Universidade Federal de Pernambuco

Centro de Informática

Pós-graduação em Ciência da Computação

UM MÉTODO DE GERENCIAMENTO DE FRAMES PARA O PROTOCOLO DFSA EM SISTEMAS RFID

Júlio Dantas de Andrade

DISSERTAÇÃO DE MESTRADO

Recife

10 de abril de 2012

Universidade Federal de Pernambuco

Centro de Informática

Júlio Dantas de Andrade

UM MÉTODO DE GERENCIAMENTO DE FRAMES PARA O PROTOCOLO DFSA EM SISTEMAS RFID

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Paulo André da Silva Gonçalves*

Recife

10 de abril de 2012

AGRADECIMENTOS

Agradeço, inicialmente, aos meus familiares, que me ajudaram em alguns momentos complicados e deram algumas opiniões e me ajudaram a tomar decisões no decorrer do mestrado.

Aos participantes do grupo de pesquisa do professor Paulo Gonçalves, que me ajudaram, por meio de opiniões e críticas sobre este trabalho e no desenvolvimento do mesmo.

Ao professor *Docteur* Paulo Gonçalves, meu orientador, que tenho certeza fez o melhor possível para me orientar enquanto estive sob sua orientação.

Gostaria de fazer uma menção honrosa para aqueles que, apesar de não serem família, possuem de, minha parte, tal consideração e também forneceram apoio em horas difíceis.

E a todas as pessoas que me ouviram, opinaram ou de algum modo, direto ou indireto, ajudaram na construção deste trabalho.

Do or do not, there is no try.

—YODA (Star Wars)

RESUMO

O DFSA (*Dynamic Framed Slotted ALOHA*) é um protocolo anticolisão popular para sistemas RFID. Neste protocolo, o tamanho de cada *frame* subsequente ao *frame* inicial é calculado dinamicamente com base na estimativa da população de etiquetas que competiram no *frame* anterior. As principais propostas de estimadores para o DFSA procuram maximizar a acurácia de suas estimativas na busca da minimização do tempo total de identificação de etiquetas. Contudo, elas desconsideram o impacto do uso da extensão *Early-End* no cálculo do tamanho dos *frames*. Esta dissertação mostra que oportunidades de otimização do tempo total de identificação de etiquetas são perdidas ao se desconsiderar o impacto dessa extensão no cálculo de tamanho de *frames* quando a mesma é utilizada no DFSA. A partir disso, esta dissertação propõe uma função para o cálculo do tamanho de *frames* que considera a estimativa do estimador usado e o impacto do uso da extensão *Early-End*. As avaliações de desempenho da função proposta com os estimadores Eom-Lee, Vogt e Schoute mostram que o tempo total de identificação de etiquetas é reduzido, respectivamente, em até 24%, 22% e 30%.

Palavras-chave: DFSA, *Early-End*, RFID, cálculo do tamanho de *frames*, tempo de identificação, desempenho.

ABSTRACT

DFSA (Dynamic Framed Slotted ALOHA) is a popular anti-collision protocol for RFID systems. In such protocol, the size of each frame succeeding the first frame is dynamically calculated based on the estimate of the number of competing tags in the previous frame. The main existing estimators for DFSA seek to increase their accuracy in order to minimize the total tags identification time. However, such estimators do not consider the impact of the Early-End extension when calculating frame sizes. In this dissertation, we show that there are missing optimization opportunities in the total tag identification time when DFSA uses such an extension but its impact is ignored in the frame size calculus. Based on this fact, we propose a function to calculate frame sizes that takes Performance evaluations of the proposed function with the Eom-Lee, Vogt, and Schoute estimators show that the total tag identification time is reduced, respectively, by up to 24%, 22%, and 30%.

Keywords: DFSA, Early-End, RFID, frame size calculus, identification time, performance.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

GLOSSÁRIO

API *Application Programming Interface*. 66

CDMA *Code Division Multiple Access*. 8

CLI *Command Line Interface*. 67

DFSA *Dynamic Framed Slotted ALOHA*. 2

FDMA *Frequency Division Multiple Access*. 8

FSA *Frame Slotted ALOHA*. 16

GLC Gerador de Congruência Linear. 66

ID Identificador. 1, 6

QT *Query Tree*. 10

RFID *Radio Frequency IDentification*. 1

SA *Slotted ALOHA*. 14

SDMA *Space Division Multiple Access*. 8

TDMA *Time Division Multiple Access*. 8

UI *User Interface*. 69

CAPÍTULO 1

INTRODUÇÃO

Existe uma crescente necessidade de organizar, identificar, rastrear e automatizar objetos. O barateamento de tecnologias e o aumento da capacidade computacional permitem que existam soluções que acompanhem essa crescente necessidade. A tecnologia RFID (*Radio Frequency IDentification* - Identificação por Radiofrequência) surgem como forte candidata para suprir essa necessidade e alterará o modo como realizamos essas tarefas.

1.1 MOTIVAÇÃO

Os sistemas RFID são um dos mais promissores para identificação automática de objetos utilizando radiofrequência. De modo geral, um sistema RFID é composto por leitores, etiquetas e um subsistema de processamento [1, 2]. O leitor possui um módulo de radiofrequência e uma unidade de controle, onde suas principais funções são ativar etiquetas, estruturar a sequência de comunicação com elas e transferir dados entre a aplicação e as mesmas. As etiquetas são anexadas a objetos de interesse do sistema e possuem um ID (IDentificador) e o subsistema de processamento pode ser um programa ou um banco de dados, dependendo da aplicação do sistema RFID.

As maiores vantagens dos sistemas RFID são que estes possuem a capacidade de se comunicar com as etiquetas sem que exista uma linha de visada direta entre as etiquetas e os leitores (a comunicação pode ser feita através de obstáculos), por consequência, não precisam que o código de identificação esteja alinhado ao leitor e a capacidade de armazenar dados numa etiqueta, além do seu ID. Graças a essas vantagens, os sistemas

RFID possuem inúmeras aplicações, como por exemplo, inventário [3], controle de acesso [4], localização de objetos [5], rastreamento de animais [6], controle de estacionamento [7], monitoramento de corredores em provas de atletismo [8].

Os sistemas RFID, apesar de suas vantagens, introduzem algumas dificuldades que precisam ser resolvidas. Dentre essas dificuldades, uma das mais críticas é o controle do acesso ao meio, pois quando duas ou mais etiquetas transmitem informações para o leitor simultaneamente, ocorre uma colisão entre os sinais, impedindo que o leitor identifique o que foi enviado. Esse problema é, de modo geral, o problema clássico de acesso múltiplo ao meio.

Contudo, não é possível portar diretamente as soluções existentes para o problema de acesso múltiplo ao meio em sistemas RFID, como soluções utilizadas para celulares e redes de computadores sem fio, pois estas necessitam que as duas partes envolvidas na comunicação (transmissor e receptor) cooperem entre si. Essa cooperação não existe em sistemas RFID, pois as etiquetas possuem limitações importantes de poder computacional, memória, custo de fabricação e consumo de energia. Estas restrições, por sua vez, geram a necessidade da criação e utilização de protocolos e algoritmos específicos para esse tipo de sistema [9, 1, 10].

No âmbito de RFID, existem, basicamente, dois tipos de colisão a serem resolvidas: colisões de leitor e colisões de etiquetas. Colisões de leitor ocorrem quando um leitor interfere com outros leitores [11, 12]. As colisões de etiquetas ocorrem quando duas ou mais etiquetas tentam transmitir para o leitor ao mesmo tempo.

Existem diversos protocolos anticolisão para o problema de colisões de etiquetas [13, 14, 15, 16, 17]. Dentre eles, o protocolo DFSA (*Dynamic Framed Slotted ALOHA*) [18, 19, 20, 21, 22, 23, 24] vem recebendo grande atenção da literatura. Neste protocolo, o leitor organiza o tempo em um ou mais *frames* onde cada *frame* está subdividido em *slots*. As etiquetas são requisitadas a transmitirem em um *slot* a cada *frame* até que sejam identificadas pelo leitor. O número total de *slots* de cada *frame* subsequente ao *frame*

inicial é calculado dinamicamente com base na estimativa da população de etiquetas que competiram por *slots* no *frame* precedente.

O desempenho do DFSA está intrinsecamente ligado à estimativa da quantidade de etiquetas que estão sendo submetidas ao processo de identificação que, por sua vez, está ligado ao tamanho do próximo *frame* que deverá ser utilizado. Desse modo, é necessário que a estimativa da população de etiquetas que estão sendo submetidas ao processo de identificação seja a mais precisa possível e que haja um modo de definir o tamanho do próximo *frame* baseado na estimativa obtida. Esses métodos de estimativa são conhecidos como estimadores ou algoritmos de estimativa. Note que a escolha do tamanho do próximo *frame* é de extrema importância, e deve ser escolhido de modo a minimizar o tempo necessário para identificar todas as etiquetas.

Dentre as principais propostas de estimadores para o DFSA, encontram-se o *Lower Bound* [25], o Schoute [26], o Vogt [25] e o Eom-Lee [18]. Todos eles definem algum modo de cálculo de tamanho de *frames*. As funções para o cálculo do tamanho do *frame* que os estimadores definem tentam minimizar a quantidade de *slots* total que será utilizada num processo de identificação, desse modo, reduzindo também o tempo de identificação necessário, pois cada tipo *slot* - vazio, bem sucedido e em colisão - possui a mesma duração. Entretanto, isso não ocorre quando utiliza-se a extensão *Early-End*, que garante ao DFSA a capacidade de detectar quando existe um *slot* onde não há transmissões por parte das etiquetas, ou seja, um *slot* vazio, e encerra esse *slot* prematuramente, alterando a duração deste tipo de *slot*.

Assim sendo, uma redução na quantidade total de *slots* pode não representar a melhor otimização para o tempo de identificação, pois, como os *slots* vazios passam a ter uma menor duração, estes devem ser preferidos ao invés dos *slots* em colisão. Desse modo, quando se utiliza o *Early-End*, é observado que a função para a escolha do tamanho do próximo *frame* fornecida pelos estimadores ignoram essa informação, perdendo um possível ponto de otimização.

1.2 OBJETIVOS

Esta dissertação visa propor uma função de cálculo de tamanho de *frames* relacionando a função de cálculo do estimador com um fator de ajuste que explora o impacto do uso da extensão *Early-End*, objetivando uma redução no tempo total de identificação de uma dada quantidade de etiquetas.

Para atingir o objetivo proposto, é necessário que os seguintes objetivos específicos sejam atingidos:

- Estudar os protocolos anticolisão para RFID, em especial o DFSA e suas extensões;
- Estudar e analisar os principais estimadores de tamanho de *frame* para o DFSA;
- Criar um simulador capaz de executar o DFSA utilizando os principais estimadores propostos na literatura;
- Definir uma nova função de cálculo de tamanho de *frame* que considere o impacto da extensão *Early-End*;
- Analisar o desempenho do DFSA utilizando a nova função proposta, comparando com os estimadores estudados.

1.3 ORGANIZAÇÃO

Esta dissertação segue a seguinte organização: o Capítulo 2 apresenta os conceitos básicos de RFID e mostra os tipos de protocolos anticolisão utilizados no problema de colisão de etiquetas. O Capítulo 3 aprofunda os protocolos baseados em ALOHA, incluindo suas possíveis extensões. O Capítulo 4 mostra o funcionamento dos principais estimadores, quais as funções que os estimadores utilizam para relacionar a estimativa do número de etiquetas com o tamanho do próximo *frame*, a modelagem do canal utilizada para as

avaliações de desempenho e seus desempenhos comparados entre si. No Capítulo 5 é apresentada a função proposta para o cálculo do tamanho do *frame*, como determinar o parâmetro da função, os resultados obtidos com a aplicação da função proposta e seu impacto no tempo de identificação. Por fim, no Capítulo 6, são apresentadas considerações finais sobre a proposta, suas contribuições e os trabalhos futuros.

CAPÍTULO 2

RFID

Este capítulo apresenta os conceitos básicos de RFID e os meios conhecidos de como resolver problemas de acesso múltiplo ao meio em redes de computadores. Também é introduzido um dos tipos de protocolos que se utiliza para resolver problemas de acesso ao meio em sistemas RFID, conhecidos como protocolos baseados em árvore.

2.1 ARQUITETURA

Sistemas RFID são um dos mais promissores para a identificação de objetos através de sinais de radiofrequência. Esses sistemas possuem diversas características desejáveis em sistemas de identificação, como por exemplo, a não necessidade de linha de visada com o objeto a ser identificado, capacidade de armazenar informações extras sobre o objeto em questão, localização do objeto, etc [27]. Esses sistemas, de modo geral, possuem três componentes [1, 2]:

- Etiquetas: As etiquetas são adicionadas ao(s) objeto(s) de interesse e possuem um identificador (ID), que é uma sequência de *bits*, que pode ou não ser único;
- Leitores: São equipamentos constituídos, em geral, de um módulo de radiofrequência e uma unidade de controle, onde suas principais funções são: ativar etiquetas, estruturar a sequência de comunicação com as etiquetas e transferir dados entre a aplicação e as etiquetas;

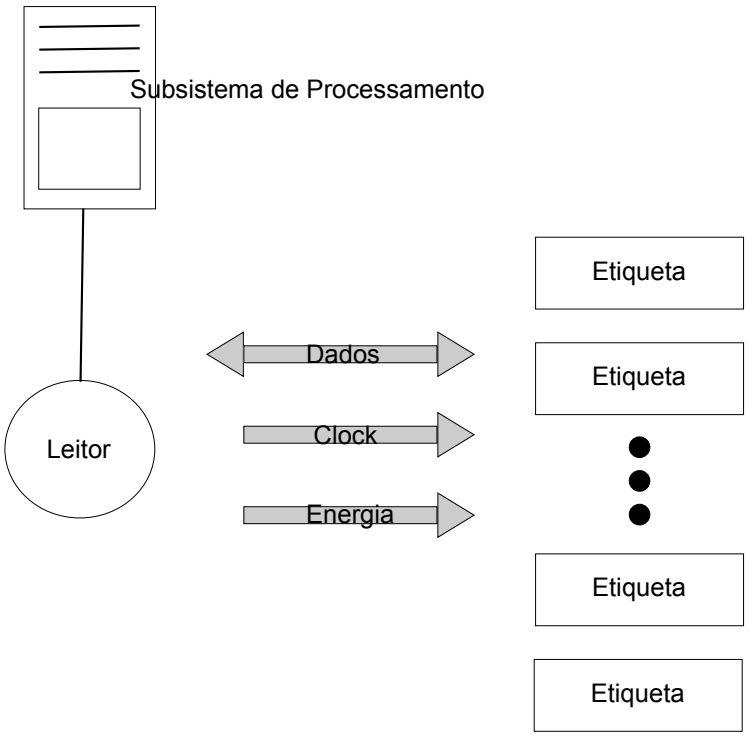


Figura 2.1: Exemplo de um sistema RFID.

- Subsistema de processamento: É um programa específico ou um banco de dados, dependendo da aplicação a ser dada para o sistema RFID.

Note que o subsistema de processamento pode estar embutido no leitor. Um exemplo dos componentes pode ser visto na Figura 2.1.

As etiquetas podem ser ainda subdivididas em três tipos: ativas, passivas e semiativas ou semi-passivas [28, 29]. Etiquetas ativas ou semi-passivas possuem uma fonte de energia interna, de modo geral, uma bateria. Essas etiquetas diferem entre si como a energia da bateria é utilizada, uma vez que, a etiqueta ativa utiliza essa bateria para todas as suas necessidades de energia, enquanto as etiquetas semi-passivas não utilizam sua fonte própria de energia para comunicação, sendo esta conseguida através de indução eletromagnética. Etiquetas passivas não possuem qualquer fonte de energia interna, logo, todas as suas necessidades de energia, isto é, energia para a comunicação e para o processamento, são extraídas da onda eletromagnética do leitor, através do processo de indução

eletromagnética.

É importante salientar que as etiquetas passivas são as mais desejadas e utilizadas, pois possuem um menor custo agregado de fabricação e manutenção, tornando estas o objeto de estudo desse trabalho. Deste modo, etiquetas passivas serão referenciadas apenas por etiquetas.

2.2 TIPOS DE COLISÃO

Graças à utilização de um meio de comunicação por radiofrequência, interferências no processo de comunicação são esperadas. Essas interferências são chamadas de colisões. Colisões geram desperdício de largura de banda, desperdício de energia e aumento do tempo de identificação, podendo facilmente inviabilizar um sistema RFID [10]. Quando consideramos o sistema RFID interferindo nele mesmo, temos dois tipos de colisões que precisam ser resolvidas [1]:

- Colisão entre leitores: Quando um ou mais leitores interferem entre si;
- Colisão entre etiquetas: Quando duas ou mais etiquetas tentam transmitir para um mesmo leitor ao mesmo tempo.

