_q$ e seja $\alpha \in \mathbb{F}_{q^m}$ um elemento primitivo. Se $\mathcal{A} = \{\alpha^0, \alpha^1, \ldots, \alpha^{m-1}\}$ então dizemos que $\mathcal{A}$ é uma *base polinomial* (ou *base padrão*). Se $\mathcal{A} = \{\alpha^{[0]}, \alpha^{[1]}, \ldots, \alpha^{[m-1]}\}$, em que $[i] = q^i$ (chamada de notação de Frobenius), então $\mathcal{A}$ é uma *base normal* e α é chamado de *elemento normal* [6].

Geralmente representamos um elemento de um corpo de extensão como sendo um vetor sobre o corpo base. Nesse caso, algumas operações, considerando o uso de bases normais, ficam menos complexas [20].

Para um vetor $\underline{a} = \begin{bmatrix} a_0 & a_1 & \cdots & a_{m-1} \end{bmatrix} \in \mathbb{F}_q^{1\times m}$, denotaremos por $\underline{a}^{\leftarrow i}$ o deslocamento cíclico para esquerda de i posições, ou seja, $\underline{a}^{\leftarrow i} = \begin{bmatrix} a_i & \cdots & a_{m-1} & a_0 & \cdots & a_{i-1} \end{bmatrix}$. Similarmente, $\underline{a}^{\rightarrow i} = \underline{a}^{\leftarrow m-i}$, isto é, denota um deslocamento cíclico para a direita. Desse modo, observamos que para um $a \in \mathbb{F}_{q^m}$, $a^{[i]} = \varphi(a)^{\rightarrow i} \cdot [[\alpha]]$, i.e, elevar algum elemento a uma potência de Frobenius corresponde a deslocamentos cíclicos. O custo computacional necessário para elevar um elemento a uma potência de Frobenius é desprezível quando se utiliza uma base normal.

Já para a multiplicação, devemos utilizar a *tabela de multiplicação* para bases normais. Seja $T \in \mathbb{F}_q^{n\times m}$ a matriz que define a tabela de multiplicação, tal que $\alpha\alpha^{[i]} = \sum_{j=0}^{m-1} T_{i,j}\alpha^{[j]}$, $i = 0, 1, \ldots, m-1$. O número de entradas não-nulas de T é chamado de *complexidade* da base normal [6]. A multiplicação segue da seguinte maneira. Sejam

$a, b \in \mathbb{F}_{q^m}$ então

$$\underline{ab} = \sum_{i=0}^{m-1} b_i \left(\underline{a}^{\leftarrow i} T\right)^{\rightarrow i}.$$

O custo computacional para realizar multiplicações em bases normais é da ordem do número de entradas não-nulas de T, portanto realizar multiplicações em bases normais só é eficiente se a matriz T for esparsa.

Como mostrado em [6], a complexidade de uma base normal tem como limitante inferior $2m - 1$. Bases normais que atingem esse limitante são chamadas de *bases ótimas*. É possível construir bases normais de baixa complexidade (não necessariamente ótimas) de maneira que as entradas da matriz T pertençam a $\mathbb{F}_p$, onde p é a característica de q (e portanto, p é primo). Nesse caso, as entradas das matrizes de conversão entre a base normal e a base polinômial também pertecem a $\mathbb{F}_p$ [20].

3.1.3 Polinômios Linearizados

Um *polinômio linearizado* (também chamado de *q-polinômio*) sobre $\mathbb{F}_{q^m}$ é um polinômio da forma

$$f(x) = \sum_{i=0}^{t} f_i x^{[i]},$$

em que $f_i \in \mathbb{F}_{q^m}$ [20, 27]. Se $f_t \neq 0$ então dizemos que o q-grau de $f(x)$ é t.

É fácil ver que a soma de dois polinômios linearizados também é um polinômio linearizado. Todavia a multiplicação entre dois polinômios linearizados não necessariamente será um polinômio linearizado. Utilizamos então uma *multiplicação simbólica* ($\otimes$), a qual é definida como $f(x) \otimes g(x) = f(g(x))$. Claramente a multiplicação simbólica é *não-comutativa*.

Sejam n e k os q-graus de $f(x)$ e $g(x)$ respectivamente. E seja t o q-grau de $P(x)$ em que $P(x) = f(x) \otimes g(x)$, então $t = n + k$. Os coeficientes de $P(x)$ podem ser calculados da seguinte maneira

$$P_\ell = \sum_{i=\max\{0,\ell-k\}}^{\min\{\ell,n\}} f_i g_{\ell-i}^{[i]} = \sum_{j=\max\{0,\ell-n\}}^{\min\{\ell,k\}} f_{\ell-j} g_j^{[\ell-j]},\ \ell = 0, \ldots, t.$$

Seja $f(x)$ um polinômio linearizado com q-grau igual a t. Definimos o polinômio linearizado *reverso* (usualmente chamado de *q-reverso*) de $f(x)$ como sendo o polinômio $\tilde{f}(x) = \sum_{i=0}^{t} \tilde{f}_i x^{[i]}$, em que $\tilde{f}_i = f_{t-i}^{[i-t]}$, $i = 0, \ldots, t$.

Para um conjunto $\mathcal{S} \subseteq \mathbb{F}_{q^m}$ chamamos de *q-polinômio mínimo de* $\mathcal{S}$ um polinômio linearizado, $M_{\mathcal{S}}(x) = \sum_{i=0}^{t} M_i x^{[i]}$ com o menor q-grau e $M_0 = 1$, cujo o espaço gerado pelas raízes de $M_{\mathcal{S}}(x)$ contenha o conjunto $\mathcal{S}$. Seja $\{s_1, \ldots, s_t\}$ uma base para espaço gerado por $\mathcal{S}$, podemos encontrar $M_{\mathcal{S}}$ de forma recursiva. Para isso computamos

$$M_{\{s_1\}} = x - \frac{s_1}{s_1^{[1]}} x^{[1]},$$

e a seguir, para $i = 2, \ldots, t$ dizemos que $z_i = M_{\{s_1,\ldots,s_{i-1}\}}(s_i)$,

$$M_{\{z_i\}}(x) = x - \frac{z_i}{z_i^{[1]}} x^{[1]}$$

e então $M_{\{s_1,\ldots,s_i\}}(x) = M_{\{z_i\}}(x) \otimes M_{\{s_1,\ldots,s_{i-1}\}}(x)$.

3.2 Códigos com Métrica de Posto

Seja $v \in \mathbb{F}_{q^m}^n$, dizemos então que o posto de v é igual ao posto de $\varphi(v)$, isto é $\operatorname{posto} v \triangleq \operatorname{posto} \varphi(v)$, em que o posto de uma matriz é definido como sendo o máximo número de linhas (ou colunas) linearmente independentes.

O posto de um vetor especifica uma norma em $\mathbb{F}_{q^m}^n$. De fato, $\operatorname{posto} v \geq 0$ para todo $v \in \mathbb{F}_{q^m}^n$ e $\operatorname{posto} v = 0$ se, e somente se v for o vetor nulo. A desigualdade triangular vem das propriedades de matrizes, e portanto, para quaisquer $v, u \in \mathbb{F}_{q^m}^n$ temos que $\operatorname{posto}(v+u) \leq \operatorname{posto} v + \operatorname{posto} u$. E finalmente, para $a \in \mathbb{F}_{q^m}$ definimos $|a| = 0$ se $a = 0$ e $|a| = 1$, caso contrário, então temos que $\operatorname{posto} av = |a| \operatorname{posto} v$, uma vez que, multiplicar um vetor por um escalar não-nulo, não altera a relação de linearidade entre os elementos.

Com essa norma podemos especificar uma métrica, a qual chamaremos de *distância de posto*, em que, para $v, u \in \mathbb{F}_{q^m}^n$ temos

$$\mathrm{d_R}(v, u) \triangleq \operatorname{posto}(u - v), \tag{3.1}$$

note que se $V, U \in \mathbb{F}_q^{n\times m}$ então a distância de posto é definida da mesma maneira, isto é, $\mathrm{d_R}(V,U) = \mathrm{posto}(U-V)$.

Um código $\mathcal{C}(n,k)$ é aquele que mapeia um vetor $u \in \mathbb{F}_{q^m}^k$ em um vetor $c \in \mathbb{F}_{q^m}^n$, $n \geq k$. Ou seja $\mathcal{C} \subseteq \mathbb{F}_{q^m}^n$ é um conjunto não-vazio de vetores que utilizam a métrica de distância de posto. Dizemos que a *mínima distância de posto* de um código $\mathcal{C}$ é

$$\mathrm{d_R}(\mathcal{C}) = \min_{\substack{c,c'\in\mathcal{C}\\ c\neq c'}} \mathrm{d_R}(c,c'). \tag{3.2}$$

Seja $\mathcal{C}(n,k)$ um código linear sobre $\mathbb{F}_{q^m}$, em que $m \geq n$. Percebe-se que $\mathrm{d_R}(\mathcal{C}) \leq \mathrm{d_H}(\mathcal{C})$, em que $\mathrm{d_H}$ denota a *distância de Hamming* [4]. Sabemos que códigos que utilizam a distância de Hamming como métrica respeitam o *limitante de Singleton* [14]. Portanto, para os códigos com métrica de distância de posto, o limitante de Singleton é válido, i.e:

$$\mathrm{d_R}(\mathcal{C}) \leq n - k + 1. \tag{3.3}$$

Códigos para os quais a igualdade é válida são chamados de códigos de *máxima distância de posto (MRD)* (do inglês, *maximum rank-distance*).

Um decodificador de *mínima distância* de um código $\mathcal{C}$ é aquele que pega uma palavra $r \in \mathbb{F}_{q^m}^n$ e leva na palavra-código $\hat{c} \in \mathcal{C}$ mais próxima (no conceito de distância de posto) [20]. Ou seja

$$\hat{c} = \operatorname*{argmin}_{c\in\mathcal{C}} \mathrm{d_R}(c,r). \tag{3.4}$$

Se $\mathrm{d_R}(c,r) < \mathrm{d_R}(\mathcal{C})/2$, então é garantido que o decodificador retornará $\hat{c} = c$.

3.3 Códigos de Gabidulin

Uma importante família de códigos MRD são os *códigos de Gabidulin*. Esse códigos existem para quaisquer q e $n \leq m$ [4]. Para que esses códigos sejam eficientes é necessário que sua complexidade seja reduzida, então iremos fazer uso de *bases normais*. Ou seja, iremos considerar que $\mathcal{A} = \{\alpha^{[0]}, \alpha^{[1]}, \ldots, \alpha^{[m-1]}\}$ é base de $\mathbb{F}_{q^m}$ sobre $\mathbb{F}_q$, em que α é um elemento primitivo de $\mathbb{F}_{q^m}$ (e também um elemento normal).

Seja $\mathcal{C} \subseteq \mathbb{F}_{q^m}$ um código de Gabidulin (n, k), onde $n \leq m$. Seja $H \in \mathbb{F}_{q^m}^{(n-k)\times n}$ a *matriz de paridade* e seja $G \in \mathbb{F}_{q^m}^{k\times n}$ a *matriz geradora* do código. Então

$$\mathcal{C} = \{c \in \mathbb{F}_{q^m}^n : Hc = 0\} = \{G^T u,\ u \in \mathbb{F}_{q^m}^k\}.$$

Por ser um código MRD, então a distância mínima é $d = \mathrm{d_R}(\mathcal{C}) = n - k + 1$.