Esses dois tipos de colisão precisam ser resolvidos para que um sistema RFID possa funcionar e, de modo geral, consistem num problema de acesso múltiplo ao meio. Apesar de colisões entre leitores serem importantes para o funcionamento de sistemas RFID [9], esse trabalho se foca no problema de colisão entre etiquetas. Exemplos de protocolos que tratam o problema de colisão entre leitores podem ser encontrados em [12, 1, 11].

Conforme mencionado, soluções para o problema de colisão entre etiquetas, que a partir deste ponto fica conhecido apenas como problema de colisão, são soluções que se enquadram no problema de acesso múltiplo ao meio. Essas soluções serão apresentadas na Seção 2.2.1.

2.2.1 Acesso Múltiplo ao Meio

As soluções para o problema de acesso múltiplo ao meio se dividem em, basicamente, quatro grupos: SDMA (*Space Division Multiple Access* - Acesso Múltiplo por Divisão de Espaço) , FDMA (*Frequency Division Multiple Access* - Acesso Múltiplo por Divisão de Frequência), CDMA (*Code Division Multiple Access* - Acesso Múltiplo por Divisão de Código) e TDMA (*Time Division Multiple Access* - Acesso Múltiplo por Divisão de Tempo).

O grupo SDMA [30] consiste em atribuir faixas de frequências diferentes para regiões adjacentes. Esta técnica é altamente utilizada por redes de celulares. Contudo a implementação dessa técnica para sistemas RFID é complexa, pois, em geral, existe um custo de implementação relativamente alto e um complicado sistema de antenas [1].

Para técnicas do tipo FDMA [1], divide-se um espectro de frequência em canais onde cada nó transmissor é alocado para um canal. Note que banda de transmissão é dividida, ou seja, no caso de haver N divisões, existem N canais que utilizam $1/N$ de banda. A utilização de FDMA para RFID é limitada, pois cada país possui sua própria regulação de frequências e a utilização de FDMA gera um aumento no custo de produção das etiquetas.

Quando utiliza-se CDMA [31], existe uma multiplexação de vários canais que utilizam um código de espalhamento espectral, onde cada canal é identificado por um código. Esse tipo de solução possui problemas para sistemas RFID, pois a multiplexação por código é computacionalmente cara, o que dificulta a implementação da solução nas etiquetas.

No modelo TDMA [32], o canal é dividido no tempo e cada nó transmite em um determinado espaço de tempo. Para aplicações RFID esta é a solução mais utilizada, pois possui baixo custo computacional para as etiquetas. Quando se utiliza TDMA, o período de transmissão disponibilizado para etiquetas pode ser classificado de três modos: vazio, quando não existe nenhuma etiqueta transmitindo; bem sucedido, quando apenas uma etiqueta esta transmitindo no dado intervalo de tempo e colisão, que ocorre quando

duas ou mais etiquetas transmitem num mesmo intervalo de tempo.

Inseridos nas soluções que utilizam TDMA, podemos subdividir, de modo geral, os protocolos anticolisão em três tipos: protocolos baseados em árvore, protocolos baseados em ALOHA e protocolos híbridos.

2.3 PROTOCOLOS BASEADOS EM ÁRVORE

Protocolos baseados em árvore são determinísticos e visam separar as etiquetas que colidiram em grupos, utilizando um filtro qualquer, como por exemplo, um número aleatório ou partes do ID da etiqueta [33]. O processo de identificação pode ser representado por uma estrutura em árvore, geralmente binária, onde os nós representam a situação do grupo de etiquetas (bem sucedida, vazio e colisão) e as folhas representam grupos unitários de etiquetas que, por sua vez, serão identificados com sucesso. Um exemplo dos protocolos baseados em árvore é o protocolo QT [34].

No protocolo QT o filtro utilizado é o ID das etiquetas, mais especificamente, partes do identificador. Como o identificador das etiquetas é binário e possui uma quantidade de *bits* predeterminada, o leitor pergunta as etiquetas se estas possuem um ID que se inicia com um dado prefixo, caso alguma etiqueta possua o prefixo igual ao perguntado, ela deve transmitir informações para o leitor, caso contrário não deve transmitir nenhum dado. Havendo transmissão por parte das etiquetas, podem acontecer duas condições: apenas uma etiqueta transmite, ou seja, a etiqueta é identificada com sucesso ou duas ou mais etiquetas transmitem ao mesmo tempo, gerando uma colisão.

Se houver colisão, o leitor concatena o prefixo que gerou a colisão de 0 e 1, gerando dois novos prefixos, que são enviados as etiquetas e assim sucessivamente até que todas as etiquetas tenham sido identificadas. De modo genérico, assuma que um prefixo p_0 binário qualquer gerou uma colisão, utilizando o lógica apresentada, os prefixos $p_1 = p_0 0$ e $p_2 = p_0 1$ serão perguntados em seguida, após essa rodada, houve nova colisão no prefixo

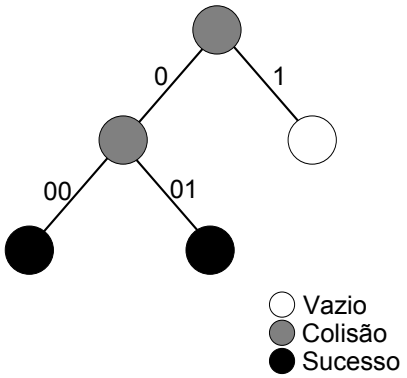


Figura 2.2: Exemplo do funcionamento do protocolo QT.

p_2, que por sua vez, irá gerar dois novos prefixos $p_3 = p_20 = p_010$ e $p_4 = p_21 = p_011$ e assim sucessivamente até que todas as etiquetas tenham sido identificadas.

Um exemplo do funcionamento desse protocolo pode ser visto na Figura 2.2. Suponha que existam duas etiquetas com os seguintes ID: 00 e 01. O leitor sempre inicia o processo de identificação com os prefixos mais básicos, isto é, o prefixo 0 e o prefixo 1. Logo, o leitor perguntará se existem etiquetas que possuem um prefixo de ID igual a 0, o que acarretará em colisão, uma vez que as duas etiquetas possuem esse prefixo. Logo após, o leitor pergunta se existe alguma etiqueta com ID iniciando em 1, que por sua vez é vazio, já que nenhuma etiqueta possui um prefixo iniciado em 1. Como existiu uma colisão para o prefixo 0, este prefixo é concatenado com 0 e 1, gerando dois novos prefixos - 00 e 01 - que por sua vez são, em ordem, perguntados às etiquetas, onde o leitor consegue identificá-las com sucesso.

Protocolos baseados em ALOHA são probabilísticos e serão vistos no Capítulo 3. Protocolos híbridos buscam utilizar partes dos protocolos em árvore e partes dos protocolos baseados em ALOHA para melhorar o processo de identificação. Exemplos de protocolos híbridos podem ser encontrados em [35, 36, 37].

2.4 RESUMO

Sistemas RFID possuem propriedades altamente desejáveis quando se trata de identificação automatizada de objetos. Entretanto, esses sistemas também possuem uma quantidade de requisitos próprios que precisam ser considerados, impostos principalmente pelas etiquetas devido à sua simplicidade e à sua não capacidade de cooperar umas com as outras para resolver os problemas da rede. Um dos mais importantes problemas em RFID é o problema de acesso múltiplo ao meio.

Soluções existentes atualmente possuem uma complexidade proibitiva quando trata-se de sistemas RFID, uma vez que sistemas RFID possuem restrições de *hardware* e custo de fabricação. Assim sendo, as soluções utilizadas para outras redes, como redes de computadores e redes celulares precisam ser adaptadas ou redesenhadas para que possam ser aplicadas no âmbito de RFID. Uma dessas soluções são os protocolos baseados em árvore. Protocolos baseado em árvore são determinísticos e utilizam algum modo de separar as etiquetas em grupos, criando uma "árvore" de colisões.

CAPÍTULO 3

PROTOCOLOS BASEADOS EM ALOHA

Este capítulo apresenta algumas soluções baseadas no protocolo ALOHA para o problema do acesso múltiplo ao meio em ambientes RFID. Dentre as versões apresentadas, será mostrada, sua versão original, as evoluções *Slotted ALOHA* e *Frame Slotted ALOHA* e as extensões *muting*, *Early-End*.

3.1 ALOHA ORIGINAL

Conforme mencionado anteriormente, protocolos baseados em ALOHA são protocolos probabilísticos que utilizam técnicas baseadas no TDMA para controlar o acesso ao meio. Esses protocolos são baseados no protocolo ALOHA original, utilizado na década de 70 para resolver o problema de alocação de canais em redes de computadores [38].

O ALOHA funciona da seguinte forma. Quando dois ou mais nós transmitem ao mesmo tempo, ocorre uma colisão e os nós que participaram da colisão utilizam um processo de *random backoff*, isto é, após um intervalo de tempo aleatório (que pode ou não ser igual para os nós) os nós transmitem novamente até que estes não possuam mais dados para transmitir. Este método de ALOHA pode ser adaptado com mudanças mínimas, tornando-o o método mais básico de funcionamento do ALOHA para sistemas RFID [2].

Em sistemas RFID, o ALOHA fica adaptado do seguinte modo. As etiquetas iniciam sua transmissão assim que possuem dados para serem enviados, de modo geral, isso ocorre quando as etiquetas entram no campo de ação do leitor. Após entrar no campo de leitor,

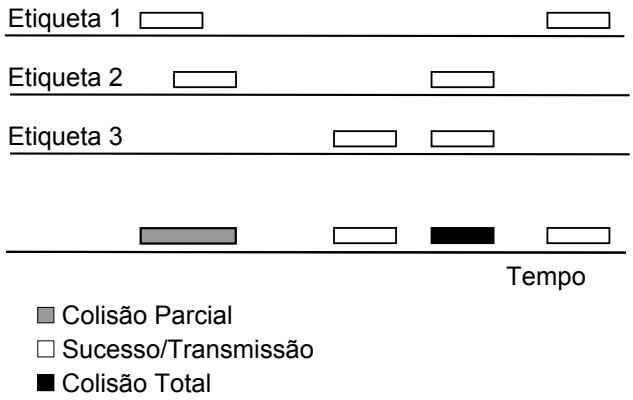


Figura 3.1: Exemplo de execução do ALOHA Puro.

as etiquetas escolhem um número aleatório que define quando as etiquetas vão transmitir. Após a transmissão, a etiqueta realiza o processo de *random backoff*, mesmo quando não há colisões, e isso se repete até que todas as etiquetas sejam identificadas.

Nessa variação, o tempo de transmissão pode ser classificado em tipos distintos. Vazio, quando nenhuma etiqueta está transmitindo; bem sucedido, quando apenas uma etiqueta está transmitindo e colisão, quando duas ou mais etiquetas transmitem. Note que, as colisões podem ser de dois tipos, parciais ou totais e colisões parciais podem ter qualquer duração variável. Um exemplo do comportamento do canal de transmissão pode ser visto na Figura 3.1.

3.1.1 Extensão Switch off/Muting

Um dos problemas mais claros do ALOHA puro é que as etiquetas que foram identificadas corretamente continuam competindo por tempo de transmissão, independente de ter havido colisão. Desse modo, a extensão *muting* pode ser utilizada para eliminar esse problema.

A extensão *muting* [2, 10] consiste em, uma vez que a etiqueta seja identificada com sucesso, o leitor envie um comando especial para a etiqueta, fazendo com que esta não tente selecionar outro tempo para transmissão. Isso reduz o número de colisões, pois

a quantidade de etiquetas tentando transmitir se reduz à medida que as etiquetas vão sendo identificadas. Um exemplo dessa extensão pode ser visto na Figura 3.2.

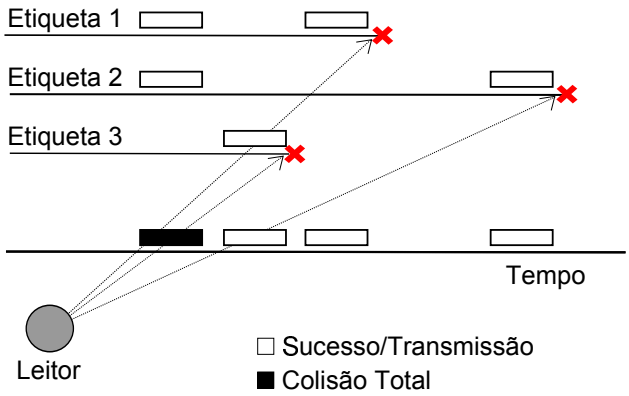


Figura 3.2: Exemplo da aplicação da extensão *Muting/Switch-off*.

3.2 SLOTTED ALOHA

No ALOHA original, os nós podem transmitir em qualquer instante de tempo [39]. A evolução desse conceito, isto é, definir espaços de tempo para que a transmissão ocorra é conhecido como *Slotted* ALOHA (SA) [2, 10, 40].

No SA, o tempo é divido em intervalos discretos bem definidos, conhecidos como *slots*, onde as etiquetas só podem transmitir no começo de um *slot* e devem acabar sua transmissão ao final deste, como pode ser visto na Figura 3.3. Esta evolução do ALOHA puro elimina colisões parciais, entretanto, a utilização dessa técnica necessita que exista uma sincronização para que ocorra o funcionamento correto do protocolo, o que, por sua vez, adiciona complexidade ao sistema.

Assim como a extensão proposta para o ALOHA puro pode ser usada no SA, o próprio SA também possui suas extensões, como pode ser visto na Seção 3.2.1.

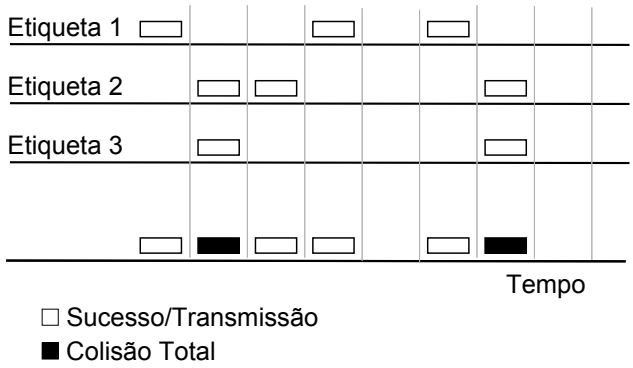


Figura 3.3: Exemplo da execução do *Slotted* ALOHA.

3.2.1 Extensão Early-End

Com o conceito de *slots*, o leitor pode facilmente saber se existe transmissão num determinado *slot* ou não. De posse desse conhecimento, o leitor pode encerrar prematuramente o *slot*, visando um ganho no tempo de identificação das etiquetas. Desse modo, a extensão *Early-End* consiste em o leitor detectar rapidamente se um *slot* é vazio e encerrá-lo antecipadamente. Essa extensão permite uma redução no tempo de identificação total e uma otimização do canal de transmissão [40]. Um exemplo do funcionamento dessa extensão pode ser visto na Figura 3.4.

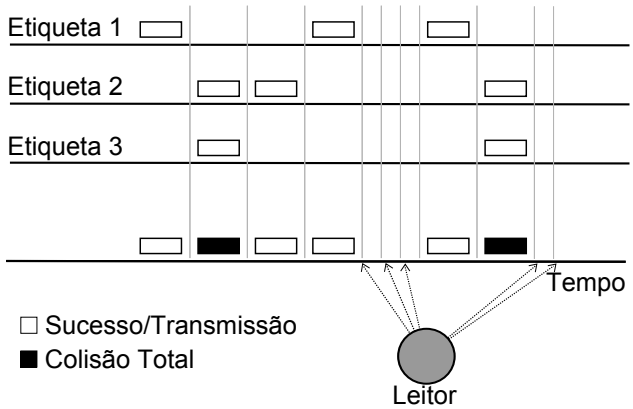


Figura 3.4: Exemplo da aplicação da extensão *Early-End*.

3.3 FRAME SLOTTED ALOHA

O *Frame Slotted* ALOHA (FSA) [2, 10], visa agrupar os *slots* em *frames*, de tamanho fixo, onde cada etiqueta pode transmitir apenas uma vez em cada *frame*, como pode ser visto na Figura 3.5. Com essa extensão se elimina o problema de uma etiqueta possuir uma frequência de resposta muito alta, poluindo o meio de comunicação e aumentando o número de colisões.

Além do procedimento de sincronização necessário, o FSA também introduz um novo problema. Qual o tamanho de *frame* deve ser utilizado, tendo em vista que, um *frame* muito pequeno gerará um grande número de colisões e um *frame* muito grande gera um excesso de *slots* vazios. O FSA propõe a utilização de um tamanho de *frame* fixo, que, logicamente, pode não ser ideal para todos os casos. Visando resolver este problema foi proposta a extensão a seguir. Note que todas as extensões disponíveis para o ALOHA Puro e o SA podem ser utilizadas no FSA.

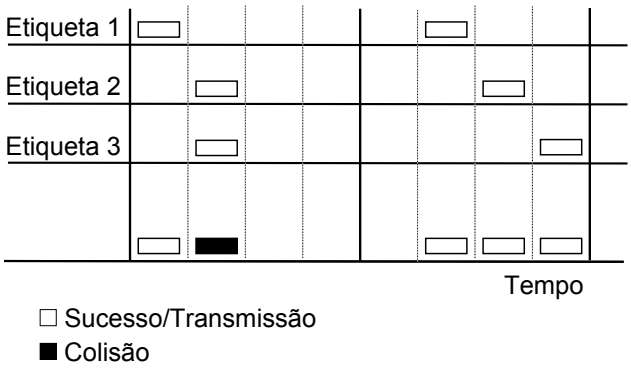


Figura 3.5: Exemplo da execução do *Frame Slotted* ALOHA.

3.3.1 Dynamic Frame Slotted ALOHA

A utilização de um método dinâmico para definir o tamanho do *frame* foi proposta, inicialmente, por Schoute [26] para um contexto geral de redes móveis.

No *Dynamic Frame Slotted* ALOHA (DFSA) [4], temos que o protocolo define o tama-

nho do próximo *frame* baseado em informações adquiridas do *frame* atual. Desse modo, existe a necessidade de métodos que utilizem essas informações para gerar o tamanho do próximo *frame*, que por sua vez, será transmitido para as etiquetas. Esses métodos são conhecidos como estimadores ou algoritmos de estimativa. Estimadores serão vistos mais a fundo no Capítulo 4. Note que, o tamanho do primeiro *frame* é definido previamente. Uma ilustração do DFSA pode ser vista na Figura 3.6.

É importante ressaltar que alguns trabalhos na literatura consideram que alterações nos estimadores geram nomes distintos para o DFSA. Contudo, o funcionamento do protocolo não se altera, apenas o modo como a estimativa é calculada. Considerando essa informação, não foi apreciado, neste trabalho, que a alteração do estimador gera novas variações do DFSA.

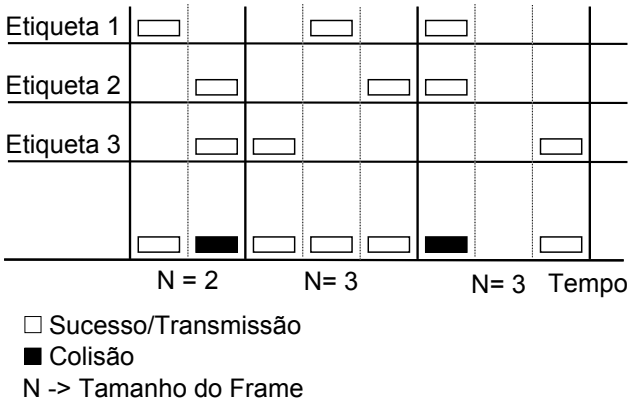


Figura 3.6: Exemplo da utilização da extensão DFSA.

3.4 RESUMO

Dentre as soluções disponíveis para resolver o problema de acesso múltiplo ao meio em RFID, uma das mais utilizadas são aquelas baseadas no protocolo ALOHA, utilizado nas redes de computadores. Esses protocolos são probabilísticos e utilizam técnicas de TDMA para controlar o acesso ao meio. Em particular, evoluções do protocolo dividem o meio em *slots*, onde as etiquetas devem escolher um desses *slots* para transmitir. A evolução

natural seguinte dos protocolos baseados em ALOHA foi agrupar esses *slots* em *frames* de tamanho fixo, de modo que uma etiqueta possa transmitir apenas uma vez num dado *frame*. Logo se percebeu que ter um tamanho de *frame* fixo era problemático, fazendo com que este tamanho de *frame* passasse a ser definido dinamicamente, gerando assim o DFSA e algoritmos conhecidos como estimadores para definir o tamanho do *frame*.

Dentre as extensões dos protocolos baseados em ALOHA, merecem destaque duas dessas. A primeira é a extensão *muting*, que permite ao leitor silenciar uma etiqueta quando esta é identificada corretamente, isso permite que etiquetas identificadas não mais disputem *slots*, reduzindo a quantidade de colisões. A segunda é a extensão *Early-End*, que habilita o leitor a encerrar um *slot* sem transmissão de modo prematuro, fazendo com que haja uma redução no tempo de identificação total e um melhor aproveitamento do canal, uma vez que não se perde tanto tempo quando não existem transmissões.

CAPÍTULO 4

ESTIMADORES

Este capítulo apresenta os algoritmos de estimativa *lower bound*, *schoute*, *vogt* e *eom-lee* utilizados no DFSA, como eles são obtidos e utilizados e uma comparação entre o desempenho desses algoritmos com objetivo analisar onde os algoritmos se sobressaem.

4.1 CONCEITOS

O desempenho máximo no DFSA, em termos de vazão, é obtido quando o tamanho do *frame* é igual à quantidade de etiquetas que é necessário se identificar [41]. Desse modo, para determinar o tamanho do próximo *frame*, é necessário que se conheça a quantidade de etiquetas a serem identificadas ou que, com base em informações do *frame* atual, se calcule a quantidade de etiquetas que estão sendo submetidas ao processo de identificação. Assumir que o sistema conhece o número de etiquetas a serem identificadas é infactível numa aplicação real, com isso, aplicações de RFID devem utilizar estimadores para "descobrir" o número de etiquetas.

Existem diversos estimadores, que aplicam variados conceitos, para estimar a quantidade de etiquetas no DFSA, como por exemplo, os trabalho encontrados em [20, 29, 42]. Dentre os diversos estimadores, foram analisados os seguintes: *Lower Bound* [25], Vogt [25], Schoute [26] e Eom-Lee [18]. Esses estimadores serão mostrados em detalhes nas seções abaixo.

Considere para todos os estimadores neste capítulo que s_v, s_s e s_c, são a quantidade de *slots* vazios, *slots* bem sucedidos e *slots* em colisão num *frame*, respectivamente. Considere

também que $\hat{f}$ é o tamanho do próximo *frame* e $\hat{n}$ é a quantidade de etiquetas estimadas.

4.2 LOWER BOUND

O estimador *Lower Bound* [25] utiliza a seguinte informação como ideia inicial: não pode haver menos de duas etiquetas envolvidas em uma colisão. Desse modo, a quantidade de etiquetas competindo num dado *frame* é igual à quantidade de *slots* bem sucedidos somada com o dobro da quantidade de *slots* em colisão. Assim sendo, o *Lower Bound* define a quantidade estimada de etiquetas e o tamanho do próximo *frame*, respectivamente, como:

$$\hat{n} = s_s + 2 \cdot s_c \ , \tag{4.1}$$

$$\hat{f} = 2 \cdot s_c \ . \tag{4.2}$$

Teoricamente, bons estimadores não deveriam ter erros de estimativa piores do que as do *Lower Bound*, pois os valores estimados por este último para a quantidade de etiquetas restantes são sempre os menores dentro de uma gama de possibilidades. Por causa disso, o estimador *Lower Bound* é comumente utilizado em avaliações de desempenho do DFSA para fins de comparação com a qualidade e impacto de estimação de outros estimadores.

4.3 SCHOUTE

O algoritmo de Schoute [26] computa o tamanho do próximo *frame* como sendo igual ao número total de *slots* em colisão no *frame* atual multiplicado por um fator igual a 2,39. Esse fator é o número médio esperado de etiquetas que transmitiram em cada *slot* em colisão no *frame* atual. Assim sendo, o método de Schoute define:

$$\hat{f} = 2,39 \cdot s_c \ . \tag{4.3}$$

A Equação (4.3) é obtida considerando um processo de chegadas do tipo Poisson, sendo uma aproximação sub-ótima para outras distribuições. Para se obter a estimativa $\hat{n}$, basta somar $\hat{f}$ com o número total de etiquetas identificadas com sucesso no *frame* atual, como pode ser visto a seguir,

$$\hat{n} = s_s + 2,39 \cdot s_c \ . \tag{4.4}$$

4.4 VOGT

O método de Vogt [25] considera que a alocação de transmissões de etiquetas em um *slot* é um problema de ocupação. Tal problema lida com a alocação aleatória de "bolas" em uma certa quantidade de "sacos", onde as "bolas" podem ser vistas como etiquetas e os "sacos" como *slots* num *frame* de interesse. Dado que existem L *slots* num *frame* e n etiquetas que competiram por *slots* nele, a quantidade r de etiquetas em um *slot* é binominalmente distribuída com parâmetros n e $1/L$ conforme representa a seguinte equação:

$$B_{n,\frac{1}{L}}(r) = \binom{n}{r} \left(\frac{1}{L}\right)^r \left(1 - \frac{1}{L}\right)^{n-r} \ . \tag{4.5}$$

Adicionalmente, em um *frame* com L *slots*, a quantidade esperada de *slots* contendo transmissões de r etiquetas é dada por:

$$a_r^{L,n} = LB_{n,\frac{1}{L}}(r) = L \binom{n}{r} \left(\frac{1}{L}\right)^r \left(1 - \frac{1}{L}\right)^{n-r} \ . \tag{4.6}$$

O método de Vogt é baseado na desigualdade de Chebyshev, a qual afirma que o

Tabela 4.1: Tamanho de *frames* para o método de Vogt.

$\hat{f}$	$\hat{n} \in [x, y]$
16	$[1, 9]$
32	$[10, 27]$
64	$[17, 56]$
128	$[51, 129]$
256	$[112, \infty]$

resultado de um experimento envolvendo uma variável aleatória X é, provavelmente, próximo ao valor esperado de X [25]. Utilizando esse conceito, Vogt propõe uma função de estimativa do número de etiquetas que busca a minimização da distância entre o vetor $< s_v, s_s, s_c >$ e seu correspondente contendo os valores esperados para s_v, s_s e s_c, onde o valor da quantidade estimada de etiquetas - $\hat{n}$ - é o valor que minimiza o resultado da função. Essa função de estimativa é mostrada pela Equação (4.7).

$$\varepsilon(L, s_v, s_s, s_c) = \min_{n} \left| \begin{pmatrix} a_0^{L,n} \\ a_1^{L,n} \\ a_{\geq 2}^{L,n} \end{pmatrix} - \begin{pmatrix} s_v \\ s_s \\ s_c \end{pmatrix} \right| \quad . \tag{4.7}$$

O estimador de Vogt também define uma função de cálculo do tamanho do próximo *frame* com base no número estimado $\hat{n}$ de etiquetas. Os resultados possíveis para a função proposta são apresentados na Tabela 4.1. Caso $\hat{n} \in [17, 27]$ tanto $\hat{f} = 32$ quanto $\hat{f} = 64$ são escolhas adequadas.

É importante ressaltar que, diferentemente de outras propostas apresentadas, o estimador de Vogt foi desenvolvido considerando-se limitações do sistema RFID *I-Code*. Neste sistema, o tamanho dos *frames* está limitado a uma potência de 2 com tamanho máximo possível igual a 256 *slots*. A maioria das pesquisas atuais envolvendo estimadores para o DFSA, utilizam o método de Vogt sem tais limitações. Para isso, o tamanho do próximo *frame* no DFSA é calculado simplesmente como:

$$\hat{f} = \hat{n} - s_s \quad . \tag{4.8}$$

É importante enfatizar que, neste trabalho, foi considerada a utilização da Equação (4.8) para calcular o tamanho do próximo *frame* ao invés da Tabela 4.1, uma vez que os outros estimadores não estão limitados a um valor múltiplo de potência de 2 ou com tamanho máximo possível de 256.

4.4.1 Análise da função ε de Vogt

A proposta de Vogt se baseia em realizar uma busca no domínio da função ε a procura do valor n que minimiza o resultado dessa função. Desse modo, é necessário que seja feito um estudo para analisar o comportamento dessa curva, visando encontrar o melhor modo de executar a busca nessa função.

Tabela 4.2: Casos propostos para análise da função ε de Vogt.

Casos	Descrição	Representação Matemática
Caso 1	Todos os *slots* em colisão	$s_c = L$
Caso 2	Todos os *slots* vazios	$s_v = L$
Caso 3	Todos os *slots* bem-sucedidos	$s_s = L$
Caso 4	Maior quantidade de *slots* em colisão	$s_c \geq s_v, s_s$
Caso 5	Maior quantidade de *slots* vazios	$s_v \geq s_c, s_s$
Caso 6	Maior quantidade de *slots* bem-sucedidos	$s_s \geq s_v, s_c$
Caso 7	Quantidade de *slots* em colisão igual a *slots* vazio	$s_c = s_v$
Caso 8	Quantidade de *slots* em colisão igual a *slots* bem-sucedidos	$s_c = s_s$
Caso 9	Quantidade de *slots* bem-sucedidos igual a *slots* vazio	$s_s = s_v$

Para realizar essa análise, neste trabalho foram propostos nove casos distintos: quando todos os *slots* estão em colisão, quando todos os *slots* são vazios, quando todos os *slots* são bem-sucedidos, quando existe um maior número de *slots* em colisão, um maior número de *slots* vazios, um maior número de *slots* bem-sucedidos, quando o número de *slots* em colisão é igual ao número de *slots* vazios, quando o número de *slots* em colisão é igual

ao número de *slots* bem-sucedidos e, por fim, quando o número de *slots* vazios é igual ao número de *slots* bem-sucedidos. Os casos podem ser vistos na Tabela 4.2.

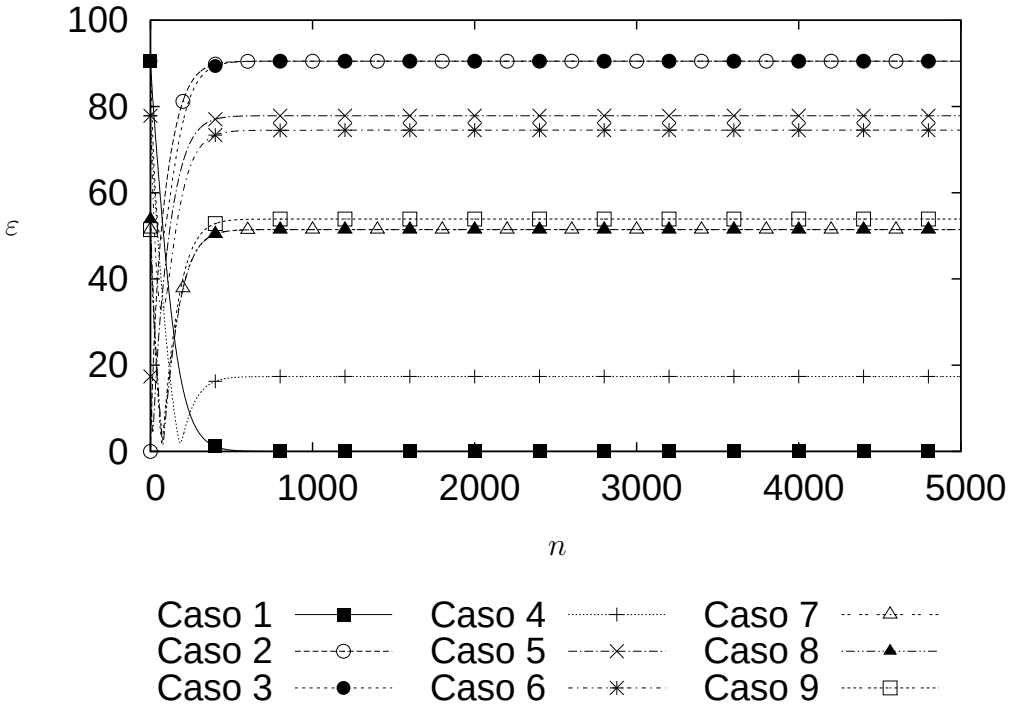
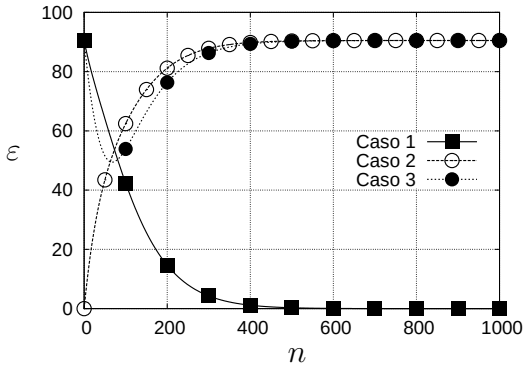


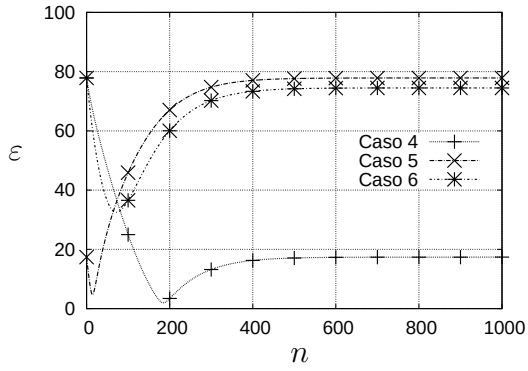
Figura 4.1: Análise de todos os casos até 5000 etiquetas.

Foi analisada a função de Vogt até uma quantidade de 5.000 etiquetas, como pode ser observado na Figura 4.1. Esse gráfico mostra que, para todos os casos, após a quantidade de aproximadamente 1.000 etiquetas o comportamento das curvas é praticamente constante. Isso mostra que é possível achar um mínimo local para todos os casos propostos. Entretanto, é necessária uma análise mais profunda dos casos para uma quantidade menor do que 1.000 etiquetas. Desse modo, os casos foram agrupados de três em três e ampliados para o intervalo $[0, 1000]$.