Sejam $h_0, h_1, \ldots, h_{n-1} \in \mathbb{F}_{q^m}$ linearmente independentes então a matriz H será da forma

$$H = \begin{bmatrix} h_0^{[0]} & h_1^{[0]} & \cdots & h_{n-1}^{[0]} \\ h_0^{[1]} & h_1^{[1]} & \cdots & h_{n-1}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ h_0^{[n-k-1]} & h_1^{[n-k-1]} & \cdots & h_{n-1}^{[n-k-1]} \end{bmatrix}. \tag{3.5}$$

É interessante fazer que $h_i = \alpha^{[i]}$, $i = 0, 1, \ldots, n-1$, quando estamos usando bases normais.

A matriz geradora pode ser qualquer matriz da forma

$$G = \begin{bmatrix} g_0^{[0]} & g_1^{[0]} & \cdots & g_{n-1}^{[0]} \\ g_0^{[1]} & g_1^{[1]} & \cdots & g_{n-1}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_0^{[k-1]} & g_1^{[k-1]} & \cdots & g_{n-1}^{[k-1]} \end{bmatrix}, \tag{3.6}$$

tal que $HG^T = 0$ e $g_0, g_1, \ldots, g_{n-1} \in \mathbb{F}_{q^m}$ sejam linearmente independentes.

3.4 Decodificação

Seja $\mathcal{C} \subseteq \mathbb{F}_{q^m}^n$ um código de Gabidulin. Seja $\mathcal{A} = \{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$ uma base de $\mathbb{F}_{q^m}$ sobre $\mathbb{F}_q$ e seja $h = \begin{bmatrix} h_0 & h_1 & \cdots & h_{n-1} \end{bmatrix}^T$ um vetor coluna correspondente à primeira linha da matriz de paridade de $\mathcal{C}$. Suponha que $r = \begin{bmatrix} r_0 & r_1 & \cdots & r_{n-1} \end{bmatrix}^T \in \mathbb{F}_{q^m}^n$ seja a palavra recebida. Sabemos que $r = c + e$, em que $c \in \mathcal{C}$ é a palavra-código enviada e

$e \in \mathbb{F}_{q^m}^n$ é o erro introduzido pelo canal tal que posto $e \triangleq \tau$. Temos que encontrar um $e \in r - \mathcal{C}$ que satisfaça $2\tau < d$.

Podemos expandir e como sendo:

$$e = \begin{bmatrix} L_1 & L_2 & \cdots & L_\tau \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_\tau \end{bmatrix} = \sum_{i=1}^{\tau} L_i V_i \tag{3.7}$$

e assim dividimos o problema em encontrar as *localizações de erro* $L_1, L_2, \ldots, L_\tau \in \mathbb{F}_q^n$ e encontrar os *valores de erro* $V_1, V_2, \ldots, V_\tau \in \mathbb{F}_{q^m}$. Note que isso é semelhante aos códigos não-binários que utilizam métrica de Hamming. É preciso deixar claro que existem mais de uma combinação de L_i e V_i, $i = 1, \ldots, \tau$, que dê o mesmo vetor de erro e, isso quer dizer que a expansão de e não é única [20].

Podemos utilizar dois métodos equivalentes. O primeiro introduzido em [4] consiste em encontrar o *polinômio de varredura de erro (ESP)* (do inglês, *error span polynomial*), que indica os valores do erro, e então encontrar a localização deles. O segundo introduzido em [22] consiste em encontrar o *polinômio de localização de erro (ELP)* (do inglês, *error locator polynomial*), que nos dá a ideia de localização do erros (na realidade, através desse polinômio encontramos os *localizadores de erro* que nada mais são do que as localizações de erro multiplicadas pela matriz de paridade, e através desses, encontramos as localizações) e então encontramos os valores de erro. Em ambos os métodos utilizamos o algoritmo de Berlekamp-Massey modificado e o algoritmo de Gabidulin. Ambos os algoritmos serão apresentados na próxima subseção.

A seguir descrevemos os métodos ESP e ELP. Para ambos os casos considere que $r = \begin{bmatrix} r_0 & r_1 & \cdots r_{n-1} \end{bmatrix}^T \in \mathbb{F}_{q^m}^n$ seja a palavra recebida.

3.4.1 Algoritmo de Decodificação ESP

Passo 1 Calcular a síndrome:

$$S_\ell = \sum_{i=0}^{n-1} h_i^{[\ell]} r_i, \ \ell = 0, \ldots, d-2. \tag{3.8}$$

Passo 2 Encontrar os coeficientes do polinômio ESP $\Gamma(x) = \sum_{i=0}^{\tau} \Gamma_i x^{[i]}$ (usando o algoritmo de Berlekamp-Massey):

$$\sum_{i=0}^{\tau} \Gamma_i S_{\ell-i}^{[i]} = 0, \ \ell = \tau, \ldots, d-2. \tag{3.9}$$

Passo 3 Encontrar uma base $(V_1, \ldots, V_\tau)$, $V_i \in \mathbb{F}_{q^m}$, $i = 1, \ldots, \tau$, para o espaço gerado pelas raízes de $\Gamma(x)$:

$$\begin{bmatrix} \varphi(V_1) \\ \vdots \\ \varphi(V_\tau) \end{bmatrix} \begin{bmatrix} \varphi\left(\Gamma(\alpha_0)\right) \\ \vdots \\ \varphi\left(\Gamma(\alpha_{m-1})\right) \end{bmatrix} = 0. \tag{3.10}$$

Note que, $\alpha_i \in \mathbb{F}_{q^m}$, $i = 0, 1, \ldots, m-1$ são os elementos da base utilizada.

Passo 4 Calcular os localizadores do erro $(X_1, \ldots, X_\tau)$, $X_i \in \mathbb{F}_{q^m}$, $i = 1, \ldots, \tau$ (usando o algoritmo de Gabidulin):

$$S_\ell = \sum_{j=1}^{\tau} X_j^{[\ell]} V_j, \ \ell = 0, \ldots, \tau. \tag{3.11}$$

Passo 5 Calcular as localizações do erro $(L_1, \ldots, L_\tau)$, $L_i \in \mathbb{F}_q^n$, $i = 1, \ldots, \tau$:

$$L_j^T = \varphi(X_j)\varphi(h)^\ddagger, \tag{3.12}$$

em que $\varphi(h)^\ddagger$ é a inversa à direita de $\varphi(h)$.

Pasoo 6 Encontrar o erro:

$$e = \sum_{j=1}^{\tau} L_j V_j \in \mathbb{F}_{q^m}^n. \tag{3.13}$$

Passo 7 Estimar a palavra-código enviada:

$$\hat{c} = r - e \in \mathbb{F}_{q^m}^n. \tag{3.14}$$

3.4.2 Algoritmo de Decodicação ELP

Passo 1 Calcular a síndrome "reversa":

$$\tilde{S}_\ell = \sum_{i=0}^{n-1} h_i r_i^{[\ell-d+2]},\ \ell = 0, \ldots, d-2. \tag{3.15}$$

Passo 2 Encontrar os coeficientes do polinômio ELP $\Lambda(x) = \sum_{i=0}^{\tau} \Lambda_i x^{[i]}$ (usando o algoritmo de Berlekamp-Massey):

$$\sum_{i=0}^{\tau} \Lambda_i \tilde{S}_{\ell-i}^{[i]} = 0,\ \ell = \tau, \ldots, d-2. \tag{3.16}$$

Passo 3 Encontrar uma base $(X_1, \ldots, X_\tau)$, $X_i \in \mathbb{F}_{q^m}$, $i = 1, \ldots, \tau$, para o espaço gerado pelas raízes de $\Lambda(x)$:

$$\begin{bmatrix} \varphi(X_1) \\ \vdots \\ \varphi X_\tau \end{bmatrix} \begin{bmatrix} \varphi(\Lambda(\alpha_0)) \\ \vdots \\ \varphi(\Lambda(\alpha_{m-1})) \end{bmatrix} = 0. \tag{3.17}$$

Passo 4 Encontrar os valores do erros $(V_1, \ldots, V_\tau)$, $V_i \in \mathbb{F}_{q^m}$, $i = 1, \ldots, \tau$: (usando algoritmo de Gabidulin):

$$\tilde{S}_\ell = \sum_{j=1} V_j^{[\ell-d+2]} X_j,\ \ell = 0, \ldots, \tau \tag{3.18}$$

Passo 5 Calcular as localizações do erro $(L_1, \ldots, L_\tau)$, $L_1 \in \mathbb{F}_q^n$, $i = 1, \ldots, \tau$:

$$L_j^T = \varphi(X_j)\varphi(h)^\ddagger, \tag{3.19}$$

em que, $\varphi(h)^\ddagger$ é a inversa à direita de $\varphi(h)$.

Passo 6 Encontrar o erro:

$$e = \sum_{j=1}^{\tau} L_j V_j \in \mathbb{F}_{q^m}^n. \tag{3.20}$$

Passo 7 Estimar a palavra-código enviada:

$$\hat{c} = r - e \in \mathbb{F}_{q^m}^n. \tag{3.21}$$

É importante ressaltar que em (3.18) ao utilizar o algoritmo de Gabidulin não obtemos $V_j,\ j = 1, \ldots, \tau$, diretamente, pois o algoritmo assume que a saída estaria sendo elevada à potência de Frobenius ℓ, portanto, seja $z_1, \ldots, z_\tau$ a saída do algoritmo de Gabidulin, explicado na seção 3.4.4, então $V_j = z_j^{[d-2]},\ j = 1, \ldots, \tau$.

3.4.3 Algoritmo de Berlekamp-Massey modificado

Seja $\Delta(x) = \sum_{i=0}^{k} \Delta_i x^{[i]}$ um polinômio linearizado. O algoritmo de Berlekamp-Massey modificado [16, 17, 18] é usado para encontrar um polinômio linearizado $\sigma(x) = \sum_{i=0}^{\nu} \sigma_i x^{[i]}$, de menor q-grau, tal que $\sigma_0 = 1$, que satisfaça

$$\sum_{i=0}^{\nu} \sigma_i \Delta_{\ell-i}^{[i]} = 0,\ \ell = \nu, \ldots, k. \tag{3.22}$$

O algoritmo de Berlekamp-Massey modificado é descrito no Algoritmo 1.

Algoritmo 1 Berlekamp-Massey modificado

Input: $\Delta(x) = \sum_{i=0}^{k} \Delta_i x^{[i]}$

$\ell \leftarrow 0$, $\sigma(x) \leftarrow x$ # Inicialização
$\sigma'(x) \leftarrow x$, $n' \leftarrow 0$, $d' \leftarrow 1$ # Variáveis de auxílio
for $n \leftarrow 0, \ldots, k$ **do**
    $d \leftarrow \sum_{j=0}^{\ell} \sigma_j \Delta_{n-j}^{[j]}$ # Cálculo da discrepância
    if $d \neq 0$ **then**
        if $2\ell \geq n$ **then**
            $\sigma(x) \leftarrow \sigma(x) - d \left(\frac{x}{d'}\right)^{[n-n']} \otimes \sigma'(x)$
        else
            $\tilde{\sigma}(x) \leftarrow \sigma(x)$ # Variável temporária
            $\sigma(x) \leftarrow \sigma(x) - d \left(\frac{x}{d'}\right)^{[n-n']} \otimes \sigma'(x)$
            $\ell \leftarrow n - \ell$
            $\sigma'(x) \leftarrow \tilde{\sigma}(x)$, $n' \leftarrow n$, $d' \leftarrow d$
        end if
    end if
end for

Output: $\sigma(x)$

3.4.4 Algoritmo de Gabidulin

Seja $y = \begin{bmatrix} y_1 & y_2 & \cdots & y_\tau \end{bmatrix}^T \in \mathbb{F}_{q^m}^\tau$ e $\omega = \begin{bmatrix} \omega_1 & \omega_2 & \cdots & \omega_\tau \end{bmatrix}^T \in \mathbb{F}_{q^m}^\tau$, o *algoritmo de Gabidulin* é usado para encontrar um vetor $z = \begin{bmatrix} z_1 & z_2 & \cdots & z_\tau \end{bmatrix}^T \in \mathbb{F}_{q^m}^\tau$ tal que

$$y_\ell = \sum_{i=1}^{\tau} z_i^{[\ell]} \omega_i, \ \ell = 1, \ldots, \tau.$$

Para resolver essa equação poderiamos utilizar eliminação Gaussiana. Mas como mostrado por [5] utilizar o algoritmo de Gabidulin requer menor complexidade computacional.

O algoritmo requer como entrada os vetores y e ω e a saída será o vetor z. Note que a saída é aquela com a *potência de Frobenius*. O algoritmo de Gabidulin [4] é descrito no Algoritmo 2.

Algoritmo 2 Algoritmo de Gabidulin

Input: $y = \begin{bmatrix} y_1 & y_2 & \cdots & y_\tau \end{bmatrix}$, $\omega = \begin{bmatrix} \omega_1 & \omega_2 & \cdots & \omega_\tau \end{bmatrix}$

for $j \leftarrow 0, 1, \cdots, \tau - 1$ **do**
  $Q_{0,j} \leftarrow y_j$
  $A_{0,j} \leftarrow \omega_j$
end for
for $i \leftarrow 1, 2, \cdots, \tau - 1$ **do**
  for $j \leftarrow i, i+1, \cdots, \tau - 1$ **do**
    $A_{i,j} \leftarrow A_{i-1,j} - \left(\frac{A_{i-1,j}}{A_{i-1,i-1}} \right)^{[-1]} A_{i-1,i-1}$
  end for
  for $j \leftarrow 0, 1, \cdots, \tau - 1 - i$ **do**
    $Q_{i,j} \leftarrow Q_{i-1,j} - \left(\frac{Q_{i-1,j+1}}{A_{i-1,i-1}} \right)^{[-1]} A_{i-1,i-1}$
  end for
end for
$z_\tau \leftarrow \frac{Q_{\tau-1,0}}{A_{\tau-1,\tau-1}}$
for $i \leftarrow 1, 2, \cdots, \tau - 1$ **do**
  $z_{\tau-i} \leftarrow \frac{Q_{\tau-1-i,0} - \sum_{j=\tau-i}^{\tau-1} A_{\tau-1-i,j} z_{j+1}}{A_{\tau-1-i,\tau-1-i}}$
end for

Output: $z = \begin{bmatrix} z_1 & z_2 & \cdots & z_\tau \end{bmatrix}$

Exemplo. Seja $q = 2$, $m = 8$ e $n = 7$. Seja também $p(x) = x^8 + x^4 + x^3 + x^2 + 1$ o polinômio primitivo de $\mathbb{F}_{2^8}$. Considere que base polinomial, isto é, $\mathcal{A} = \{1, 2, 4, \ldots, 128\}$. E seja $\mathcal{C} \subseteq \mathbb{F}_{2^8}^7$ um código matricial com distância mínima $d = \mathrm{d_R}(\mathcal{C}) = 5$. Suponha que $c = \begin{bmatrix} 95 & 88 & 241 & 102 & 157 & 108 & 230 \end{bmatrix}^T$ seja a palavra-código enviada e $r = \begin{bmatrix} 190 & 29 & 241 & 135 & 57 & 200 & 7 \end{bmatrix}^T$ a palavra recebida.

	Algoritmo ESP
1	$S = \begin{bmatrix} 121 & 142 & 140 & 124 \end{bmatrix}$
2	$\Gamma(x) = x^{[0]} + 69x^{[1]} + 10x^{[2]}$
3	$V = \begin{bmatrix} 225 & 164 \end{bmatrix}^T$
4	$X = \begin{bmatrix} 75 & 50 \end{bmatrix}$
5	$L = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}^T$
6	$e = \begin{bmatrix} 225 & 69 & 0 & 225 & 164 & 164 & 225 \end{bmatrix}^T$
7	$\hat{c} = \begin{bmatrix} 95 & 88 & 241 & 102 & 157 & 108 & 230 \end{bmatrix}^T$

	Algoritmo ELP
1	$\tilde{S} = \begin{bmatrix} 103 & 30 & 204 & 121 \end{bmatrix}$
2	$\Lambda(x) = x^{[0]} + 6x^{[1]} + 150x^{[2]}$
3	$X = \begin{bmatrix} 121 & 50 \end{bmatrix}$
4	$V = \begin{bmatrix} 225 & 69 \end{bmatrix}^T$
5	$L = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}^T$
6	$e = \begin{bmatrix} 225 & 69 & 0 & 225 & 164 & 164 & 225 \end{bmatrix}^T$
7	$\hat{c} = \begin{bmatrix} 95 & 88 & 241 & 102 & 157 & 108 & 230 \end{bmatrix}^T$

Note que, os valores de $V_1, \ldots, V_\tau$ e $L_1, \ldots, L_\tau$ são diferentes em cada método. Todavia, o vetor erro é igual para ambos. Ou seja, conforme citado anteriormente, existem mais de uma maneira de se decompor o vetor de erro e.

□

3.5 Decodificação Generalizada

Como dito anteriormente, podemos expandir o vetor de erro, como mostrado em (3.7), que será reproduzido a seguir por conveniência

$$e = \sum_{i=1}^{\tau} L_i V_i, \tag{3.23}$$

em que L_i é chamado de localização de erro e V_i é chamado de valor de erro, $i = 1, \ldots, \tau$.

Muitas vezes o decodificador conhece alguns valores do erro ou então algumas localizações do erro. Utilizando esse fato conseguimos aumentar a capacidade de decodificação. Chamamos de:

- *Apagamento* - quando conhecemos a localização, mas não o valor;
- *Desvio* - quando conhecemos o valor, mas não a localização;
- *Erro* - quando nem a localização, nem o valor são conhecidos.

Seja $\hat{L} \in \mathbb{F}_q^{n\times\mu}$ a matriz em que cada coluna representa um *apagamento*, tal que $\text{posto}\,\hat{L} = \mu$. Seja $\hat{V} \in \mathbb{F}_{q^m}^{\delta}$ o vetor em que cada elemento representa um *desvio*, tal que $\text{posto}\,\hat{V} = \delta$. Iremos impor que $L_j = \hat{L}_j$, $j = 1, \ldots, \mu$ e $V_{\mu+j} = \hat{V}_j$, $j = 1, \ldots, \delta$. Seja $\epsilon = \tau - \mu - \delta$, nosso desafio é encontrar o menor valor de ϵ (note que, se fosse para minimizar τ, bastaria escolher $\epsilon = -\mu - \delta$ de modo que $\tau = 0$) de tal forma que o vetor de erro possa ser escrito como

$$e = \hat{L}V_a + L_d\hat{V} + L_e V_e,$$

em que $V_a \in \mathbb{F}_{q^m}^{\mu}$, $L_d \in \mathbb{F}_q^{n\times\delta}$, $L_e \in \mathbb{F}_q^{n\times\epsilon}$ e $V_e \in \mathbb{F}_{q^m}^{\epsilon}$.

Na decodificação generalizada há um incremento na capacidade de correção do código. O código $\mathcal{C}$ com distância mínima d consegue recuperar c a partir de r, de forma única, se e somente se $2\tau < d + \mu + \delta$ ou, de modo equivalente, $2\epsilon + \delta + \mu < d$[20].

Os algoritmos ESP e ELP ainda podem ser usados nesse caso, todavia o passo 2 deverá ser alterado. Suponha que $(r, \hat{L}, \hat{V}) \in \mathbb{F}_{q^m}^{n} \times \mathbb{F}_q^{n\times\mu} \times \mathbb{F}_{q^m}^{\delta}$ seja a tupla recebida e seja h o vetor correspondente a primeira linha da matriz de paridade H.

Nos algoritmos a seguir, o subscrito U indica informação obtida a partir dos apagamentos, o subscrito D indica informação obtida a partir

dos desvios e o subscrito F indica informação desconhecida (ou seja, referente aos erros). Polinômios com $\sim$ (til) são os chamados polinômios $q-$reversos.

Para ambos os métodos devemos calcular

Passo 2.a

$$\hat{X}_j = \hat{L}_j^T h,\ j = 1, \ldots, \mu \tag{3.24}$$

Passo 2.b

$$\Lambda_U(x) = M_{\{\hat{X}_1, \ldots, \hat{X}_\mu\}}(x). \tag{3.25}$$

Passo 2.c

$$\Gamma_D(x) = M_{\{\hat{V}_1, \ldots, \hat{V}_\mu\}}(x). \tag{3.26}$$

No algoritmo generalizado iremos fazer uso dos polinômios q-reversos explicados anteriormente. Considere que $\epsilon = \tau - \mu - \delta$.

3.5.1 Algoritmo de Decodifição Generalizado ESP

Passo 2.d Definir:

$$S(x) = \sum_{i=0}^{d-2} S_i x^{[i]}, \tag{3.27}$$

em que S_i é calculado em (3.8).

Passo 2.e Calcular:

$$S_{DU}(x) = \Gamma_D(x) \otimes S(x) \otimes \tilde{\Lambda}_D(x). \tag{3.28}$$

Passo 2.f Utilizando o algoritmo de Berlekamp-Massey, encontrar o polinômio $\Gamma_F(x)$ tal que:

$$\sum_{i=0}^{\epsilon} \Gamma_{F,i} S_{DU,\mu+\delta+\ell-i}^{[i]} = 0,\ \ell = \epsilon, \ldots, d-2-\mu-\delta. \tag{3.29}$$

Passo 2.g Calcular:

$$S_{FD}(x) = \Gamma_F(x) \otimes \Gamma_D(x) \otimes S(x). \tag{3.30}$$

Passo 2.h Encontrar $\zeta_1, \ldots, \zeta_\mu$, usando o algoritmo de Gabidulin, tal que:

$$S_{FD,\ell} = \sum_{j=1}^{\mu} \hat{X}_j^{[\ell]} \zeta_j, \ \ell = \epsilon + \delta, \ldots, d-2 \tag{3.31}$$

Passo 2.i Calcular:

$$\Gamma_U(x) = M_{\{\zeta_1,\ldots,\zeta_\mu\}}(x). \tag{3.32}$$

Passo 2.j Calcular:

$$\Gamma(x) = \Gamma_U(x) \otimes \Gamma_F(x) \otimes \Gamma_D(x). \tag{3.33}$$

Usar o algoritmo de decodificação ESP a partir de (3.10).