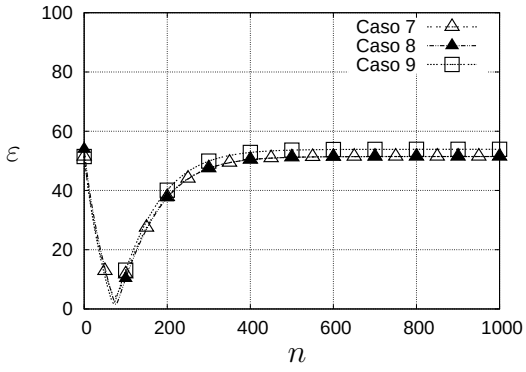
É possível observar que o comportamento dos Casos 3, 4, 5, 6, 7, 8 e 9 (Figuras 4.2(a), 4.2(b) e 4.2(c)) é similar, ou seja, as curvas possuem um mínimo local facilmente detectável. Entretanto, os Casos 1 e 2 (Figura 4.2(a)) possuem comportamentos diferenciados e necessitam que sejam avaliados individualmente.



(a) Casos 1, 2 e 3.



(b) Casos 4, 5 e 6.



(c) Casos 7, 8 e 9.

Figura 4.2: Casos para o intervalo $[0, 1000]$ etiquetas.

O Caso 1, diferentemente dos outros casos, não se torna constante com o passar do tempo, mas sim decai lentamente tendendo a zero quando a quantidade de etiquetas tende a infinito (a prova matemática pode ser vista no Apêndice B). Contudo, os sistemas computacionais modernos possuem um limite no valor que pode ser representado, logo, quando ocorrer este caso, um algoritmo de busca exaustiva forçará a busca até que o resultado da função chegue a zero ou que ocorra um estouro dos valores de quantidade de etiquetas. Desse modo, foi necessário adaptar o algoritmo para que não ocorra casos onde a simulação fique em *loop* infinito.

A solução utilizada foi truncar o valor da função quando ocorrer este caso. Quando o Caso 1 é detectado, utiliza-se apenas as 6 primeiras casas decimais do resultado da função, que por sua vez é conseguido através de uma multiplicação por 10^6, seguido

de um arredondamento para baixo (função *ceil()* em C++) e o resultado dessas duas operações é, por sua vez, divido por 10^6.

No Caso 2, nota-se que a função possui seu mínimo quando a quantidade de etiquetas é igual a zero e o resultado da função também é igual a zero, o que nunca deveria acontecer com um estimador utilizado para o protocolo DFSA, pois se temos um tamanho de *frame* igual a zero, é impossível que as etiquetas definam qual sua posição de transmissão. Como não se utilizou, nesse trabalho, a tabela de tamanhos de *frame* proposta por Vogt, para evitar o problema mencionado, se colocou como condição que os tamanhos dos *frames* gerados por esse estimador são sempre maiores ou iguais a dois.

4.5 EOM-LEE

O método Eom-Lee [18] propõe o uso de um algoritmo iterativo para se estimar a quantidade de etiquetas competindo por *slots* em um *frame* e o tamanho $\hat{f}$ de seu próximo *frame*. Primeiramente, Eom-Lee define L como sendo o tamanho do *frame* que será analisado para se estimar o tamanho do *frame* imediatamente posterior. O valor de L é assumido ser igual ao número estimado de etiquetas que competiram no *frame* multiplicado por um fator β que deverá ser determinado. Assim sendo, o valor de L pode ser representado por:

$$L = \beta \cdot \hat{n} \ . \tag{4.9}$$

Também assume-se que quando uma colisão ocorre, o número de etiquetas competindo em um *slot* em colisão é igual a γ. Considerando também que o tamanho do próximo *frame* pode ser dado pela quantidade de etiquetas estimada que estão se submetendo ao processo de identificação no *frame* atual subtraido da quantidade de etiquetas identificadas com sucesso no *frame*, temos que o valor de $\hat{f}$ pode ser encontrado através da seguinte equação:

$$\hat{f} = \hat{n} - s_s = \gamma \cdot s_c \ . \tag{4.10}$$

Assim como o algoritmo de Vogt, este estimador considera que a ocupação dos *slots* pode ser aproximada por uma distribuição binomial de acordo com a Equação (4.5). Entretanto, assumindo que L é grande o suficiente, a Equação (4.5) pode ser aproximada por uma distribuição de Poisson com média $\frac{\hat{n}}{L}$. Desse modo, resolvendo a Equação (4.5) para $r = 0$, $r = 1$ e $r \geq 2$ o lado esquerdo e o direito da Equação (4.10) podem ser aproximados, respectivamente, por:

$$\hat{n} - s_s \approx \hat{n} \left(1 - e^{-\frac{1}{\beta}}\right), \tag{4.11}$$

$$\gamma \cdot s_c \approx \gamma \cdot \beta\hat{n} \left(1 - \left(1 + \frac{1}{\beta}\right) e^{-\frac{1}{\beta}}\right). \tag{4.12}$$

Resolvendo as Equações (4.10), (4.11), (4.12) para γ, obtemos

$$\gamma = \frac{1 - e^{-\frac{1}{\beta}}}{\beta \left(1 - \left(1 + \frac{1}{\beta}\right) e^{-\frac{1}{\beta}}\right)} \ . \tag{4.13}$$

Encontrar uma solução fechada para determinar os valores de γ e β a partir da Equação (4.13) é um desafio. Por causa disso, o método Eom-Lee utiliza um algoritmo iterativo para encontrar tais valores. Considere γ_k e β_k sendo, respectivamente, uma aproximação para o valor de γ e de β na k-ésima iteração do algoritmo. Essas aproximações são obtidas de acordo com as seguintes Equações:

$$\beta_k = \frac{L}{\gamma_{k-1} \cdot s_c + s_s} \ , \tag{4.14}$$

$$\gamma_k = \frac{1 - e^{-\frac{1}{\beta_k}}}{\beta_k \left(1 - \left(1 + \frac{1}{\beta_k}\right) e^{-\frac{1}{\beta_k}}\right)} \ . \tag{4.15}$$

No primeiro passo, considera-se $\beta_1 = \infty$ e $\gamma_1 = 2$ e em cada passo, o k seguinte se determina uma nova aproximação para β e γ com o auxílio das Equações (4.14) e (4.15), respectivamente. Quando $|\gamma_{k^*-1} - \gamma_{k^*}|$ for menor que um limiar pré-definido $\epsilon_{threshold}$, o processo iterativo é interrompido. γ_{k^*-1} e γ_{k^*} representam, respectivamente, a aproximação anterior e atual para o valor de γ. A partir de então, o tamanho $\hat{f}$ do próximo *frame* e a quantidade estimada $\hat{n}$ de etiquetas são obtidos, respectivamente, pelas Equações (4.16) e (4.17), onde β_{k^*} é a aproximação mais recente para o valor de β.

$$\hat{f} = \gamma_{k^*} \cdot s_c \ . \tag{4.16}$$

$$\hat{n} = \frac{\hat{f}}{\beta_{k^*}} \ . \tag{4.17}$$

4.6 AVALIAÇÃO E ANÁLISE DOS ESTIMADORES

Devido à existência de vários estimadores, se torna necessário comparar o desempenho dos múltiplos algoritmos visando entender em quais situações é melhor utilizar um estimador em relação aos outros. Considerando isto, apresenta-se, nesta seção, uma avaliação de desempenho dos estimadores mostrados neste capítulo.

As métricas de desempenho estudadas foram: o número total de *slots* utilizados no processo de identificação, o tempo total para a identificação de todas as etiquetas, o número total de *slots* vazios, o número total de *slots* em colisão e a média do erro absoluto de estimação por *frame* ao longo do processo de identificação. O erro absoluto de estimação é definido como o módulo da diferença entre o número real e o número estimado de etiquetas em um *frame* de interesse.

O cenário de simulação possui um leitor e diversas etiquetas, onde uma vez iniciado o processo de identificação este só é encerrado quando todas as etiquetas são identificadas. Nenhuma nova etiqueta é inserida enquanto o processo de identificação está em execução.

A fim de se estudar apenas o impacto dos estimadores no desempenho do processo de identificação, considera-se um canal de comunicação livre de erros.

Para cada estimador, os resultados apresentados foram obtidos a partir da média dos resultados de 2.000 simulações.

Como parâmetros de simulação, foi utilizado o protocolo DFSA com as extensões *Early-End* e *Muting* habilitadas, tamanho do *frame* inicial de 64 *slots*, taxa de transmissão de 40 kbps, tanto para *downstream* como para *upstream* e as etiquetas possuem um identificador de 128 *bits*. Particularmente para o estimador Eom-Lee, adota-se o valor de $\epsilon_{threshold}$ igual a 0,001, conforme os autores originais sugeriram.

Para a execução das simulações, foi desenvolvido um simulador voltado para RFID, em C++, multiplataforma, capaz de simular diversas condições para sistemas RFID, dentre elas, diferentes quantidades de etiquetas, algoritmos de estimativa, etc. A opção de desenvolver um simulador foi feita por que a utilização de um simulador genérico de redes de computadores possuiria um alto custo para ser adaptado para sistemas RFID e as soluções encontradas de simuladores RFID foram feitas para casos específicos e não possuem o suporte necessário para que estes fossem alterados para os propósitos desta dissertação. Vale salientar que o simulador foi validado através de comparações com os resultados obtidos em outros artigos produzidos pela comunidade científica que avaliaram o DFSA e/ou os estimadores estudados nesta dissertação.

4.7 CANAL DE COMUNICAÇÃO

Neste trabalho, o canal de comunicação foi modelado a partir de dados fornecidos pela especificação *EPCglobal Class-1 Gen-2* [43]. Essa especificação define as temporizações para os diferentes tipos de *slots* no canal de comunicação e as regras de comunicação entre etiquetas e leitores. O leitor deve se referir à especificação usada para maiores informações sobre como calcular os tempos T_1 a T_5 explicitados nesta seção.

Considere a simbologia T =>R como sendo a representação de uma comunicação de etiqueta para leitor e a simbologia R =>T como sendo a representação de uma comunicação de leitor para etiqueta. Com base na especificação *EPCglobal Class-1 Gen-2*, modelou-se o canal de comunicação do seguinte modo: toda comunicação (*i.e.* comandos e respostas) é precedida de um período de sinalização, independente de seu sentido. Na comunicação R =>T, a sinalização pode ser de dois tipos: *preâmbulo* e *frame-sync*, onde o preâmbulo é utilizado apenas quando o leitor envia um comando de início de *frame* e o *frame-sync* é utilizado para qualquer outro tipo de comando. Na comunicação T =>R, a sinalização é feita através de um preâmbulo de tamanho igual a 6 *bits* conforme define a norma utilizada.

Imediatamente após o comando de início de *frame*, o leitor envia um comando de início de *slot*. A duração de um comando de início de um *slot* é igual a T_1. Este valor representa a soma da duração da sinalização com o tempo de transmissão e atraso de propagação. Após o envio desse comando, três situações podem acontecer: a ausência de respostas, respostas múltiplas ou uma única resposta. Um *slot* em colisão possui duração T_2. Essa duração é composta pelo tempo de propagação somado ao tempo de transmissão da mensagem das etiquetas e à duração do preâmbulo usado pelas mesmas. Caso o leitor transmita um comando de início de *slot*, se ele não começar a receber os *bits* iniciais de ao menos uma resposta, a duração do *slot* é encurtada. Um *slot* de tempo vazio, possui duração igual a $T_3 = max(RT_{Cal}, 10T_{pri})$, conforme parâmetros definidos na especificação [43]. A duração de um *slot* bem sucedido é igual a T_4 e considera a soma dos atrasos de transmissão e propagação da mensagem da etiqueta. O tempo T_5 representa o tempo necessário para identificar uma etiqueta e silenciá-la.

A Figura 4.3 ilustra os três casos de *slots* possíveis em um determinado *frame* e os intervalos de tempo mais importantes. Nessa ilustração, o leitor envia um comando de início de *slot* para as etiquetas das quais as respostas colidem em um *slot*. Em seguida, o leitor envia um novo comando de início de *slot* para o qual não há respostas. Por fim,

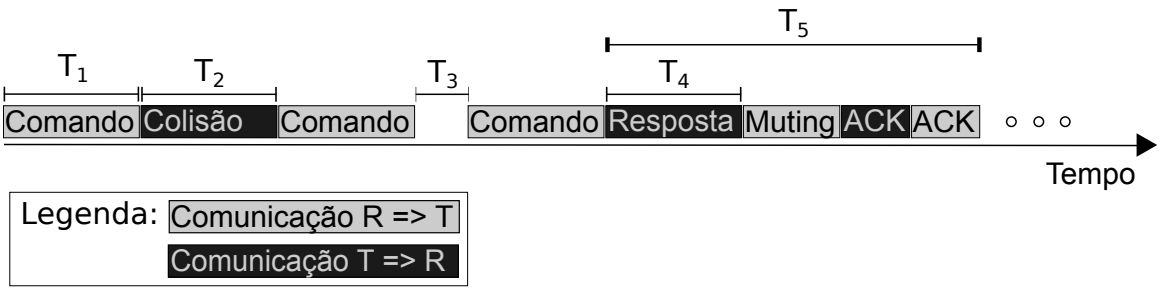


Figura 4.3: Modelo do canal de comunicação.

o leitor envia um comando de início de *slot* para o qual apenas uma etiqueta responde. Após a resposta da etiqueta, o leitor envia o comando *Muting*, a etiqueta responde com um ACK e o leitor confirma a recepção desse ACK com outro ACK. Nesse ponto, a etiqueta é considerada identificada. Note que os tempos $T_1, T_2, T_3, T_4 e T_5$ são constantes.

4.8 DESEMPENHO DOS ESTIMADORES

O desempenho encontrado dos estimadores são mostrados nas Figuras 4.4, 4.5, 4.6.

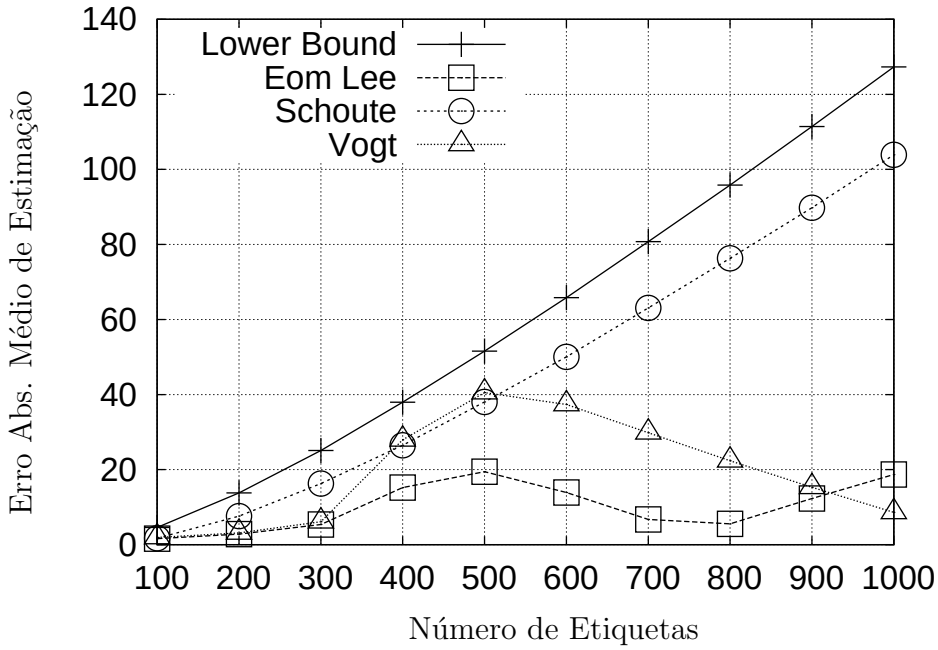


Figura 4.4: Desempenho dos estimadores considerando erro absoluto médio.

A Figura 4.4 mostra o erro absoluto médio de estimação para cada um dos estimadores quando o número total de etiquetas varia de 100 a 1000. Observa-se que o erro absoluto médio dos estimadores *Lower Bound* e Schoute possui um crescimento exponencial leve em função do aumento do número de etiquetas a serem identificadas. Dentre todos os estimadores avaliados, o Eom-Lee é o melhor de todos até pouco mais de 900 etiquetas. Em torno desse valor, o Vogt passa a ser melhor. É importante observar que no intervalo de 100 a 300 etiquetas, o erro absoluto médio do estimador Vogt é muito próximo ao observado pelo o estimador Eom-Lee. Já, entre 300 e 500 etiquetas, o erro absoluto médio do estimador Vogt cresce rapidamente. Contudo, observa-se que, a partir de 600 etiquetas, o erro absoluto médio do Vogt volta a se aproximar cada vez mais do erro observado com o uso do estimador Eom-Lee.

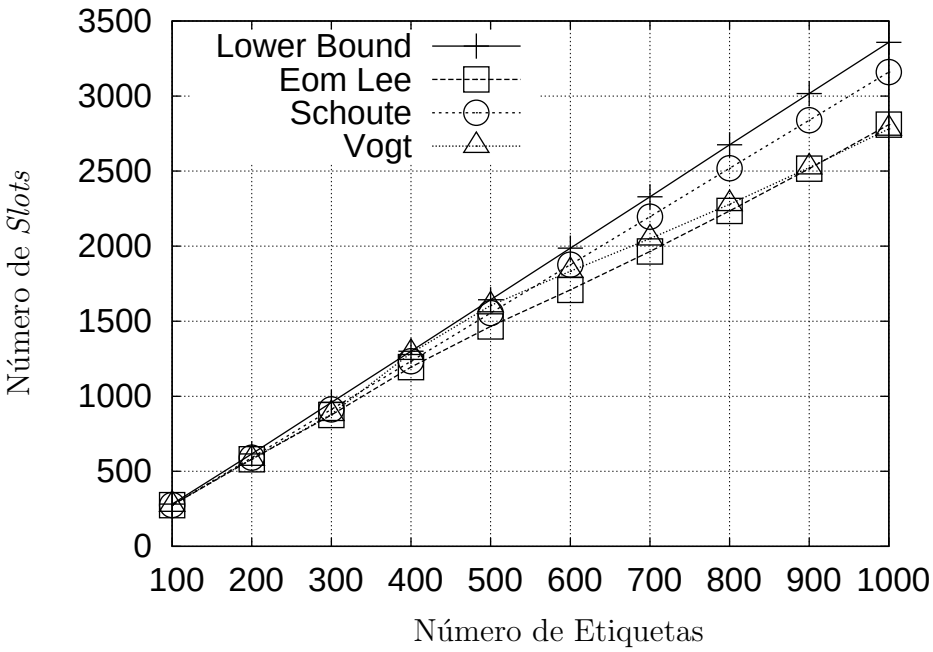
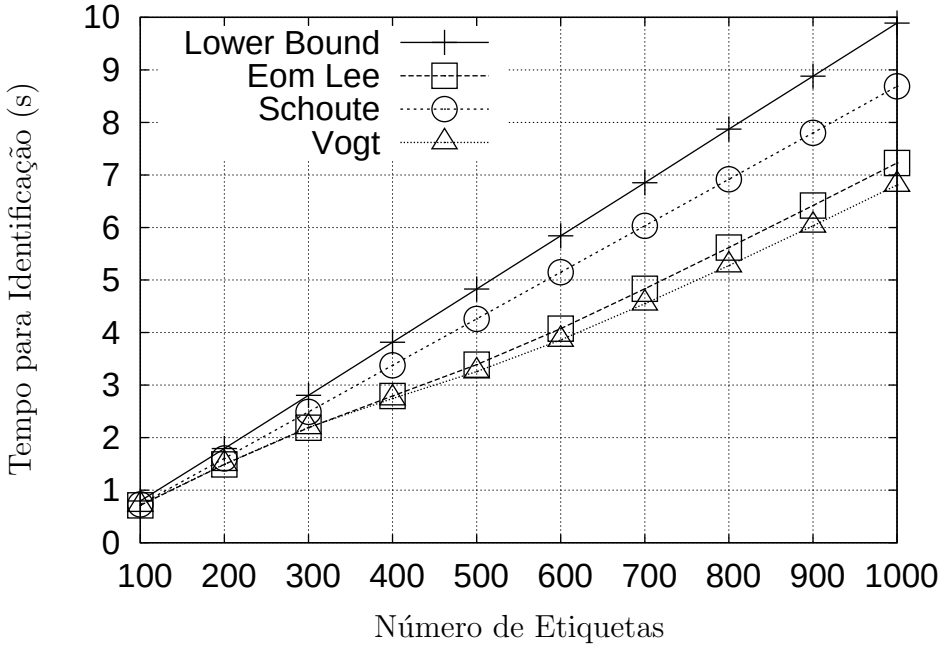


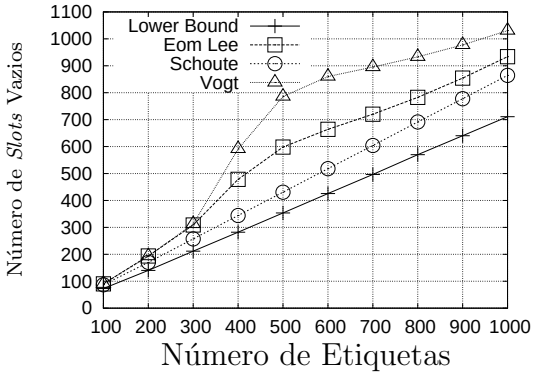
Figura 4.5: Desempenho dos estimadores considerando quantidade de *slots*.

A Figura 4.5 mostra o número total médio de *slots* usados em função da quantidade de etiquetas a serem identificadas. Até pouco menos de 900 etiquetas, o estimador Eom-Lee produziu os melhores resultados, embora seja verificado que os demais estimadores

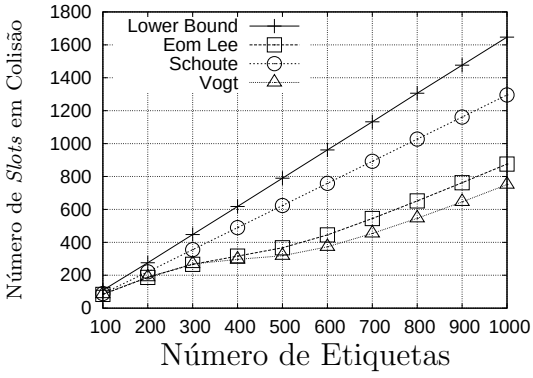
tenham produzido resultados próximos para determinadas quantidades de etiquetas. Note que até uma quantidade de 500 etiquetas, os estimadores possuem pouca diferença quando se trata de quantidade de *slots* e que, a partir desse ponto, as diferenças começam a aumentar. A partir de 800 etiquetas observa-se que o estimador Vogt possui desempenho similar ao estimador Eom-Lee.



(a) Tempo de identificação.



(b) Quantidade de *slots* vazios.



(c) Quantidade de *slots* em colisão.

Figura 4.6: Desempenho dos estimadores considerando outras métricas.

A Figura 4.6(a) mostra a média do tempo total de identificação em função do número

total de etiquetas. Em particular, observa-se sob o ponto de vista dessa métrica, que os estimadores Vogt e Eom-Lee são os melhores e permitem um tempo de identificação comparável até 400 etiquetas. A partir desse valor, o Vogt leva uma ligeira vantagem sobre o Eom-Lee. Isso é explicado pelo impacto da extensão *Early-End*, pois como pode-se perceber, o algoritmo de estimativa Vogt possui uma quantidade maior de *slots* vazios (Figura 4.6(b)) e consequentemente uma quantidade menor de colisões (Figura 4.6(c)) quando comparado ao estimador Eom-Lee.

Vale salientar que algumas conclusões importantes sobre o DFSA e a extensão *Early-End* são obtidas através da análise dos estimadores:

- Uma melhoria na quantidade total de *slots* utilizados pode não implicar em uma redução no tempo total do processo de identificação;
- Minimizar o erro de identificação sem considerar qual tamanho de *frame* escolher não minimiza o tempo de identificação;
- Reduzir adequadamente a quantidade de *slots* em colisão com aumento adequado na quantidade de *slots* vazios em cada *frame* gerado, contribui para a minimização do tempo total de identificação;
- A métrica de avaliação de desempenho mais importante é o tempo total para a identificação das etiquetas, uma vez que é o fator que, efetivamente, afeta o usuário do sistema.

4.9 RESUMO

Graças à característica do DFSA de atingir o máximo desempenho, em termos de vazão, quando se conhece o número de etiquetas a serem identificadas e de definir o tamanho do *frame* dinamicamente, é necessário que existam algoritmos capazes de estimar a quantidade de etiquetas no meio, uma vez que, conhecer quantas etiquetas estão se submetendo

ao processo de identificação é infactível no mundo real. Desse modo, surgiram diversos tipos de estimadores, desde os mais simples matematicamente, até os que aplicam técnicas de inteligência artificial.

Estimadores possuem duas funções básicas: estimar a quantidade de etiquetas e definir o tamanho do próximo *frame*. Dentre os diversos estimadores, foram estudados quatro: *Lower Bound*, Schoute, Vogt e Eom-Lee.

O estimador *Lower Bound* é uma derivação lógica do processo de colisão que, por sua vez, dita que para existir uma colisão são necessárias ao menos duas etiquetas transmitindo ao mesmo tempo. O estimador de Schoute se utiliza de probabilidade para calcular qual a provável quantidade de etiquetas que estarão transmitindo num *slot* em colisão. Vogt também se utiliza de probabilidade, considerando a equação de Chebyshev e utilizando uma busca numa função para estimar a quantidade de etiquetas. Finalmente, Eom-Lee também utiliza probabilidade, mas cria um fórmula iterativa para estimar a quantidade de etiquetas.

Dentre os métodos utilizados para selecionar o tamanho do próximo *frame*, o *Lower Bound* e o Schoute se valem do mesmo método, isto é, número de etiquetas estimada menos a quantidade de etiquetas identificadas com sucesso. Vogt, que fez um estudo com um sistema real, utiliza uma tabela onde, dando uma quantidade de etiquetas dentro de um certo limite, um tamanho de *frame* é escolhido. Essa tabela não foi utilizado porque possui um tamanho máximo para o próximo *frame* de 256 *slots* e todos os tamanhos de *frames* possíveis tem que ser múltiplo de potência de 2. Desse modo, foi empregado para o estimador de Vogt o método utilizado pelos algoritmos *Lower Bound* e Schoute. Eom-Lee, conforme mencionado, utiliza seu método iterativo.

Graças à análise realizada, foram obtidas conclusões importantes sobre o protocolo DFSA com a utilização da extensão *Early-End*, quais sejam:

- Uma melhoria na quantidade total de *slots* utilizados pode não implicar em uma

redução no tempo total do processo de identificação;

- Minimizar o erro de identificação sem considerar qual tamanho de *frame* escolher não minimiza o tempo de identificação;

- Reduzir adequadamente a quantidade de *slots* em colisão com aumento adequado na quantidade de *slots* vazios em cada *frame* gerado, contribui para a minimização do tempo total de identificação;

- A métrica de avaliação de desempenho mais importante é o tempo total para a identificação das etiquetas, uma vez que é o fator que, efetivamente, afeta o usuário do sistema.

CAPÍTULO 5

A FUNÇÃO DE CÁLCULO PROPOSTA E RESULTADOS

Este capítulo apresenta a função proposta para o cálculo do tamanho de *frames* quando é considerado o impacto da extensão *Early-End*, os resultados obtidos com a aplicação da função e seu impacto no tempo de identificação.

5.1 FUNÇÃO PROPOSTA

Quando se considera o número total de *slots* utilizados em um processo de identificação como equivalente ao tempo de identificação, é assumido, indiretamente, que os diferentes tipos de *slots* possuem o mesmo custo em relação ao tempo, ou seja, um *slot* em colisão possui a mesma duração de um *slot* vazio e de um *slot* bem sucedido. Entretanto, isso não ocorre quando a extensão *Early-End* é utilizada, como visto na Seção 4.8.

Considerando a diferença de duração dos *slots*, temos que o melhor desempenho do DFSA não é atingido quando reduzimos a quantidade de *slots* utilizados num processo de identificação, mas quando minimizamos o tempo total de identificação, conforme visto no Capítulo 4. Assumindo a utilização do *Early-End*, temos que a minimização do tempo ocorre quando priorizamos a utilização de *slots* em vazio em vez de *slots* em colisão. É importante salientar que o aumento descontrolado da quantidade de *slots* vazios irá custar mais que um *slot* em colisão, eventualmente.

Desse modo, é proposta uma função de cálculo de tamanho de *frames* que explora

as diferentes durações dos *slots* geradas pela utilização da extensão *Early-End* visando a redução do tempo de identificação. Como o tamanho do próximo *frame* no DFSA depende do número total de etiquetas estimadas, é proposto que o tamanho $\hat{f}$ do próximo *frame* calculado pelo estimador seja ajustado por um fator multiplicativo δ_i. Onde i representa o tamanho do *frame* inicial. Assim sendo, é proposta a seguinte função F que relaciona a função de cálculo do estimador com esse fator de ajuste:

$$F(\delta_i, \hat{f}) = \delta_i \hat{f} \quad . \tag{5.1}$$

O valor de δ_i a ser utilizado na Equação (5.1) precisa ser determinado individualmente para cada estimador, buscando-se a minimização do tempo total do processo de identificação de etiquetas. Para que exista a minimização do tempo de identificação, é necessário definir o modelo do canal de comunicação. O modelo do canal utilizado foi o mesmo mostrado no Capítulo 4, Seção 4.7. A determinação do valor de δ será apresentada na Seção 5.2.

5.2 DETERMINAÇÃO DO FATOR δ

Os valores de δ_i a serem utilizados na função de cálculo $F(\delta_i, \hat{f})$ proposta foram determinados através de simulações, avaliando o tempo total de identificação.

Os parâmetros de simulação foram: taxa de transmissão de 40 kbps e o identificador das etiquetas com tamanho de 128 *bits*. Os fatores utilizados foram o *frame* inicial igual a 64 e 128 *slots*, o valor de δ_i foi variado de 1 a 6 em passos de 0,2, onde o valor máximo de 6 foi definido por que o aumento demasiado na quantidade de *slots* vazios tornará, eventualmente, o processo mais lento, e variando a quantidade de etiquetas em 100, 300, 500, 700 e 1.000 etiquetas. Os estimadores Eom-Lee, Vogt e Schoute foram avaliados. Vale salientar que no gráficos apresentados nesta seção, os pontos são uma média obtida a partir de 2.000 execuções para cada conjunto de fatores de simulação, isto é, o ponto

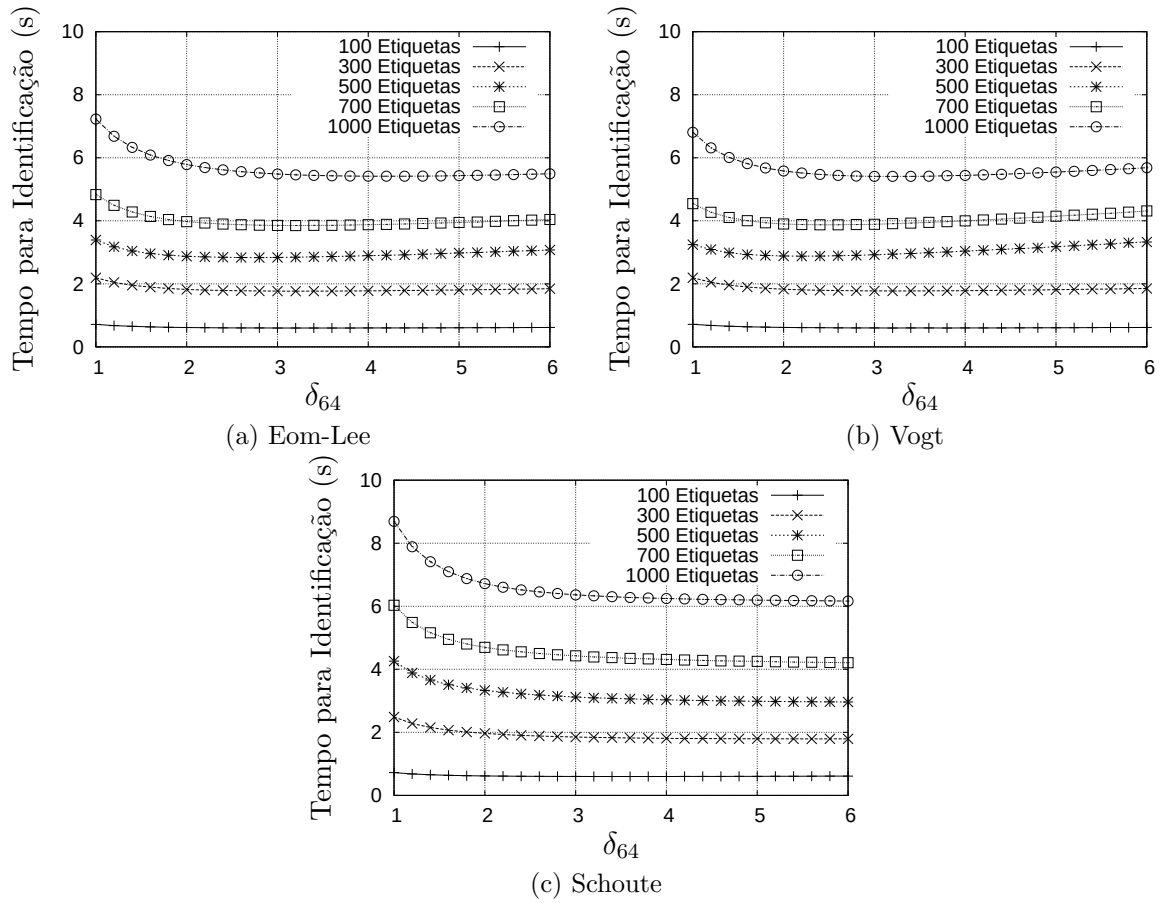
para o valor de $\delta_i = 1$, $L_0 = 64$ e 100 etiquetas foi conseguido através da média de 2.000 execuções com esses fatores, por exemplo.