3.5.2 Algoritmo de Decodifição Generalizado ELP

Passo 2.d Definir:

$$\tilde{S}(x) = \sum_{i=0}^{d-2} \tilde{S}_i x^{[i]}, \tag{3.34}$$

em que $\tilde{S}_i$ é calculado em (3.15).

Passo 2.e Calcular:

$$S_{UD}(x) = \Lambda_U(x) \otimes \tilde{S}(x) \otimes \tilde{\Gamma}(x^{[d-2]})^{[-d+2]}. \tag{3.35}$$

Passo 2.f Utilizando o algoritmo de Berlekamp-Massey, encontrar o polinômio $\Lambda_F(x)$, tal que:

$$\sum_{i=0}^{\epsilon} \Lambda_{F,i} S_{UD,\mu+\delta+\ell-i}^{[i]} = 0, \ \ell = \epsilon, \ldots, d-2-\mu-\delta. \tag{3.36}$$

Passo 2.g Calcular:

$$S_{FU}(x) = \Lambda_F(x) \otimes \Lambda_U(x) \otimes \tilde{S}(x). \tag{3.37}$$

Passo 2.h Encontrar $\xi_1, \ldots, \xi_\delta$, usando o algoritmo de Gabidulin, tal

que:

$$S_{FU,\ell} = \sum_{j=1}^{\delta} \hat{V}_j^{[\ell-d+2]} \xi_j, \ \ell = \epsilon + \mu, \ldots, d-2 \tag{3.38}$$

Passo 2.i Calcular:

$$\Lambda_D(x) = M_{\{\xi_1,\ldots,\xi_\delta\}}(x). \tag{3.39}$$

Passo 2.j Calcular:

$$\Lambda(x) = \Lambda_D(x) \otimes \Lambda_F(x) \otimes \Lambda_U(x). \tag{3.40}$$

Usar o algoritmo de decodificação ELP a partir de (3.17).

Note que no passo aonde é utilizado o algoritmo de Gabidulin (em ambos os métodos) será necessária uma manipulação. Isto porque o algoritmo de Gabidulin assume que a saída está sendo elevada a uma potência de Frobenius. Para resolver isso, podemos utilizar o q-reverso.

Exemplo. Seja $q = 2$, $m = 8$ e $n = 7$ e seja $\mathcal{A} = \{1, 2, 4, \ldots, 128\}$ uma base de $\mathbb{F}_{2^8}$ sobre $\mathbb{F}_2$. Considere um código de Gabidulin $\mathcal{C}$ com distância mínima $d = 5$. Seja $c = \begin{bmatrix} 251 & 162 & 184 & 5 & 252 & 248 & 5 \end{bmatrix}^T \in \mathcal{C}$ a palavra-código enviada e suponha que o decodificador receba:

$$\begin{aligned} r &= \begin{bmatrix} 131 & 254 & 190 & 5 & 252 & 220 & 95 \end{bmatrix}^T \\ \hat{L} &= \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T \\ \hat{V} &= \begin{bmatrix} 34 \end{bmatrix} \end{aligned}$$

Por facilidade iremos apenas demonstrar utilizando o algoritmo ESP generalizado. Note que, se utilizarmos o algoritmo ELP generalizado, os resultados intermediários serão diferentes, todavia o vetor de erro encontrado no final será o mesmo. Percebemos que $\mu = \delta = 1$. A

decodificação segue como:

$$\begin{aligned}
\hat{X} &= \left[67\right] \\
\Lambda_U(x) &= x^{[0]} + 213x^{[1]} \\
\tilde{\Lambda}_U(x) &= 83x^{[0]} + x^{[1]} \\
\Gamma_D(x) &= x^{[0]} + 57x^{[1]} \\
S(x) &= 47x^{[0]} + 94x^{[1]} + x^{[2]} + 57x^{[3]} \\
S_{DU}(x) &= 180x^{[0]} + 141x^{[1]} + 133x^{[2]} + 52x^{[3]} + 165x^{[4]} + 166x^{[5]}
\end{aligned}$$

Note que, para utilizar o algoritmo de Berlekamp-Massey não precisamos utilizar todos os coeficientes de $S_{DU}(x)$, apenas $S_{DU,\ell}$, $\ell = \mu + \delta, \ldots, d-2$. Considerando isso, podemos prosseguir com a decodificação.

$$\begin{aligned}
\Gamma_F(x) &= x^{[0]} + 26x^{[1]} \\
S_{FD}(x) &= 47x^{[0]} + 105x^{[1]} + 150x^{[2]} + 222x^{[3]} + 166x^{[4]} + 58x^{[5]}
\end{aligned}$$

Note que o q-grau de $\Gamma_F(x)$ é igual a $\epsilon = 1$. Para encontrar ζ devemos resolver a equação (3.31) usando o algoritmo de Gabidulin. Todavia, como explicado anteriormente, o algoritmo assume que a saída é aquela com a potência de Frobenius. Por isso, devemos calcular o q-reverso da equação (3.31) antes de usar o algoritmo de Gabidulin. Feito isso, obtemos os valores a seguir.

$$\begin{aligned}
\zeta &= \left[119\right] \\
\Gamma_U(x) &= x^{[0]} + 121x^{[1]} \\
\Gamma(x) &= x^{[0]} + 90x^{[1]} + 189x^{[2]} + 45x^{[3]}
\end{aligned}$$

Agora prosseguimos igual ao caso em que não há apagamentos nem

desvios.

$$V = \begin{bmatrix} 34 & 36 & 120 \end{bmatrix}^T$$
$$X = \begin{bmatrix} 69 & 38 & 67 \end{bmatrix}$$
$$L = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T$$
$$e = \begin{bmatrix} 120 & 92 & 6 & 0 & 0 & 36 & 90 \end{bmatrix}^T$$
$$\hat{c} = \begin{bmatrix} 251 & 162 & 184 & 5 & 252 & 248 & 5 \end{bmatrix}^T$$

O posto $e = 3 > (d-1)/2$, e portanto, se não tivessemos utilizado o algortimo de decodificação generalizado não conseguiriamos decodificar a mensagem. □

3.6 Codificação

Seja $u = \begin{bmatrix} u_0 & u_1 & \cdots & u_{k-1} \end{bmatrix}^T \in \mathbb{F}_{q^m}^k$ a mensagem que desejamos codificar. Uma maneira de codificação seria, simplesmente, multiplicar pela matriz geradora, isto é, $c = G^T u \in \mathcal{C}$. Podemos fazer proveito da forma como G é construida. Seja $u(x) = \sum_{i=0}^{k-1} u_i x^{[i]}$ então $c_i = u(g_i)$, $i = 0, \ldots, n-1$.

Como mostrado em [20], quando o código é construido com uma alta taxa e usamos bases normais de baixa complexidade, podemos reduzir o custo operacional da codificação.

Para isso, suponha que $r_i = 0$, $i = 0, \ldots, n-k-1$ e $r_i = u_{i-n+k}$, $i = n-k, \ldots, n-1$, e então usamos o algoritmo de decodificação para apagamentos em $r = \begin{bmatrix} r_0 & r_1 & \cdots & r_{n-1} \end{bmatrix}$. Note que, como já conhecemos todas as localizações dos apagamentos, o algoritmo de decodificação pode ser simplificado. Ou seja, L_j é um vetor coluna que tem como entrada 1 na j-ésima posição e 0 nas demais, $j = 1, \ldots, d-1$.

O polinômio ELP $\Lambda(x)$ será igual ao mínimo polinômio de $X_1, \ldots, X_{d-1}$, em que $X_j = L_j^T h,\ j = 1, \ldots, d-1$, ou seja, o polinômio será $\Lambda(x) = M_{\{X_1, \ldots, X_{d-1}\}}(x)$. A seguir podemos usar o algoritmo de deco-

dificação ELP (sem a generalização).

3.7 Considerações sobre a Complexidade

A complexidade de ambos os métodos (ESP e ELP) é equivalente no caso em que não há apagamentos nem desvios [5, 20]. Quando há mais informações sobre apagamentos do que desvios, i.e, $\mu > \delta$, então o método ELP é mais eficiente. Já quando há mais informações sobre desvios, ou seja, $\delta > \mu$ o método ESP se mostra mais eficiente.

Note que, como estamos trabalhando com bases normais, o custo computacional para elevar algum elemento do corpo a uma potência de Frobenius é desprezível. E a complexidade da multiplicação entre elementos é da ordem da complexidade da base normal [6].

3.8 Aplicação dos Códigos de Gabidulin na Codificação de Rede

Como visto na seção 2.3 quando uma rede apresenta erros e deficiência de posto, códigos que utilizam a distância de Hamming pode não ser suficientes. Entretando códigos que utilizam a distância de posto são capazes de corrigir *erros de posto*, o que os torna úteis na codificação de rede.

Devido a bijeção entre um corpo finito e sua $m-$ésima extensão, os códigos de Gabidulin, que trabalha com vetores, pode ser visto como um código matricial. Seja $\mathcal{C} \subseteq \mathbb{F}_{q^m}^n$ um código de Gabidulin (n, k). Suponha que $U \in \mathbb{F}_q^{k\times m}$ seja a matriz tal que cada linha contém um pacote que o nó-fonte deseja transmitir. Podemos então utilizar a bijeção e transformar essa matriz em um vetor, isso é $u = \varphi^{-1}(U) \in \mathbb{F}_{q^m}^k$. Usando o código $\mathcal{C}$, podemos transformar o vetor u em um vetor $c \in \mathcal{C}$ e então utilizarmos novamente a bijeção para formar a matriz $C = \varphi(c) \in \mathbb{F}_q^{n\times m}$. Note que a matriz C é uma palavra-código.

Seja $X = \begin{bmatrix} I_n & x \end{bmatrix} \in \mathbb{F}_q^{n\times n+m}$ a matriz enviada pelo nó-fonte, em que, I_n é a matriz identidade $n \times n$ e $x = C \in \mathbb{F}_q^{n\times m}$ é a matriz obtida no processo anterior (isto é, $x \in \mathcal{C}$).

Suponha que uma rede a deficiência de posto seja ρ e que t arestas tenham erros inseridos. Então o código $\mathcal{C}$ é capaz de recuperar a

mensagem, de forma única, se e somente se $2t + \rho < \mathrm{d_R}(\mathcal{C})$ [20].