As Figuras 5.1 e 5.2 mostram, respectivamente, a influência do fator de ajuste no tempo total de identificação de etiquetas para um *frame* inicial de 64 *slots* e de 128 *slots*. Em particular, observa-se que a minimização do tempo total de identificação depende do valor do fator de ajuste, o qual depende também da quantidade total de etiquetas a serem identificadas. O problema, em questão, é encontrar um valor adequado para o fator de ajuste que permita a obtenção de ganhos no tempo total de identificação para qualquer quantidade de etiquetas no intervalo $[100, 1000]$.

É importante salientar que o valor de δ_i será ótimo apenas para uma pequena quantidade de etiquetas dentro do intervalo $[100, 1000]$, mas que o valor utilizado deve sempre reduzir o tempo de identificação para qualquer quantidade de etiquetas dentro desse intervalo.

No caso do Eom-Lee, o valor de δ_{64} (Figura 5.1(a)) que minimiza o tempo total de identificação para 300 e 500 etiquetas é igual a 3,0. Esse valor também permite uma redução no tempo total de identificação para o caso de haver 100, 700 ou 1000 etiquetas a serem identificadas, apesar do tempo total não ser minimizado. Enquanto $\delta_{64} = 3{,}0$ não é um valor ótimo para 100, 700 e 1000 etiquetas, o módulo da diferença do tempo total de identificação com tal valor e o valor do δ_{64} ótimo para tais quantidades de etiquetas é, respectivamente, igual a 0,5 ms, 3,39 ms e 69,54 ms. Como é possível observar, essas diferenças não são significativas.

Com relação ao Vogt, o valor de δ_{64} (Figura 5.1(b)) que minimiza o tempo total de identificação para 100 e 300 etiquetas é igual a 3,2. Com esse valor, o tempo total de identificação também é reduzido quando há 500, 700 ou 1000 etiquetas a serem identificadas, embora o tempo total de identificação não seja minimizado. O módulo da diferença do tempo total de identificação entre o caso de $\delta_{64} = 3{,}2$ e os fatores ótimos para as quantidades de etiquetas de 100, 300 e 1000 é, respectivamente, igual a 63,7 ms, 354,66 ms e



(a) Eom-Lee

(b) Vogt

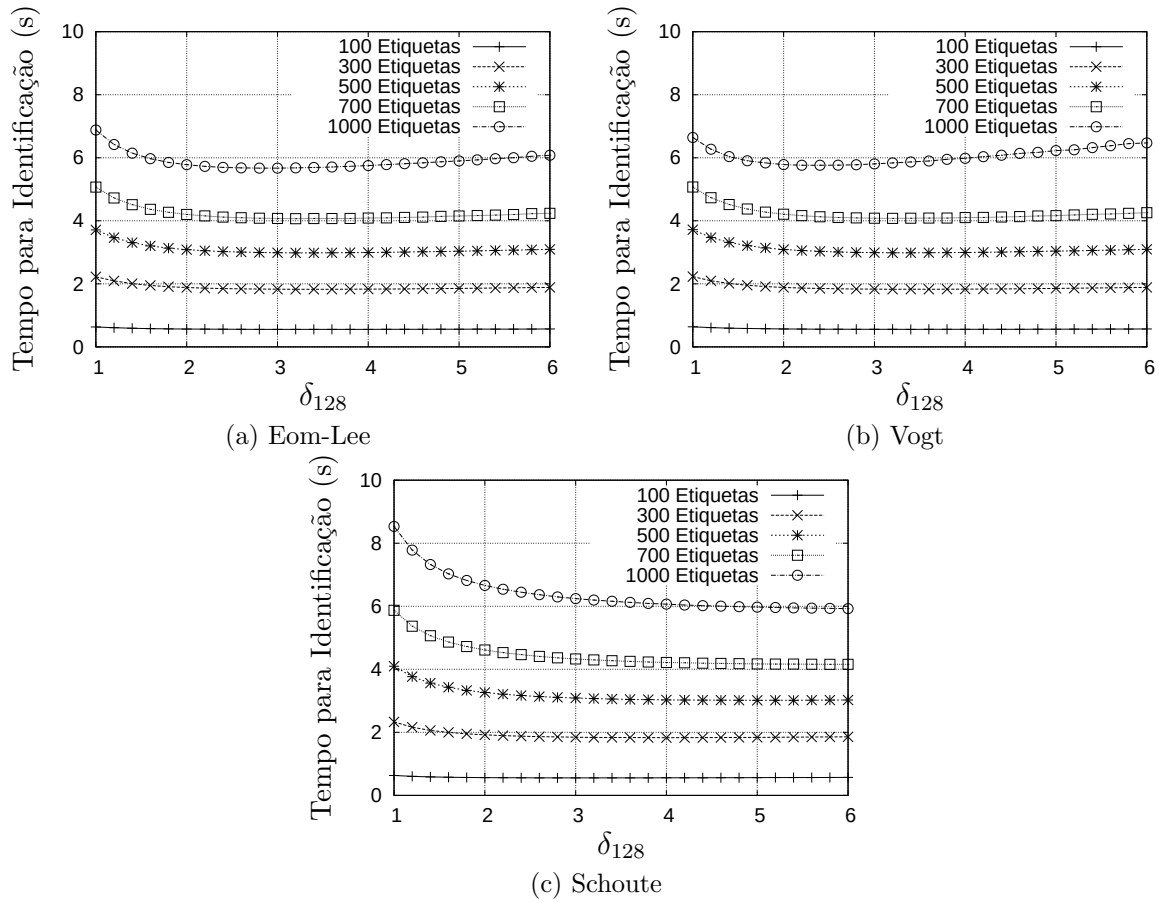
(c) Schoute

Figura 5.1: Influência do fator δ para tamanho de *frame* inicial de 64 *slots*.

0,3 ms. Mais uma vez, as diferenças não são significativas.

No caso do Schoute, $\delta_{64} = 6{,}0$ (Figura 5.1(c)) produz o menor tempo total de identificação para 300, 500, 700 e 1000 etiquetas. Para uma quantidade de etiquetas igual a 100, o valor de δ_{64} que minimiza o tempo total de identificação é igual a 3,6. Contudo, para essa quantidade de etiquetas, um $\delta_{64} = 6{,}0$ ainda produz ganhos de desempenho em relação ao caso de se utilizar um valor de $\delta_{64} = 1{,}0$. O módulo da diferença do tempo total de identificação entre o valor ótimo para 100 etiquetas e o valor proposto para δ_{64} é igual a 14,05 ms.

No caso do Eom-Lee para um *frame* inicial de 128 *slots*, o valor do fator de ajuste (Figura 5.2(a)) que minimiza o tempo total de identificação para 500 e 700 etiquetas é



(a) Eom-Lee

(b) Vogt

(c) Schoute

Figura 5.2: Influência do fator δ para tamanho de *frame* inicial de 128 *slots*.

igual a 3,2. Esse valor também permite uma redução no tempo total de identificação para 100, 300 e 1000 etiquetas, embora o tempo total de identificação não seja minimizado. Apesar de $\delta_{128} = 3{,}2$ não ser um valor ótimo para 100, 300 e 1000 etiquetas, o módulo da diferença do tempo total de identificação com tal valor e o valor do δ_{128} ótimo para tais quantidades de etiquetas é, respectivamente, igual a 0,47 ms, 1,1 ms e 6,72 ms. Logo, a diferença não é significativa. De acordo com as análises feitas, $\delta_{64} = 3{,}0$ e $\delta_{128} = 3{,}2$ são valores adequados para o fator de ajuste.

Para o estimador de Vogt com um *frame* inicial de 128 *slots*, o valor do fator de ajuste (Figura 5.2(b)) que minimiza o tempo total de identificação para 300 e 700 etiquetas também é igual a 3,2. Esse valor também permite uma redução no tempo total de

identificação para o caso de haver 100, 500 ou 1000 etiquetas a serem identificadas, mesmo não minimizando o tempo total. O módulo da diferença do tempo total de identificação obtido ao se utilizar $\delta_{128} = 3{,}2$ e os fatores ótimos para quantidades de etiquetas de 100, 500 e 1000 é, respectivamente, igual a 0,472 ms, 1,062 ms e 75,018 ms. Com base nas análises apresentadas, $\delta_{64} = 3{,}2$ e $\delta_{128} = 3{,}2$ são valores adequados para o fator de ajuste.

Finalmente, para o estimador de Schoute e um *frame* inicial de 128 *slots*, $\delta_{128} = 5{,}8$ (Figura 5.2(c)) minimiza o tempo total de identificação para 700 etiquetas. O módulo da diferença do tempo total de identificação entre o caso de $\delta_{128} = 5{,}8$ e os fatores ótimos para 100, 300, 500 e 1000 etiquetas é, respectivamente, igual a 14,591 ms, 23,405 ms, 5,996 ms e 8,313 ms. Mais uma vez, a diferença de tempo observada não é significativa. Com base no exposto, $\delta_{64} = 6{,}0$ e $\delta_{128} = 5{,}8$ são valores adequados para o fator de ajuste.

5.3 RESULTADOS

Uma vez que temos os valores de δ, se torna necessário avaliar o desempenho quando a função proposta é utilizada. Essa avaliação será mostrada a seguir. Cada ponto mostrado nos gráficos desta seção é uma média de 2.000 execuções.

5.3.1 Métricas

As seguintes métricas de desempenho foram avaliadas: quantidade total de *slots*, quantidade total de *slots* vazios, quantidade total de *slots* em colisão, tempo total para a identificação das etiquetas, eficiência de *slots* e eficiência temporal. Define-se *eficiência de slots* como sendo o número total de *slots* bem-sucedidos dividido pelo número total de *slots* utilizados no processo de identificação. Define-se *eficiência temporal* como sendo o número total de etiquetas multiplicado por T_5 (Seção 4.7) e dividido pela duração total do processo de identificação.

5.3.2 Cenário e Condições para Simulação

O cenário de simulação possui um leitor e diversas etiquetas, onde uma vez iniciado o processo de identificação este só é encerrado quando todas as etiquetas são identificadas. Nenhuma nova etiqueta é inserida enquanto o processo de identificação está em execução. A fim de se estudar apenas o impacto dos estimadores no desempenho do processo de identificação, considera-se um canal de comunicação livre de erros e sem efeito de captura.

Como parâmetros de simulação, foi utilizado o protocolo DFSA com as extensões *Early-End* e *Muting* habilitadas, tamanho do *frame* inicial de 64 *slots*, taxa de transmissão de 40 kbps, tanto para *downstream* como para *upstream* e as etiquetas possuem um identificador de 128 *bits*. Particularmente para o estimador Eom-Lee, adota-se o valor de $\epsilon_{threshold}$ igual a 0,001, conforme os autores originais sugeriram. A modelagem do canal utilizada foi a mesma mostrada no Capítulo 4, Seção 4.7.

Os fatores de simulação considerados foram: quantidade de etiquetas varia de 100 a 1000, com passo igual a 100; *frame* inicial de 64 e 128 *slots* e os valores de δ_i são utlizados conforme estimador e *frame* inicial estudados.

5.3.3 Análise dos Resultados

As Figuras de 5.3, a 5.8 apresentam a comparação entre os resultados obtidos com e sem a função proposta de acordo com os estimadores estudados. Para todos os casos avaliados, a função proposta permitiu uma melhoria na eficiência temporal devido à redução no tempo total de identificação de etiquetas. É importante notar que, apesar do aumento da quantidade total de *slots* utilizados no processo de identificação, a função proposta permite explorar as vantagens da extensão *Early-End*, reduzindo o tempo total do processo de identificação, que, conforme visto anteriormente, é a métrica mais importante.

Essa redução ocorre, pois a função proposta superestima a quantidade de etiquetas

com o intuito de permitir uma quantidade "razoável" de *slots* vazios a cada novo *frame* gerado. Em outras palavras, existe uma priorização dos *slots* vazios, reduzindo os *slots* em colisão.

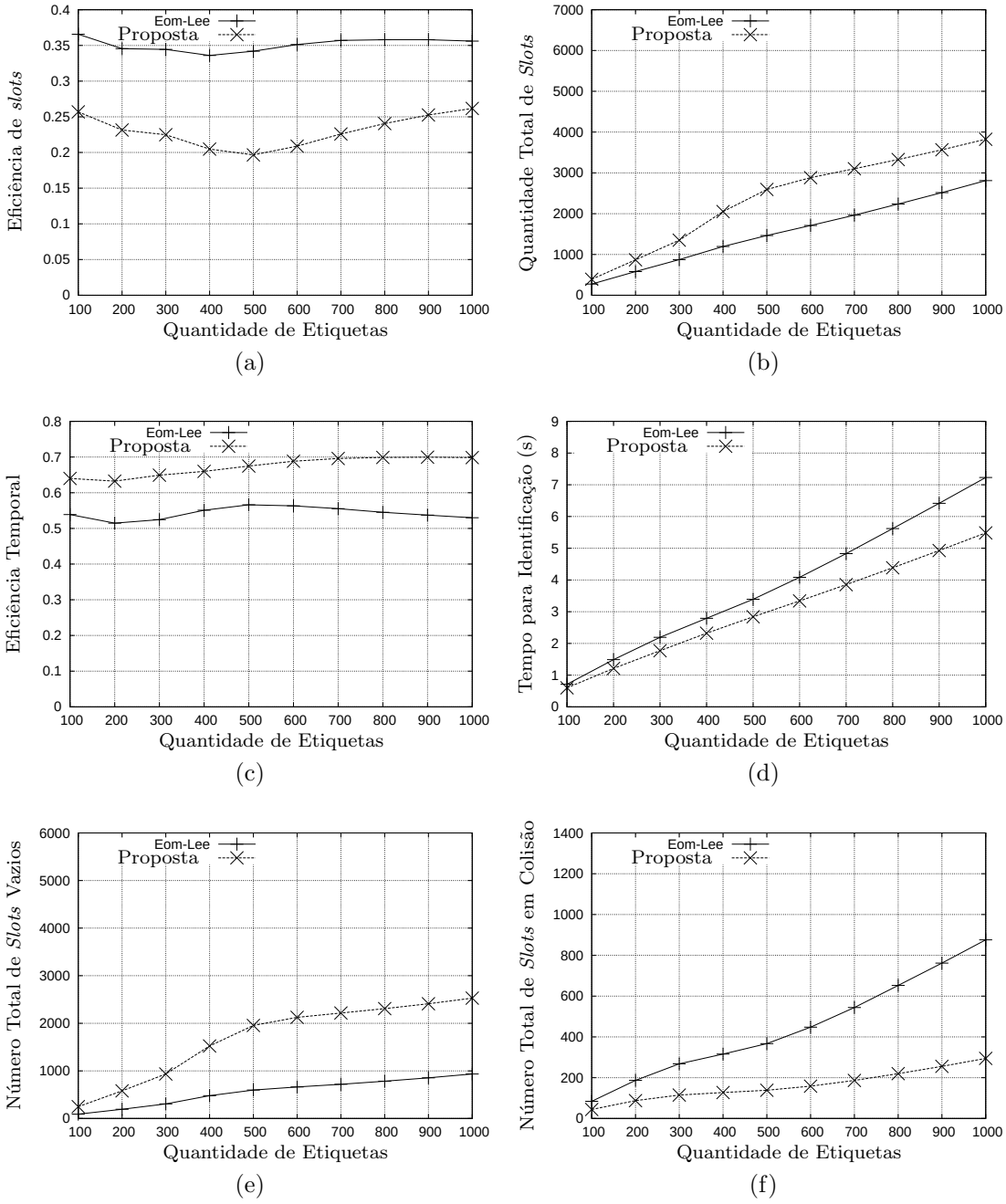


Figura 5.3: Resultados com o uso do estimador Eom-Lee, *frame* inicial igual a 64 *slots*.

A Figura 5.3 mostra os resultados para o estimador Eom-Lee com *frame* inicial igual a

64. Conforme previsto, Figura 5.3(e), o método proposto contém uma quantidade muito maior de *slots* vazios do que o método original, reduzindo a quantidade total de *slots* em colisão, como mostra a Figura 5.3(f). Também pode-se observar na Figura 5.3(b) que quando se utiliza o método proposto são necessários aproximadamente 32% *slots* a mais do que no método original para 1.000 etiquetas, o que impacta na eficiência, Figura 5.3(a), onde a eficiência de *slots* do Eom-Lee está, aproximadamente entre 34% e 36% para todos os casos enquanto quando se utiliza o fator δ_{64} temos uma eficiência de 20% à 26%. Entretanto, ganha-se no tempo de identificação, como mostrado na Figura 5.3(d), onde o método proposto possui um ganho aproximado de 24%, para 1000 etiquetas, se comparado com o método original; isto é, se reflete na eficiência temporal, onde o método proposto possui uma eficiência temporal entre 63% e 70%, enquanto originalmente essa eficiência está entre 51% e 57%, como pode ser visto na Figura 5.3(c).