Suponha que a matriz que contém os pacotes recebidos pelo nó-destino seja da forma $Y = \begin{bmatrix} \bar{A} & y \end{bmatrix}$. Note que, se não houver erros na rede, então $\bar{A} = A$ e $y = Ax$. Como mostrado por [20], nosso desafio é encontrar uma palavra-código $\hat{x} \in \mathcal{C}$ tal que

$$\hat{x} = \underset{x \in \mathcal{C}}{\operatorname{argmin}} \operatorname{posto}(y - \bar{A}x). \tag{3.41}$$

Note que se a matriz $\bar{A}$ for inversível, e considerando que $N = n$, podemos definir $r = \bar{A}^{-1}y \in \mathbb{F}_q^{n \times m}$ e então nosso desafio é encontrar $\hat{x} = \operatorname{argmin}_{x \in \mathcal{C}} \operatorname{posto}(r - x)$. Esse caso é semelhante ao apresentado na seção 3.4, portanto os algoritmos apresentados ali são suficientes para recuperar a mensagem.

Porém, em geral, a matriz $\bar{A}$ pode ser não-inversível. Por conveniência, assuma que $\operatorname{posto} Y = N$, isto é, desconsidere os pacotes linearmente dependentes aos demais recebidos pelo nó-destino. Seja $\mu = n - \operatorname{posto} \bar{A}$ a deficiência de posto de coluna e seja $\delta = N - \operatorname{posto} \bar{A}$ a deficiência de posto de linha da matriz $\bar{A}$. Seja $\mathrm{RRE}(Y)$ a forma escalonada reduzida (por linhas) de Y. O processo de decodificação é explicado em [20, 22], e será reproduzido (de forma simplificada) aqui:

Passo 1: Converter Y para sua forma escalonada reduzida.

Note que as linhas nulas de $\mathrm{RRE}(Y)$ são excluídas, uma vez que nenhuma informação pode ser obtida a partir de um vetor nulo.

Passo 2: Inserir μ linhas nulas.

Seja $\bar{Y}$ a matriz construida a partir de $\mathrm{RRE}(Y)$. Devemos inserir linhas nulas de modo que os pivôs ficam na posição i, i da matriz $\bar{Y}$, $i = 1, \ldots, N$. Dessa maneira a matriz $\bar{Y}$ será do tipo:

$$\bar{Y} = \begin{bmatrix} J & r \\ 0 & \hat{V} \end{bmatrix}, \tag{3.42}$$

em que $r \in \mathbb{F}_q^{n \times m}$, $\hat{V} \in \mathbb{F}_q^{\delta \times m}$ e $J \in \mathbb{F}_q^{n \times n}$, onde J é uma matriz que para as colunas que tem o pivô ela é igual a matriz identidade.

Passo 3: Extrair as matrizes $r, \hat{L}, \hat{V}$.

As matrizes r e $\hat{V}$ são obtidas diretamente de $\bar{Y}$. Para obter a matriz $\hat{L}$ devemos considerar uma sub-matriz de J, composto pelas colunas de J que não apresentam um pivô. A matriz $\hat{L}$ será aquela que ao ser subtraida dessa sub-matriz de J teremos uma sub-matriz da matriz identidade.

Chamamos de *redução* da matriz Y a tupla $(r, \hat{L}, \hat{V}) \in \mathbb{F}_q^{n\times m} \times \mathbb{F}_q^{n\times\mu} \times \mathbb{F}_q^{\delta\times m}$ obtida do processo acima. Note que agora caimos num caso semelhante ao apresentado na seção 3.5 e, portanto, devemos utilizar os algoritmos de decodificação generalizado.

Exemplo. Seja $q = 5$, $m = 8$, $n = 7$ e $\mathrm{d_R}(\mathcal{C}) = 5$. Suponha que

$$x = \begin{bmatrix} 2 & 0 & 4 & 1 & 2 & 0 & 2 & 3 \\ 0 & 1 & 3 & 2 & 3 & 4 & 0 & 2 \\ 1 & 2 & 3 & 2 & 4 & 1 & 3 & 1 \\ 2 & 0 & 2 & 4 & 3 & 4 & 0 & 0 \\ 0 & 4 & 4 & 4 & 1 & 1 & 2 & 2 \\ 1 & 2 & 0 & 4 & 0 & 2 & 4 & 4 \\ 0 & 0 & 1 & 1 & 4 & 0 & 0 & 0 \end{bmatrix},$$

seja a mensagem transmitida pelo nó-fonte (antes da inserção do cabeçalho). E suponha que a matriz recebida pelo nó-destino seja

$$Y = \left[\begin{array}{ccccccc|cccccccc} 0 & 1 & 3 & 1 & 3 & 0 & 4 & 3 & 0 & 2 & 4 & 4 & 2 & 1 & 3 \\ 1 & 0 & 3 & 0 & 0 & 1 & 2 & 2 & 0 & 4 & 0 & 1 & 1 & 2 & 4 \\ 1 & 3 & 4 & 1 & 3 & 0 & 3 & 1 & 4 & 4 & 0 & 2 & 1 & 1 & 1 \\ 4 & 3 & 1 & 4 & 0 & 4 & 2 & 1 & 3 & 3 & 1 & 1 & 2 & 2 & 0 \\ 4 & 1 & 3 & 2 & 3 & 1 & 2 & 3 & 0 & 4 & 2 & 3 & 2 & 1 & 0 \\ 2 & 4 & 0 & 1 & 0 & 0 & 1 & 3 & 3 & 1 & 4 & 1 & 2 & 3 & 2 \\ 2 & 4 & 2 & 2 & 2 & 1 & 3 & 0 & 1 & 2 & 0 & 0 & 4 & 1 & 0 \end{array}\right].$$

A redução de Y é mostrada a seguir.

Passo 1. Encontrando a forma escalonada reduzida de Y, note que, excluimos a linha nula nesse passo, dessa forma temos que posto $Y =$

$N = 6$, enquanto posto $\bar{A} = 5$, e portanto, temos que $\mu = 2$ e $\delta = 1$.

$$\mathrm{RRE}(Y) = \left[\begin{array}{cccccccc|cccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 4 & 0 & 2 \\ 0 & 1 & 0 & 0 & 3 & 0 & 1 & 0 & 3 & 1 & 0 & 0 & 2 & 1 & 3 \\ 0 & 0 & 1 & 0 & 3 & 0 & 2 & 0 & 2 & 3 & 4 & 1 & 3 & 2 & 3 \\ 0 & 0 & 0 & 1 & 1 & 0 & 2 & 0 & 0 & 0 & 1 & 4 & 3 & 3 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 2 & 3 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 4 & 2 & 4 & 1 & 2 & 4 \\ \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} \end{array}\right]$$

Passo 2. Ao inserir linhas nulas nos locais apropriados, percebemos a formação das sub-matrizes J, r e $\hat{V}$, além de uma sub-matriz nula.

$$\bar{Y} = \left[\begin{array}{cccccccc|cccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 4 & 0 & 2 \\ 0 & 1 & 0 & 0 & 3 & 0 & 1 & 0 & 3 & 1 & 0 & 0 & 2 & 1 & 3 \\ 0 & 0 & 1 & 0 & 3 & 0 & 2 & 0 & 2 & 3 & 4 & 1 & 3 & 2 & 3 \\ 0 & 0 & 0 & 1 & 1 & 0 & 2 & 0 & 0 & 0 & 1 & 4 & 3 & 3 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 2 & 3 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 4 & 2 & 4 & 1 & 2 & 4 \end{array}\right]$$

Passo 3. Na matriz J, a 5^{a} e a 7^{a} coluna não apresentam pivôs. Considerando uma matriz formada somente por essas colunas, então a matriz $\hat{L}$ será tal que ao ser subtraida teremos uma sub-matriz, formada pela 5^{a} e 7^{a} coluna, da matriz identidade (lembre que $-1 \equiv 4$ em $\mathbb{F}_5$). As matrizes r e $\hat{V}$ são obtidas diretamente no passo anterior. Assim

$$r = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 4 & 0 & 2 \\ 0 & 3 & 1 & 0 & 0 & 2 & 1 & 3 \\ 0 & 2 & 3 & 4 & 1 & 3 & 2 & 3 \\ 0 & 0 & 0 & 1 & 4 & 3 & 3 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 3 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \hat{L} = \begin{bmatrix} 1 & 0 \\ 3 & 1 \\ 3 & 2 \\ 1 & 2 \\ 4 & 0 \\ 0 & 1 \\ 0 & 4 \end{bmatrix}$$

$$\hat{V} = \begin{bmatrix} 1 & 2 & 4 & 2 & 4 & 1 & 2 & 4 \end{bmatrix}$$

Note que, a partir da decodificação generalizada, temos que $\tau = \text{posto}(r - x) = 3$, $\epsilon = \tau - \mu - \delta = 3 - 2 - 1 = 0$ e, então $2\epsilon + \mu + \delta = 3 < \mathrm{d_R}(\mathcal{C}) = 5$. Portanto, com o código matricial adequado podemos recuperar a mensagem enviada pelo nó-fonte.

□

Devemos agora garantir que ao usar a redução de Y, os valores de ϵ, δ e μ sejam tais que o código $\mathcal{C}$ seja capaz de corrigir. Lembre-se que, se t arestas foram corrompida e a deficiência de posto é ρ, o código $\mathcal{C}$ é capaz de recuperar a mensagem de forma única se $2t + \rho < \mathrm{d_R}(\mathcal{C})$. Em [20] é mostrado que se essa condição for satisfeita então os valores de ϵ, δ e μ encontrados serão tais que $2\epsilon + \delta + \mu < \mathrm{d_R}(\mathcal{C})$.

É importante notar que, se $t = 0$, isto é, não há erros na rede, somente apagamentos, então $\mu = \rho$ e $\epsilon = \delta = 0$. E portanto o código $\mathcal{C}$ é capaz de recuperar a mensagem se $\mu < \mathrm{d_R}(\mathcal{C})$.

3.9 Implementação

O codificador e o decodificador de Gabidulin foram implementados em *Python*[2], usando a plataforma *SAGE* [25]. Visando diminuir a complexidade, as rotinas foram feitas usando bases normais e polinômios linearizados.

Tais rotinas serão unidas com o simulador de redes, que será explicado no próximo capítulo. Assim poderemos imaginar um cenário comlpleto de codificação da mensagem, transmissão em uma rede com

[2] `http://www.python.org/`

codificação de rede, e por fim, a decodificação poderá ser feita, mesmo que a rede apresente erros e apagamentos.

CAPÍTULO 4

Aplicação: Minimização de Overhead em Redes com Apagamento

Iremos agora unir codificação de rede e códigos matriciais corretores de erro (em especial códigos de Gabidulin) e ver o efeito em uma aplicação prática, a minimização do *overhead* em redes com apagamentos, ou seja, considaremos que a rede não apresenta erros. O overhead nos dá uma ideia da informação redundante recebida por um nó-destino, seja porque esse nó recebeu pacotes linearmente dependentes aos demais, seja porque tivemos que inserir o cabeçalho de identificação em cada pacote.

Nesse capítulo explicaremos com maiores detalhes o overhead e uma estratégia para minimizá-lo. A fim de obter resultados mais realistas, foi implementado um simulador de redes que retorna o número de pacotes que um nó-destino deve receber para realizar a decodificação. Esse simulador será explicado nesse capítulo.

Por fim, serão mostrados os resultados obtidos utilizando a estratégia de minimização de overhead e o simulador para algumas topologias de rede.

4.1 Overhead

Considere o caso que não há erros, isto é, $t = 0$, porém pode haver apagamentos na rede. Sabemos que podemos caracterizar a rede atráves da equação

$$Y = AX, \tag{4.1}$$

em que, $X \in \mathbb{F}_q^{k\times(k+m)}$, $Y \in \mathbb{F}_q^{N\times(k+m)}$ e $A \in \mathbb{F}_q^{N\times k}$.

Sabe-se que para recuperar a mensagem é necessário que posto $A = k$. Portanto, o menor valor de N que isso pode ocorrer é quando $N = k$ (note que N ser o menor possível implica que o nó-destino receba a menor quantidade possível de pacotes para recuperar a mensagem). Todavia, devido aos apagamentos que podem ocorrer na rede a matriz A apresenta uma deficiência de posto. Como visto na seção 2.3, quando a matriz A apresenta deficiência de posto é necessário esperar que o nó-destino receba mais pacotes. Esses pacotes a mais dão origem ao que chamamos de *overhead*.

O overhead pode ser definido como a quantidade de informação não-útil (redundante) em relação a quantidade de informação útil recebida pelo nó-destino. Além dos apagamentos, essa redundância pode ocorrer devido as combinações feitas para o envio de um pacote (lembre-se que os coeficientes são aleatórios então existe a probabilidade não-nula de um pacote ser linearmente dependente aos demais), como também ocorre devido ao cabeçalho de identificação colocado em cada pacote.

Considere que o nó-fonte deseja enviar um arquivo com D bits. Devido a certas limitações, a rede permite enviar pacotes com no máximo P bits, em que $P < D$. Assim, o nó-fonte divide o arquivo em k pacotes, tal que $k = D/P$. Sabemos da codificação de rede linear (em particular aleatória) que cada pacote pode ser visto como um vetor $m-$dimensional sobre $\mathbb{F}_q$, em que $m = P/\log_2 q$. Por conveniência, iremos assumir que D, P, k e m são escolhidos de tal maneira que todos eles são números inteiros.

4.1.1 Overhead de Dependência Linear

O *overhead de dependência linear* ocorre devido à possibilidade de um nó-destino ter que receber mais do que k pacotes para que posto $A = k$. Esses pacotes a mais recebidos são devidos as combinações lineares poderem gerar um pacote linearmente dependente, assim como o fato

de ocorrer apagamentos na rede [21, 23].

O overhead de dependência linear (ϵ_D) é definido como sendo

$$\epsilon_D \triangleq \frac{E[N] - k}{k}, \tag{4.2}$$

em que $E[N]$ é o valor esperado do número de pacotes que o nó-destino recebe até que posto $A = k$.

Podemos calcular o overhead teórico para o caso de uma rede simples, com apenas um nó-fonte e um nó-destino, e sem nós-intermediários. Para calcular o valor esperado do números de pacotes que o nó-destino deve receber é usado o conceito de cadeias de Markov absorventes [8]. Em [21] o cálculo é demonstrado. A seguir reproduzimos o resultado final:

$$E[N] = \sum_{r=0}^{k-1} \frac{1}{1 - q^{r-k}}.$$

Finalmente, podemos calcular o overhead de dependência linear

$$\epsilon_D = \frac{1}{k} \sum_{r=0}^{k-1} \frac{q^{r-k}}{1 - q^{r-k}}, \tag{4.3}$$

note que, aumentando o tamanho do corpo finito conseguimos reduzir o overhead de dependência linear [21, 23].

4.1.2 Overhead de Cabeçalho

O *overhead de cabeçalho* ocorre devido ao fato de termos que inserir um cabeçalho de informação, contendo k símbolos, em cada pacote [21, 23]. O overhead de cabeçalho (ϵ_H) é definido como

$$\epsilon_H \triangleq \frac{1}{P} k \log_2 q = \frac{k}{m}.$$

Note que, se k e q estiverem fixo e P for suficientemente grande, o tamanho do cabeçalho é desprezível (e consequentemente o overhead de cabeçalho é desprezível). Porém, em pacotes com P pequeno, o overhead de cabeçalho pode ser significativo. Por fim, se k e P estiverem fixo então o overhead de cabeçalho aumenta conforme aumentamos o tamanho do corpo finito utilizado.

4.1.3 Outros Overheads

Há ainda outros tipos de overhead, como o *overhead de geração* [21, 23]. Este overhead tem como origem o fato do nó-destino continuar recebendo pacotes de uma determinada geração, mesmo que ela já esteja completa. Note que este pacote será linearmente dependente aos demais. Esse overhead independe do tamanho do corpo finito utilizado e pode ser minimizado utilizando uma melhor estratégia na escolha de gerações [24]. Com isso, o overhead de geração não faz parte do escopo desse traballho.

4.1.4 Overhead Intrínseco

O *overhead intrínseco* é aquele que depende dos parâmetros básicos da rede, tais como o corpo finito utilizado, o número de pacotes enviados pelo nó-fonte, etc (portanto não consideramos o overhead de geração) [21, 23].

Na Fig. 4.1 podemos ver a representação do overhead intrínseco. Em azul (\\\\), a parte correspondente ao overhead de cabeçalho. Em verde (////), a parte correspondente ao overhead de dependência linear.

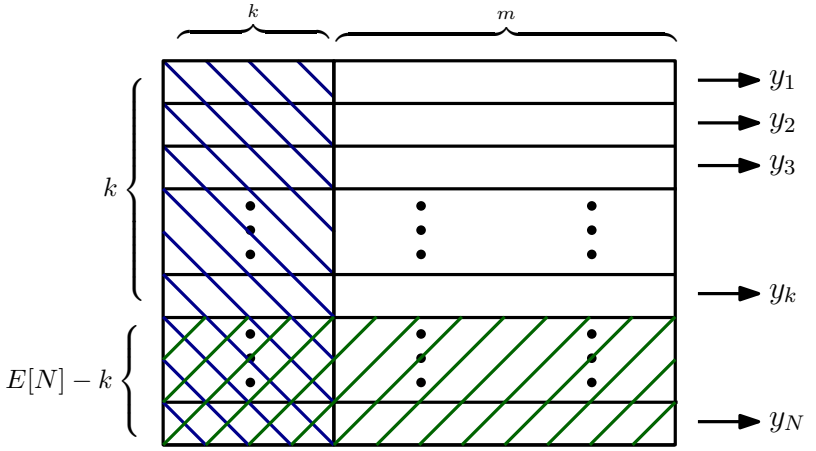


Figura 4.1: Representação do overhead intríseco nos pacotes recebidos pelo nó-destino.

O overhead intrínseco (ϵ_I) pode ser calculado como:

$$\begin{aligned}\epsilon_I &= \frac{\text{símbolos com informação não-util}}{\text{símbolos com informação útil}} \\ &= \frac{k^2 + (E[N]-k)(k+m)}{km} \\ &= \frac{(E[N]-k)m + k^2 + (E[N]-k)k}{km} \\ &= \frac{E[N]-k}{k} + \frac{k}{m} + \left(\frac{E[N]-k}{k}\right)\frac{k}{m} \\ &= \epsilon_D + \epsilon_H + \epsilon_D\epsilon_H, \end{aligned} \tag{4.4}$$

o termo cruzado muitas vezes é desprezado (note que ele será pequeno em relação a ϵ_D e ϵ_H).

O overhead intrínseco mostra claramente o *tradeoff* entre o overhead de dependência linear e o overhead de cabeçalho. Enquanto um é minimizado aumentando o tamanho do corpo finito, o outro é maximizado.

Para uma rede com apenas um nó-fonte e um nó-destino, o overhead intrínseco é mostrado na Fig. 4.2. Notamos claramente que, quando o nó-fonte envia poucos pacotes (k pequeno) o overhead de depedência linear é mais relevante, fazendo com que a melhor escolha de corpo finito seja q tão grande quanto possível. Todavia, conforme aumentamos o tamanho da geração, o overhead de cabeçalho ganha mais relevância e devemos então diminuir o tamanho do corpo finito para minimizar o overhead intrínseco.

4.2 Minimizando o Overhead Intrínseco

Seja $\mathcal{C} \subseteq \mathbb{F}_q^{n\times m}$ um código MRD com parâmetros (n,k) e seja $d = n-k+1$ a mínima distância de $\mathcal{C}$. Suponha que k pacotes da fonte, sejam previamente codificados, gerando então n pacotes. Note que agora $X = \begin{bmatrix} I_n & x \end{bmatrix} \in \mathbb{F}_q^{n\times n+m}$, $x \in \mathcal{C}$, é matriz que contém os pacotes enviados pelo nó-fonte.

A princípio, para o nó-destino conseguir recuperar a mensagem original teríamos que esperar até que $\operatorname{posto} A = n$, e sem a codificação bastaria esperar até que $\operatorname{posto} A = k$. Todavia, devido a propriedade de correção de deficiência de posto, uma vez que a matriz A tenha posto igual a k podemos utilizar o algoritmo de decodificação generalizada e

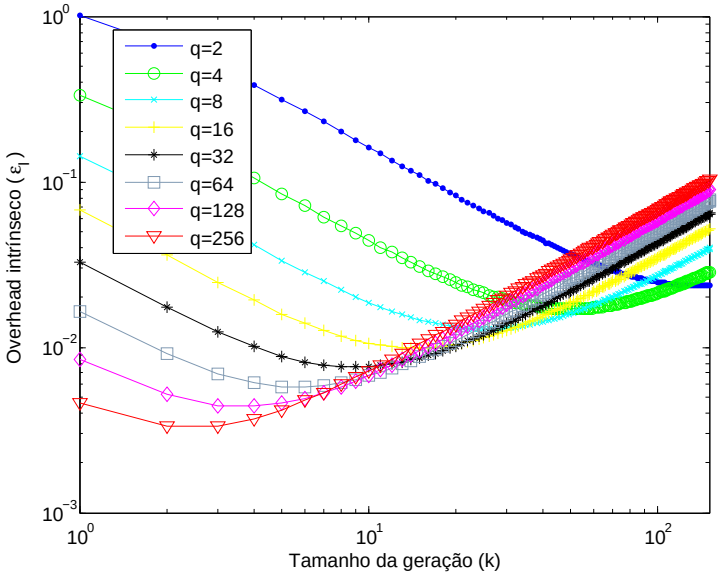


Figura 4.2: Overhead intrínseco para a rede com um nó-fonte e um nó destino.

recuperar a mensagem original [21, 23].

Como mostrado na Fig. 4.3, quando codificamos a mensagem aumentamos o número de pacotes enviados pelo nó-fonte o que aumenta o tamanho do cabeçalho (e por consequência, o overhead de cabeçalho). Assim precisamos escolher um ponto ótimo, com o qual conseguimos reduzir o overhead de depência linear, sem aumentar muito o overhead de cabeçalho.