A Figura 5.4 mostra os resultados para o estimador Eom-Lee com *frame* inicial igual a 128. Pode ser visto nas Figuras 5.4(e) e 5.4(f) que a proposta apresentada nesta dissertação aumenta a quantidade de *slots* vazios e reduz a quantidade de *slots* em colisão conforme esperado. Também é possível observar que o método proposto possui uma ganho de até, aproximadamente, 17% no tempo de identificação (Figura 5.4(d)), apesar do aumento de até 78% na quantidade de total de *slots* necessários para realizar o processo de identificação (Figura 5.4(b)). Em termos de eficiência, é possível perceber através das Figuras 5.4(a) e 5.4(c) que o método proposto possui uma eficiência temporal entre 63% e 70% comparado com o método original, que possui uma eficiência temporal entre 50% e 60% e, conforme esperado, a eficiência de *slots* do método original é um pouco maior do que a obtida utilizando o método proposto.

A Figura 5.5 mostra os resultados para o estimador de Vogt para um *frame* inicial de 64. É possível observar que a eficiência temporal do método proposto é de aproximadamente 70% para 1.000 etiquetas e de 59% para o método original (Figura 5.5(c)). Esse ganho é explicado pelo ganho no tempo de identificação, que é de até 22% se comparado

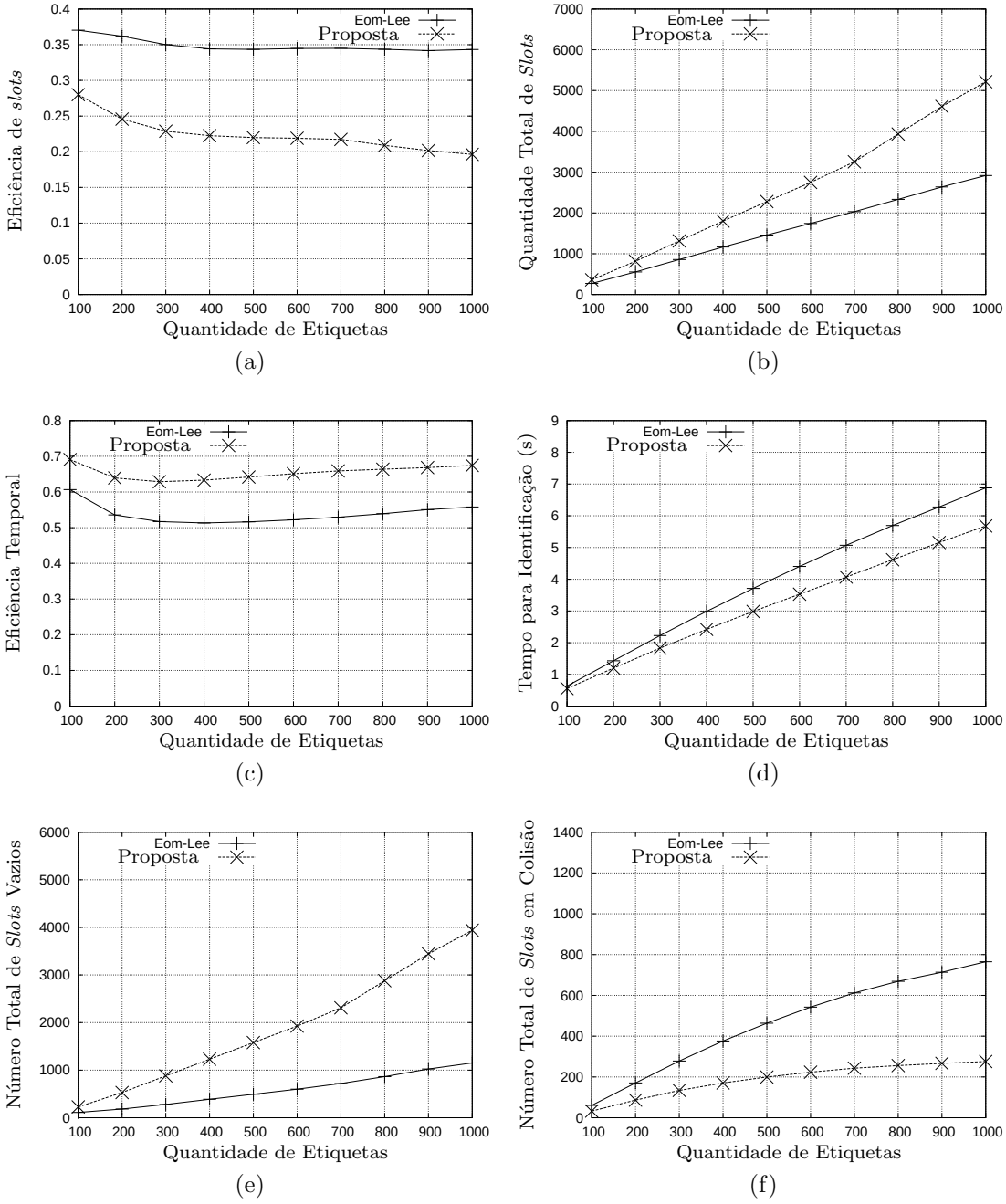


Figura 5.4: Resultados com o uso do estimador Eom-Lee, *frame* inicial igual a 128 *slots*.

com a utilização do estimador sem a proposta (Figura 5.5(d)). Em termos de quantidade total de *slots* e eficiência de *slots*, Figuras 5.5(b) e 5.5(a), o método original se sai melhor, conforme esperado. Entretanto, é importante lembrar que a métrica mais importante é o tempo de identificação. Esse comportamento é suportado pela troca de *slots* em colisão

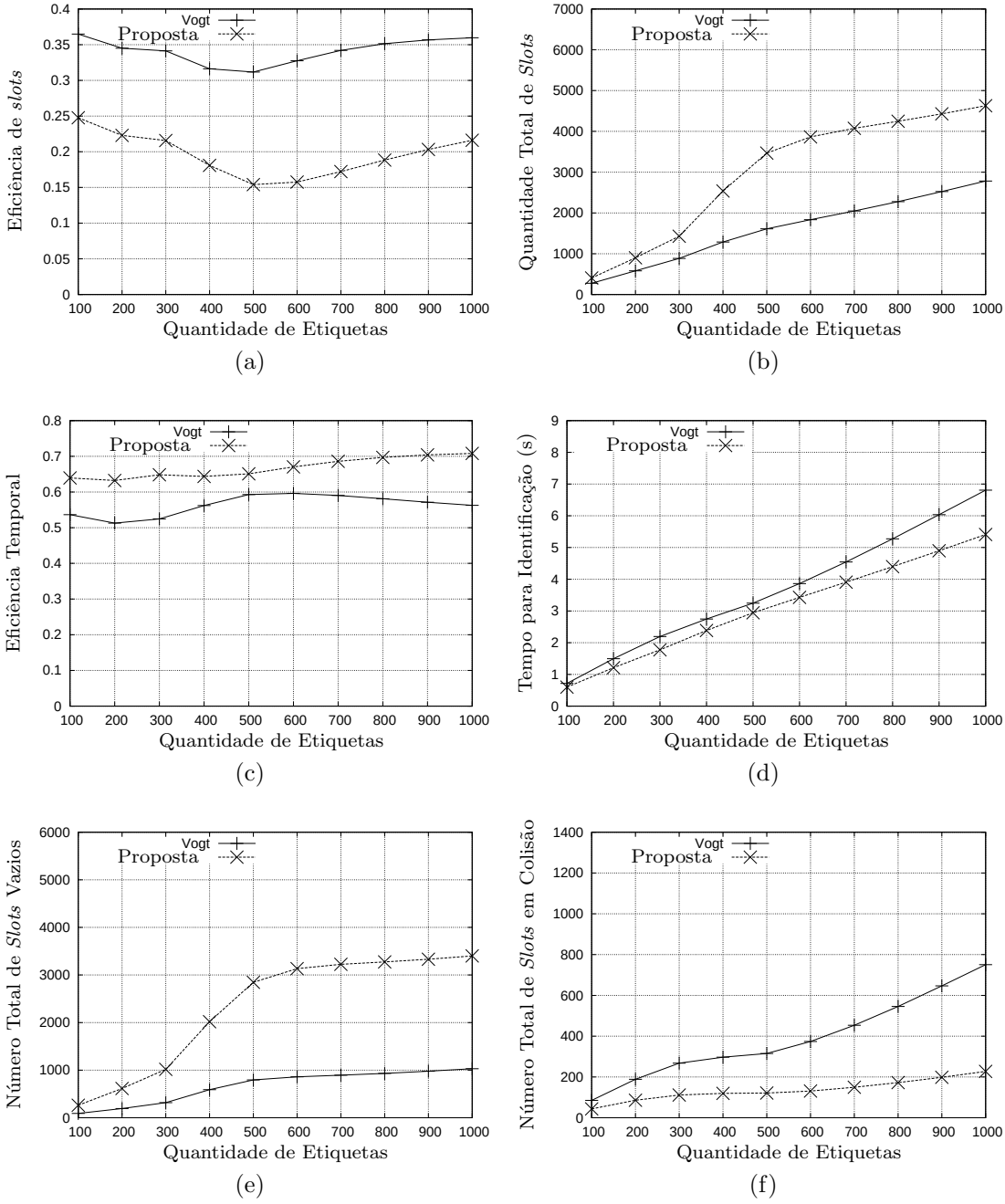


Figura 5.5: Resultados com o uso do estimador Vogt, *frame* inicial igual a 64 *slots*.

por *slots* vazios (estes com menor duração), como esperado quando o método proposto é utilizado, como pode ser visto nas Figuras 5.5(f) e 5.5(e).

A Figura 5.6 mostra os resultados para o estimador Vogt com *frame* inicial igual a 128. Conforme previsto, Figura 5.6(e), o método proposto contém uma quantidade

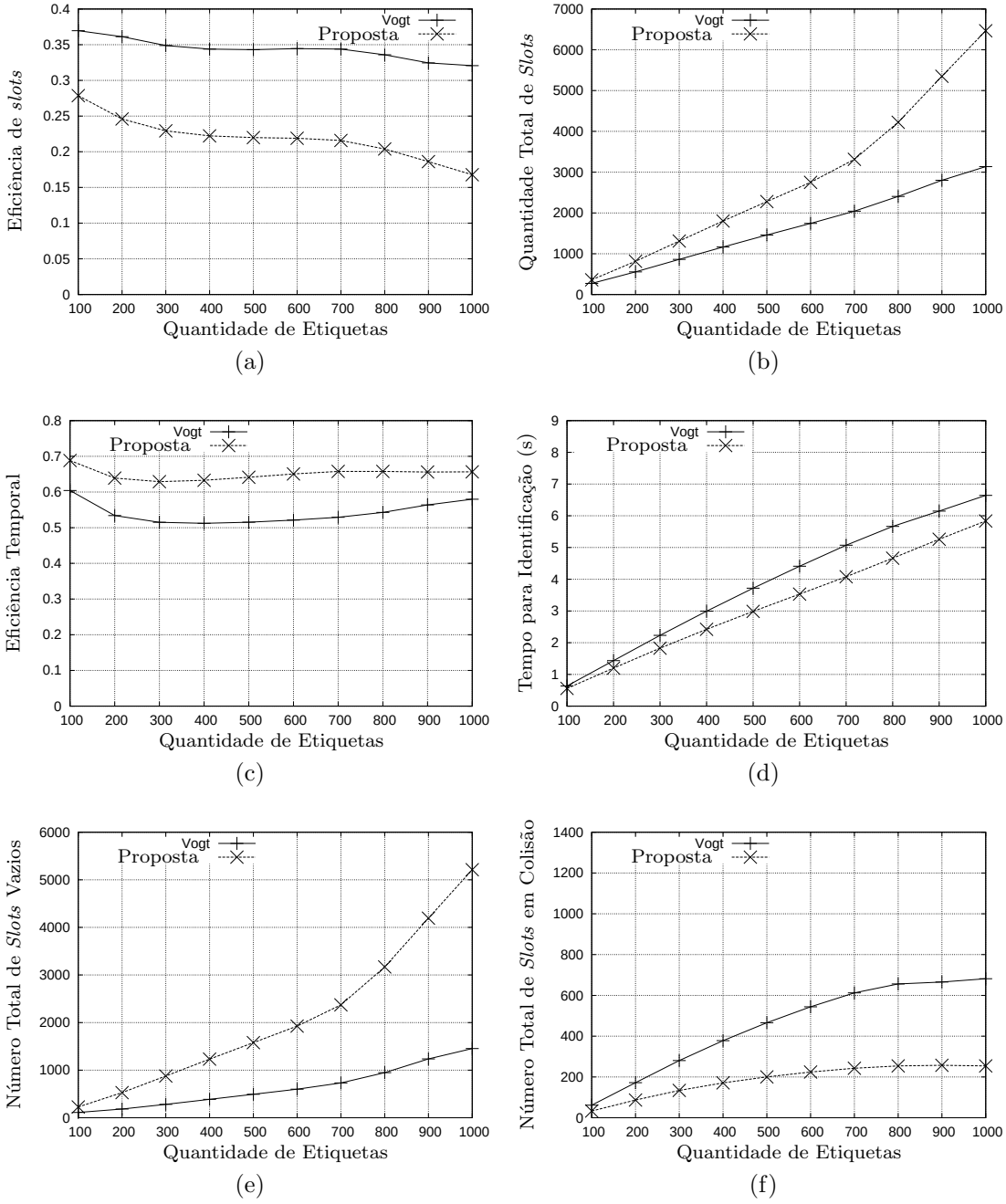


Figura 5.6: Resultados com o uso do estimador Vogt, *frame* inicial igual a 128 *slots*.

muito maior de *slots* vazios do que o método original, reduzindo a quantidade total de *slots* em colisão, como mostra a Figura 5.6(f). Também pode-se observar na Figura 5.6(b) que quando se utiliza o método proposto são necessários aproximadamente 113% *slots* a mais do que no método original para 1.000 etiquetas, o que impacta na eficiência, Figura

5.6(a), onde a eficiência de *slots* do Vogt está, aproximadamente entre 33% e 37% para todos os casos enquanto quando se utiliza o fator δ_{128} temos uma eficiência de 17% à 27%. Entretanto, ganha-se no tempo de identificação, como mostrado na Figura 5.6(d), onde o método proposto possui um ganho aproximado de 12%, para 1.000 etiquetas, se comparado com o método original; isto é, se reflete na eficiência temporal, onde o método proposto possui uma eficiência temporal de 65% para 1000 etiquetas, enquanto originalmente essa eficiência é de 58%, como pode ser visto na Figura 5.6(c).

A Figura 5.7 mostra os resultados para o estimador Schoute com *frame* inicial igual a 64. Pode ser visto nas Figuras 5.7(e) e 5.7(f) que a proposta apresentada nesta dissertação aumenta a quantidade de *slots* vazios e reduz a quantidade de *slots* em colisão conforme esperado. Também é possível observar que o método proposto possui uma ganho de até, aproximadamente, 30% no tempo de identificação (Figura 5.7(d)), apesar do aumento de até 90% na quantidade de total de *slots* necessários para realizar o processo de identificação (Figura 5.7(b)). Em termos de eficiência, é possível perceber através das Figuras 5.7(a) e 5.7(c) que o método proposto possui uma eficiência temporal entre 60% e 65% comparado com o método original, que possui uma eficiência temporal entre 44% e 53% e, conforme esperado, a eficiência de *slots* do método original é um pouco maior do que a obtida utilizando o método proposto.