Considere a rede que contém um nó-fonte, um nó-destino e sem nós-intermediários. Seja $\mu = n - k$ a redundância inserida pelo código $\mathcal{C}$. Substituindo na equação (4.4), desprezando o termo cruzado, e escrevendo em função de μ, o overhead intrínseco será

$$\epsilon_I(\mu) = \frac{k+\mu}{m} + \frac{1}{k}\sum_{r=0}^{k-1} \frac{1}{q^{k+\mu-r}-1}, \tag{4.5}$$

assim, como mostrado em [21, 23], podemos aproximar o valor de μ^*,

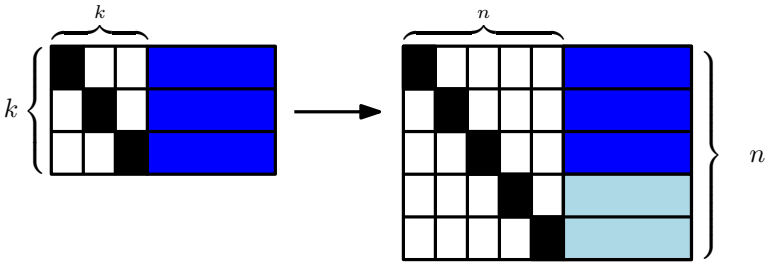


Figura 4.3: Transformando os k pacotes em n pacotes usando códigos MRD. Note que o tamanho do cabeçalho também aumenta.

que é aquele que minimiza o overhead intrínseco como sendo

$$\mu^* \approx \lfloor \log_q(1 + m/k) \rfloor, \tag{4.6}$$

em que $\lfloor x \rfloor$ denota o maior inteiro menor ou igual a x.

A Fig. 4.4 nos mostra o que acontece ao usar essa estratégia em uma rede com um nó-fonte e um nó-destino. Como a rede é simples, foi possível calcular tanto o resultado teórico que foi usado para comparar com o resultado obtido através do simulador (que será explicado posteriormente).

Um fato que é possível notar é que o tamanho do corpo que minimiza o overhead é $q = 2$, independente do tamanho de geração utilizado. Isso evita que tenhamos que mudar o corpo finito utilizado para cada tamanho de geração. E mais do que isso, $q = 2$ é o menor corpo finito que existe, assim as operações nesse corpo são mais simples.

4.3 Redes mais Complexas

É natural se questionar se esses resultados permancem caso a rede seja mais complexa. Calcular o valor de μ^* para qualquer rede pode ser um processo complexo. Porém utilizando o valor calculado para a rede com um nó-fonte e um nó-destino é possível obter os mesmo resultados desde que haja uma *política de acumulação* [21, 23].

A ideia básica dessa estratégia é que nós intermediários devem esperar até terem h pacotes linearmente independentes (chamaremos h de *limiar de acumulação*) para então enviar um pacote.

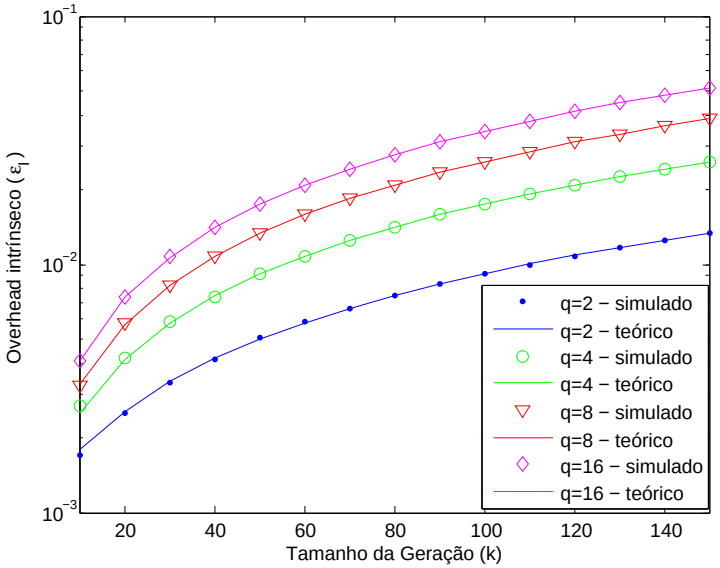


Figura 4.4: Overhead intrínseco para a rede com um nó-fonte e um nó destino, usando códigos MRD e fazendo $\mu = \mu^*$.

A estratégia evita com que nós, com poucos pacotes linearmente independentes, enviem pacotes que teriam alta probabilidade de serem redundantes.

Pode-se pensar que quanto maior o limiar de acumulação, mais facilmente será evitado que pacotes redundantes sejam transmitidos. Todavia, conforme mostrado na Fig. 4.5, em um certo ponto o overhead intrínseco "satura", isto é, não apresenta melhoras significativas usando um limiar maior.

4.4 Simulador de Redes

Para obter resultados mais realistas foi implementado um simulador de redes, em linguagem *Python*, utilizando a plataforma *SAGE* [25]. Com o simulador podemos saber quantos pacotes o nó-destino deve receber para realizar a decodificação. Os códigos MRD podem ou não ser usados. Os parâmetros de entrada do simulador são:

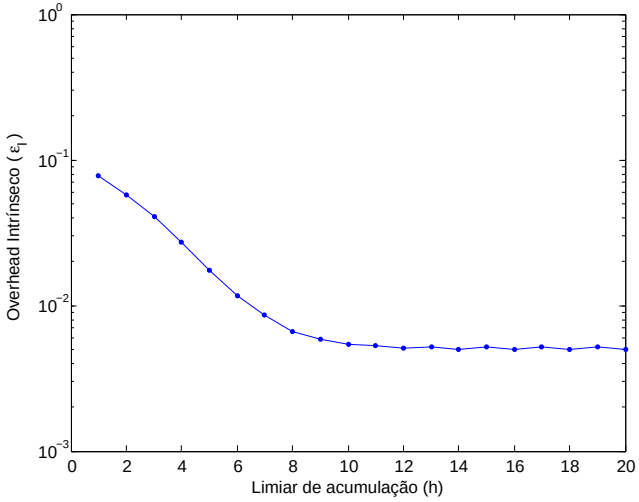


Figura 4.5: Overhead intrínseco para a rede com um nó-fonte e dois nós-destino, usando $\mu = \mu^*$ e $k = 50$.

- O tamanho do corpo finito utilizado - q;
- O tamanho de cada geração - k;
- O número de pacotes enviado pelo nó-fonte - n;
- Número de gerações - L;
- Limiar de acumulação - h;
- Probabilidade de apagamento - p_ρ.

Também é necessário especificar o grafo utilizado para resepresentar a rede, assim como o conjunto de nós-destino T e o nó-fonte s.

A seguir, criamos um "buffer" para cada nó da rede; esse buffer armazenará a matriz de transferência de cada nó. Para o nó-fonte esse buffer é substituido pela matriz identidade. Note que o nó-fonte contém os pacotes originais e não combinações desses pacotes.

Para verificar se há uma ligação entre um nó e outro usamos a matriz de adjacência do grafo H. Se o coeficiente $h_{ij} \geq 1$ então o nó i pode enviar para o nó j. Note que, na prática, todos os nós enviam

pacotes ao mesmo tempo, porém, na simulação, o envio de pacotes é feito de forma sequencial. Para evitar que um nó utilize um pacote, que na prática ele ainda não recebeu, na combinação linear que ele irá enviar, os pacotes enviados em um uso do canal são armazendos em uma matriz temporária. Ao final de cada iteração, essa matriz é anexada ao buffer de cada nó. Assim, não precisamos nos preocupar com a numeração dada a cada nó. Todavia, é usual fazer com que o nó-fonte tenha o índice 0.

Após selecionar qual aresta irá enviar o pacote, o nó deve escolher qual geração será usada. A escolha da geração não interfere no restante da simulação, ou seja, é função à parte. Como, nos exemplos utilizados, trabalhamos apenas com gerações únicas, esse passo se torna trivial, porém o simulador é genérico o suficiente pra lidar com qualquer número de gerações. Escolhida a geração, é necessário verificar se o posto da matriz de transferência do nó (e daquela geração) é maior que o limiar de acumulação definido. Se não for, simplesmente vamos para a aresta seguinte. Caso seja maior, então é feita uma combinação linear, com os coeficientes sendo escolhidos de forma aleatória e uniforme sobre $\mathbb{F}_q$.

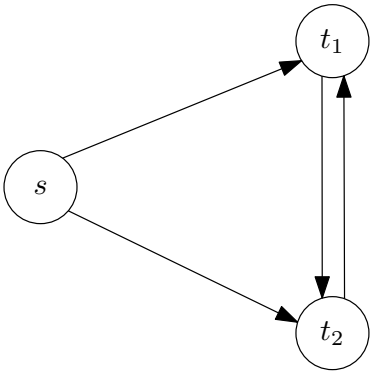
O pacote pode ser, então, enviado para o nó subsequente, porém, é necessário verificar se o pacote foi apagado ou não. Se trabalhassemos, além de apagamentos, com erros, o vetor de erro seria somado durante esse passo. Recebendo ou não o pacote, passamos a fazer o mesmo processo na próxima aresta (que é determinada atraves da matriz H).

Quando o pacote chega em um nó-destino, adicionamos mais 1 em um acumulador, assim teremos o controle de quantos pacotes chegaram em cada destino. Assim que todos as gerações de todos os nós-destino estejam completas a simulação é finalizada, e teremos então, o número de pacotes necessários para que todas as matrizes de transferência tenham posto igual a k.

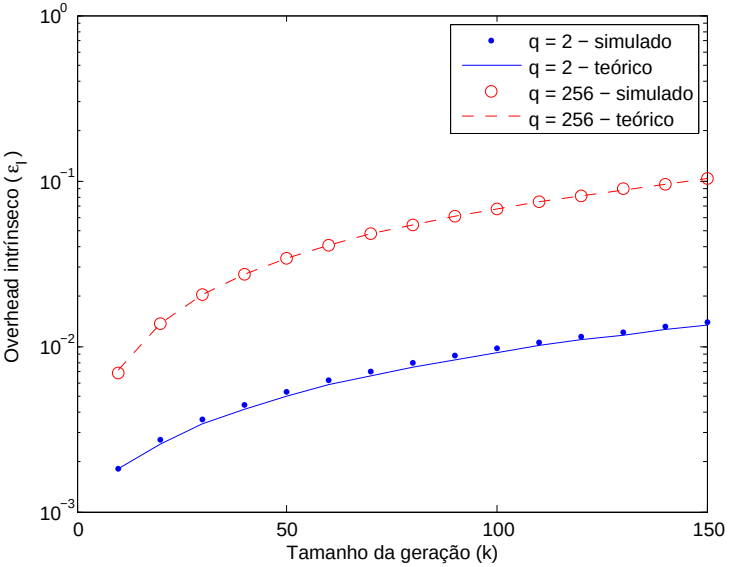
4.5 Resultados

A seguir serão apresentados os resultados da simulação para várias topologias de rede. Tais resultados são comparados com o resultado para um rede com um nó-fonte e um nó-destino, obtido na Fig. 4.4. Para todas as simulações, a redundância inserida foi $\mu = \mu^*$. Além disso, foi utilizado o limiar de acumulação $h = 10$ e probabilidade de

apagamento $p_\rho = 0{,}11$.

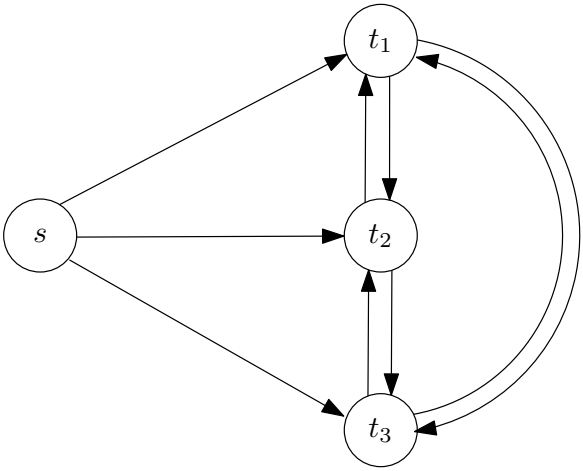


(a) Topologia da Rede.

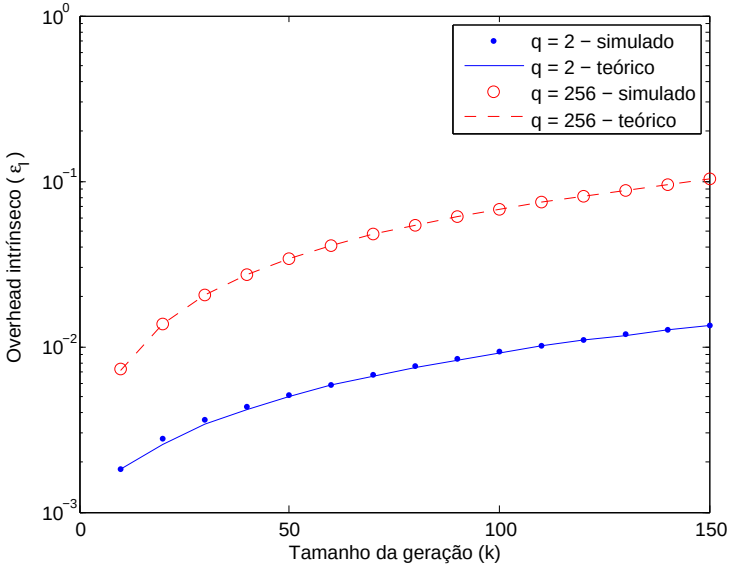


(b) Overhead intrínseco da rede.

Figura 4.6: Rede com um nó-fonte e dois nós-destinos que comunicam-se entre si.

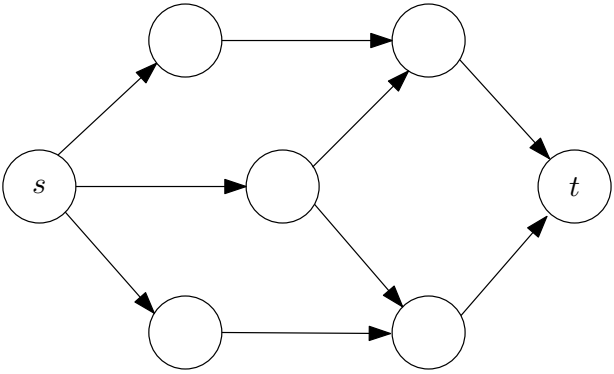


(a) Topologia da Rede.

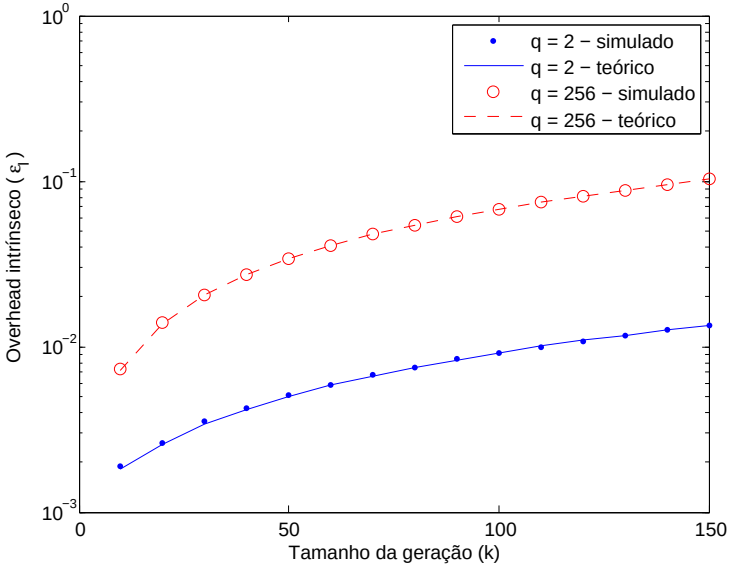


(b) Overhead intrínseco da rede.

Figura 4.7: Rede com um nó-fonte e três nós-destinos que comunicam-se entre si.



(a) Topologia da Rede.



(b) Overhead intrínseco da rede.

Figura 4.8: Rede com um nó-fonte, um nó-destino e cinco nós-intermediários que auxiliam a comunicação.

Ao analisar as figuras notamos que o tamanho do corpo finito que minimiza o overhead intrínseco é $q = 2$. Também é possível notar que esse resultado se mantém, independente da topologia da rede, quando utilizamos a estratégia com o limiar de acumulação.

CAPÍTULO 5

Conclusão

A codificação de rede linear aleatória se mostra útil nos casos em que a topologia de rede varia com o tempo, uma vez que a codificação de rede não depende da topologia. Além disso, em casos com vários destinos, a codificação de rede permite atingir melhores taxas de transmissão.

Mesmo na presença de erros, a codificação de rede ainda se mostra útil desde que utilizemos em conjunto um código matricial corretor de erros. Esse código pode ser, em particular, o código de Gabidulin, o qual utiliza um mapeamento entre um corpo finito e sua extensão. Tal código possui a capacidade de corrigir erros de posto, o que o torna interessante na codificação de rede.

O overhead intrínseco, que é um dos desafios encontrados na codificação de rede, pode ser minimizado utilizando códigos de Gabidulin. Mais do que isso, ao utilizar os códigos corretores de erros, vimos que o mínimo overhead ocorre para $q = 2$, o que reduz a complexidade computacional das operações. Note que é muito mais simples trabalhar em corpos cuja a cardinalidade é um número primo do que trabalhar com alguma extensão. Isso também é uma vantagem se pensarmos em possíveis implementações práticas em hardware.

Essa estratégia, de minimização do overhead, pode ser usada em

qualquer topologia de rede, desde que respeitando o limiar de acumulação. Assim, a inclusão dessa estratégia em futuras aplicações pode ser interessante.

5.1 Trabalhos futuros

Algumas sugestões para futuros trabalhos:

- Incluir, nas simulações, o overhead de geração, causado pelo fato de uma geração que esteja completa continuar recebendo pacotes, e então medir a eficiência desse método considerando todos os overheads;
- Incluir também erros na rede. Tais erros devem estar dentro dos limites impostos na seção 2.3. E então, novamente, verificar se o overhead ainda pode ser minimizado ao utilizarmos códigos matriciais;
- Criar um cenário completo de simulação, em Python, o qual poderá ser disponibilizado com a comunidade
- Fazer implementações práticas da codificação de rede com códigos matriciais.
- Investigar o uso dos códigos de Gabidulin em sistemas de armazenamento distribuído [19].

Referências bibliográficas

[1] R. Ahlswede, N. Cai, R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul 2000.

[2] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proceeding of Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Oct 2003, pp. 40–49.

[3] B. Cohen, "The BitTorrent protocol specification," Jan 2008, http://www.bittorrent.org/beps/bep_0003.html.

[4] E. M. Gabidulin, "Theory of codes with maximum rank distance," *Problems Inform. Transmission*, vol. 21, pp. 1–12, Jul 1985, in English translation.

[5] M. Gadouleau and Z. Yan, "Complexity of decoding Gabidulin codes," in *Proceeding of 42nd Annual Conference on Information Sciences and Systems (CISS'08*, Princeton, NJ, Mar 2008, pp. 1081–1085.

[6] S. Gao, "Normal bases over finite fields," Ph.D. dissertation, University of Warteloo, Warteloo, Ontario, Canada, 1993.

[7] P. Garrett, *The Mathematics of Coding Theory.* Pearson Prentice Hall, 2004.

[8] C. M. Grinstead and J. L. Snell, *Introduction to Probability*, 2006.

[9] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, Oct 2006.

[10] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 1973–1982, Jun 2005.

[11] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct 2003.

[12] F. R. Kschischang, "An introduction to network coding," in *Newtork Coding: Fundamentals and Applications*, M. Medard and A. Sprintson, Eds. Academic Press, 2011, ch. 1.

[13] R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, Feb 2003.

[14] S. Lin and D. J. Costello Jr., *Error Control Coding*, 2nd ed. Pearson Prentice Hall, 2004.

[15] C. C. Pinter, *A Book of Abstract Algebra*. McGraw-Hill Book Company, 1982.

[16] G. Richter and S. Plass, "Error and erasure decoding of rank-codes with a modified Berlekamp-Massey algorithm," in *Proceeding of V International ITG Conference on Source and Channel Coding*, Erlangen, Germany, Jan 2004, pp. 249–256.

[17] V. Sidorenko, L. Jiang, and M. Bossert, "Skew-feedback shift-register synthesis and decoding interleaved Gabidulin codes," *IEEE Transactions on Information Theory*, vol. 57, pp. 621–632, Feb 2011.

[18] V. Sidorenko, G. Richter, and M. Bossert, "Linearized shift-register synthesis," *IEEE Transactions on Information Theory*, vol. 57, no. 9, pp. 6025–6032, Sep 2011.

[19] N. Silberstein, A. S. Rawat, and S. Vishwanath, "Error resilience in distributed storage via rank-metric codes," in *Proceeding of Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Oct 2012, pp. 1150–1157.

[20] D. Silva, "Error control for network coding," Ph.D. dissertation, University of Toronto, Toronto, Canada, Feb 2009.

[21] ——, "Minimum-overhead network coding in the short packet regim," in *Proceedings of International Symposium on Network Coding (NetCod'12)*, Cambrigde, MA, Jun 2012, pp. 173–178.

[22] D. Silva, F. R. Kschischang, and R. Kötter, "A rank-metric approach to error control in random network coding," *IEEE Transactions on Information Theory*, vol. 54, no. 9, pp. 3951–3967, Sep 2008.

[23] D. Silva and R. B. Venturelli, "Codificação de rede com mínimo overhead utilizando códigos de máxima distância de posto," in *Proceeding of XXX Simpósio Brasileiro de Telecomunicações (SBrT'12)*, Brasilia, Brazil, Set 2012.

[24] D. Silva, W. Zeng, and F. R. Kschischang, "Sparse network coding with overlapping classes," in *Proceedings of Workshop on Network Coding, Theory, and, Applications (NetCod'09)*, Lausanne, Jun 2009, pp. 74–79.

[25] W. Stein *et al.*, *Sage Mathematics Software (Version 5.12)*, The Sage Development Team, 2014, http://www.sagemath.org.

[26] D. B. West, *Introduction to Graph Theory*, 2nd ed. Pearson Pretice Hall, 2001.

[27] H. Xie, J. Lin, Z. Yan, and B. W. Suter, "Linearized polynomial interpolation and its applications," *IEEE Transactions on Signal Processing*, vol. 61, no. 1, pp. 206–217, Jan 2013.