A Figura 5.8 mostra os resultados para o estimador de Schoute para um *frame* inicial de 128. É possível observar que a eficiência temporal do método proposto é de aproximadamente 68% para 1.000 etiquetas e de 60% para o método original (Figura 5.8(c)). Esse ganho é explicado pelo ganho no tempo de identificação, que é de até 30% se comparado com a utilização do estimador sem a proposta (Figura 5.8(d)). Em termos de quantidade total de *slots* e eficiência de *slots*, Figuras 5.8(b) e 5.8(a), o método original se sai melhor, conforme esperado. Entretanto, é importante lembrar que a métrica mais importante é o tempo de identificação. Esse comportamento é suportado pela troca de *slots* em colisão por *slots* vazios (estes com menor duração), como esperado quando o método proposto é

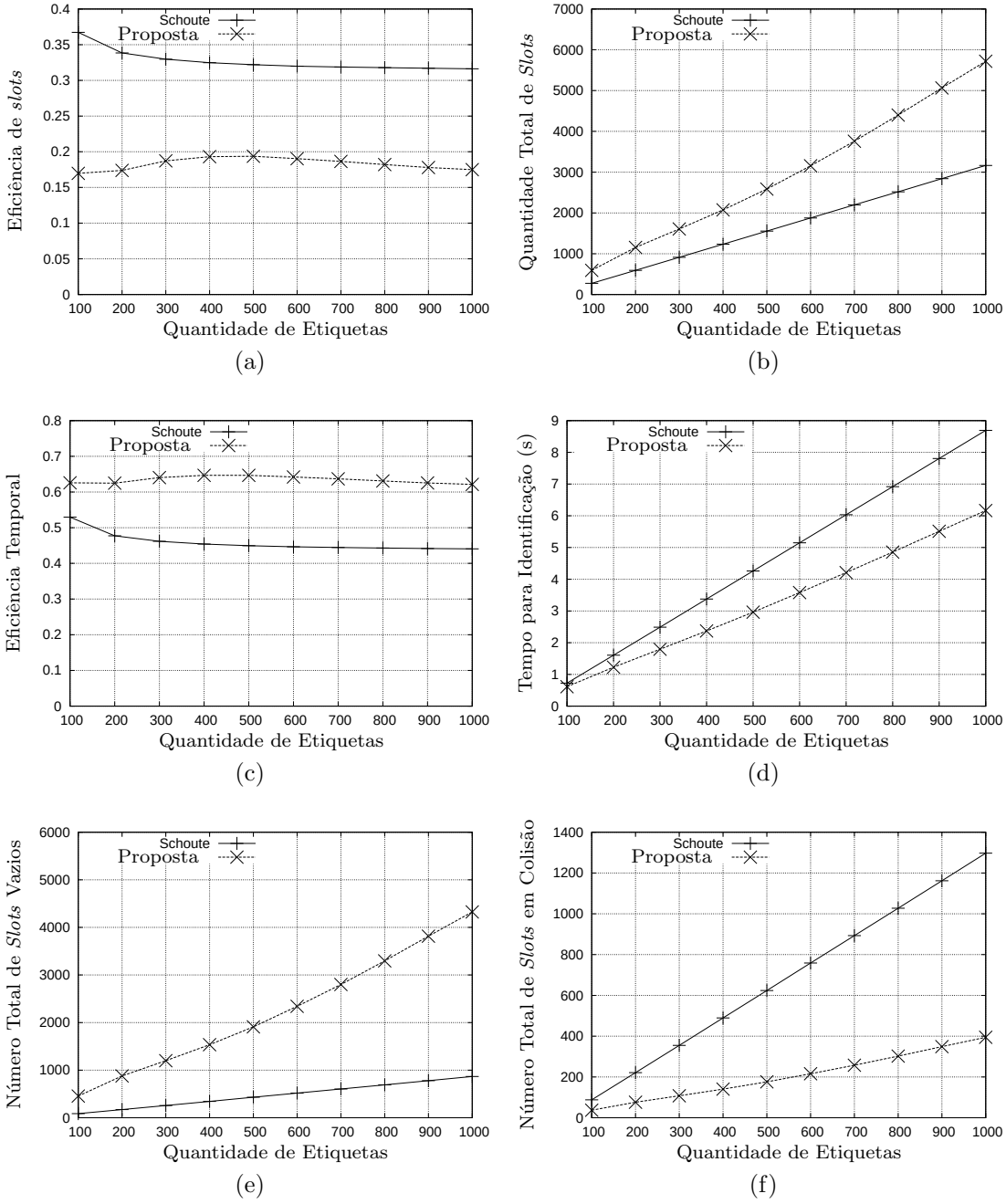


Figura 5.7: Resultados com o uso do estimador Schoute, *frame* inicial igual a 64 *slots*.

utilizado, como pode ser visto nas Figuras 5.8(f) e 5.8(e).

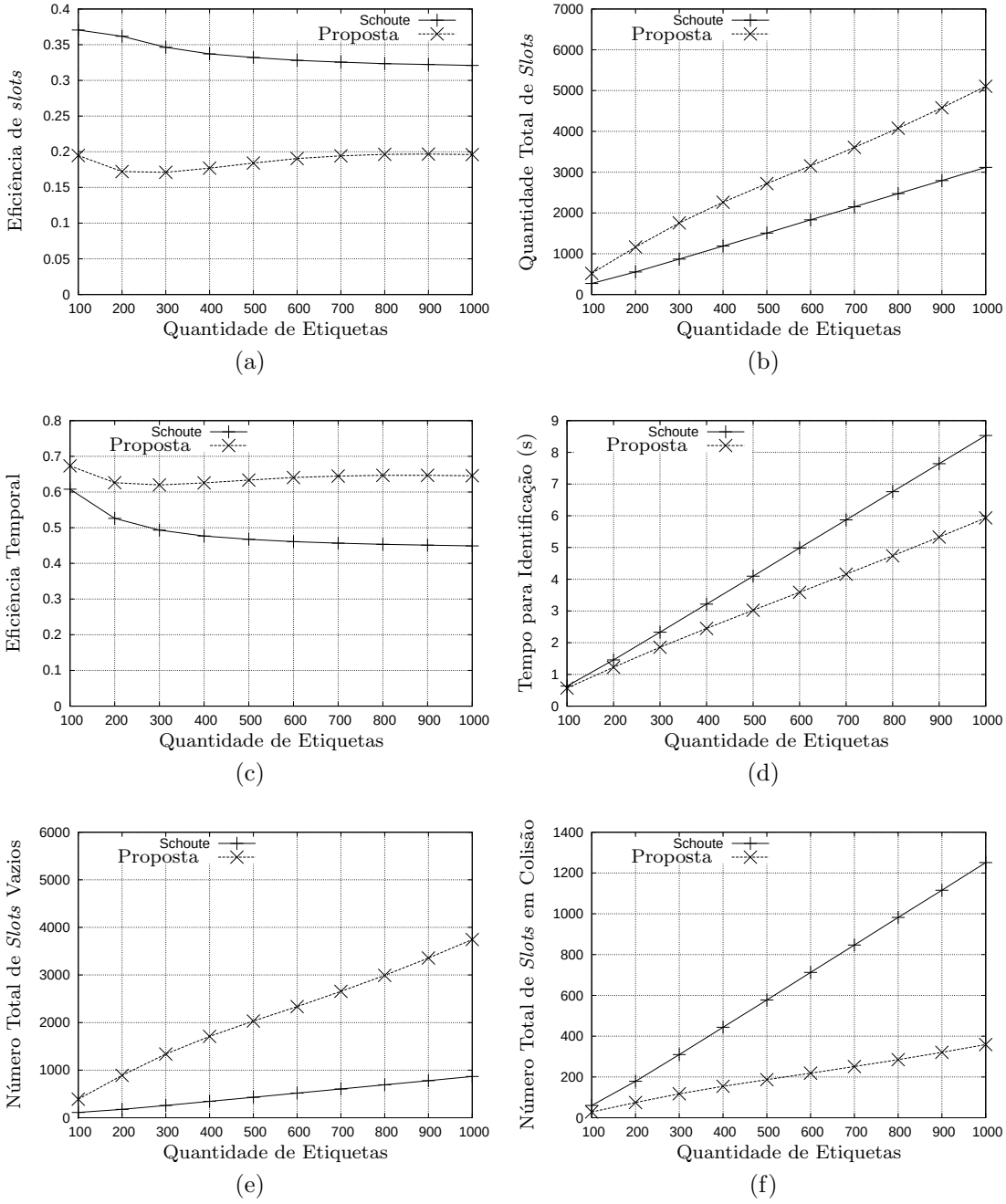
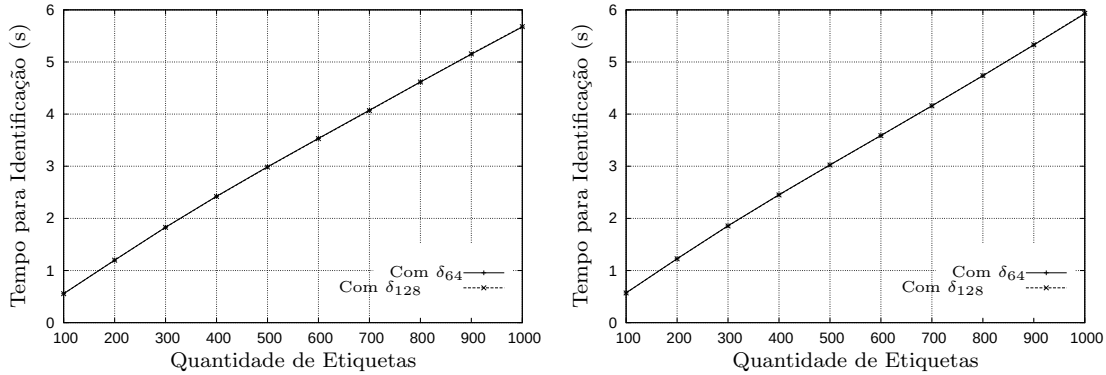


Figura 5.8: Resultados com o uso do estimador Schoute, *frame* inicial igual a 128 *slots*.

5.4 COMPARAÇÃO ENTRE OS RESULTADOS

Os fatores de ajuste encontrados na Seção 5.2 para cada estimador não diferiram significativamente ao se mudar o tamanho do *frame* inicial de 64 para 128 *slots*. Além disso,



(a) Eom-Lee (b) Schoute

Figura 5.9: Impacto dos fatores de ajuste para um *frame* inicial de 128 *slots*.

os valores de δ_{64} e δ_{128} encontrados são sub-ótimos para várias quantidades de etiquetas. Contudo, o tempo total de identificação ao se utilizar um fator ótimo não se mostrou significativamente diferente do tempo obtido com os fatores sub-ótimos indicados. Tudo isso sugere que para ambos os tamanhos de *frame* inicial estudados, um mesmo fator de ajuste pode ser utilizado sem prejuízo perceptível no tempo total de identificação. A avaliação dessa hipótese é apresentada na Figura 5.9. Ela mostra o tempo total de identificação para o Eom-Lee e para o Schoute com a função proposta considerando um *frame* inicial igual a 128 *slots* e os fatores de ajuste δ_{64} e δ_{128}. Note que não há diferenças perceptíveis no tempo total de identificação.

Não foi realizada esta comparação com o estimador de Vogt por que para este estimador, o valor de $\delta_{64} = \delta_{128}$.

5.5 RESUMO

Quando se considera a extensão *Early-End*, a duração de um *slot* vazio é menor que um *slot* em colisão. Assim sendo, quando existe uma priorização de *slots* vazios sobre *slots* em colisão, é obtido um melhor tempo de identificação.

Desse modo, é proposta uma função de cálculo de tamanho de *frames* que garante essa

Tabela 5.1: Valores de δ sugeridos.

Estimador	δ_{64}	δ_{128}
Eom-Lee	3,0	3,2
Vogt	3,2	3,2
Schoute	6,0	5,8

priorização, visando reduzir o tempo total de identificação. A função proposta possui um parâmetro δ_i que precisa ser determinado, onde i é o tamanho do *frame* inicial. Note que o função proposta utiliza o resultado fornecido pelos estimadores e pode ser estendida para qualquer estimador, desde que seja utilizada a metodologia proposta. Para determinar o fator δ_i, foi considerado um *frame* inicial de 64 e 128 *slots* e foram utilizadas simulações de modo a determinar qual seu melhor valor. Os valores indicados neste trabalho podem ser vistos na Tabela 5.1.

Tendo como base os valores mostrados na Tabela 5.1, foram executadas simulações de modo a comparar o tempo de identificação quando se aplica ou não a função proposta no DFSA. Os resultados obtidos mostram uma melhora no tempo de identificação para todos os estimadores e todas as quantidade de etiquetas estudadas. As melhoras obtidas para o estimador Eom-Lee foram de até, aproximadamente, 24% e 17% para um *frame* inicial de 64 *slots* e de 128 *slots*, respectivamente. Para o algoritmo de Vogt, se atingiu melhoras de até, aproximadamente, 22% e 12% para um *frame* inicial de 64 *slots* e de 128 *slots*, respectivamente. Finalmente, para o estimador de Schoute, as melhoras obtidas foram de até, aproximadamente, 30% para as duas quantidades de *frame* inicial estudadas.

Também foi possível observar que quando se utilizam os fatores de ajuste obtidos para o tamanho de *frame* inicial de 64 *slots* numa simulação com o tamanho de *frame* inicial de 128 *slots* não existem diferenças perceptíveis no tempo de identificação.

CAPÍTULO 6

CONCLUSÃO E TRABALHOS FUTUROS

Foi apresentado, neste trabalho, os conceitos básicos de RFID e alguns de seus protocolos anticolisão. Dentre os protocolos anticolisão, foi apresentado o protocolo DFSA e suas extensões como objeto de estudo.

Um dos fatores determinantes no desempenho do DFSA é como determinar o tamanho do próximo *frame*. Para tal, o DFSA se utiliza de estimadores que são responsáveis por calcular qual a quantidade de etiquetas que estão se submetendo ao processo de identificação.

Contudo, esses estimadores, de modo geral, não visam melhorar o tempo de identificação e não consideram o impacto da extensão *Early-End*, que tem relação direta com o tempo de identificação. Assim sendo, foi necessário um estudo dos estimadores considerando o tempo de identificação. Desse estudo foi possível concluir:

- Uma redução na quantidade total de *slots* utilizados pode não implicar em uma redução no tempo total do processo de identificação;
- Minimizar o erro de identificação sem considerar qual tamanho de *frame* escolher não reduz o tempo de identificação;
- Reduzir adequadamente a quantidade de *slots* em colisão através de um aumento na quantidade de *slots* vazios em cada *frame* gerado, contribui para a minimização do tempo total de identificação;

- A métrica de avaliação de desempenho mais importante é o tempo total para a identificação das etiquetas, uma vez que o tempo é o elemento que, efetivamente, afeta o usuário do sistema.

Considerando essas conclusões, é possível melhorar o tempo de identificação sem que haja perda na precisão dos estimadores. Desse modo, este trabalho apresentou uma função de cálculo de tamanho de *frames* para o DFSA que considera o impacto do uso da extensão *Early-End*. A função proposta relaciona o tamanho do próximo *frame* calculado por um algoritmo de estimação com um fator de ajuste.

Os experimentos realizados com a função proposta permitiram observar uma melhora no tempo de identificação de etiquetas. Essa redução no tempo de identificação foi de, aproximadamente, até 24%, 22% e 30% para os estimadores Eom-Lee, Vogt e Schoute, respectivamente. É importante salientar que a função proposta pode ser utilizada em qualquer estimador que possua um método para gerar o tamanho do próximo *frame* com a aplicação da metodologia apresentada nesta dissertação. Vale ressaltar que esta dissertação produziu um artigo, encontrado em [44].

Dentre os possíveis trabalhos futuros, é possível citar: evoluir o método de definição de δ_i, saindo de um método experimental para um método analítico, reduzindo o trabalho para adaptar a função proposta para outros estimadores.

Outro trabalho futuro é realizar um estudo considerando um modelo de canal mais realista, com ruído e efeito de captura (probabilidade do leitor de identificar uma etiqueta mesmo que haja colisão).

Também é possível expandir o estudo feito para ambientes com maior quantidade de etiquetas se submetendo ao processo de identificação, relacionar esse trabalho com o que foi apresentado nesta dissertação, verificando se existe a necessidade de haver múltiplos valores de δ dada uma certa quantidade de etiquetas.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] D.-H. Shih, P.-L. Sun, D. C. Yen, and S.-M. Huang, "Taxonomy and Survey of RFID Anti-collision Protocols," *Computer Communications*, vol. 29, no. 11, pp. 2150–2166, 2006.

[2] L. A. Burdet, "RFID Multiple Access Method," *Technical Report ETH*, 2004.

[3] K. H. Doerr, W. R. Gates, and J. E. Mutty, "A Hybrid Approach to the Valuation of RFID/MEMS Technology Applied to Ordnance Inventory," *International Journal of Production Economics*, vol. 103, no. 2, pp. 726 – 741, 2006.

[4] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*, 2nd ed. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[5] L. Ni, Y. Liu, Y. C. Lau, and A. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2003, pp. 407 – 415.

[6] M. L. Ng, K. S. Leong, D. Hall, and P. Cole, "A Small Passive UHF RFID Tag for Livestock Identification," in *Proceedings of the IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications (MAPE)*, vol. 1, August 2005, pp. 67 – 70 Vol. 1.

[7] M.-S. Jian, K. S. Yang, and C.-L. Lee, "Modular RFID Parking Management System Based on Existed Gate System Integration," *WSEAS Transactions on Systems (WTOS)*, vol. 7, pp. 706–716, June 2008.

[8] B. Carbunar, M. K. Ramanathan, M. Koyutürk, S. Jagannathan, and A. Grama, "Efficient Tag Detection in RFID Systems," *Journal of Parallel and Distributed Computing*, vol. 69, no. 2, pp. 180 – 196, 2009.

[9] D. W. Engels and S. E. Sarma, "The Reader Collison Problem," in *Proceedings of the International Conference on Systems, Man, and Cybernetics*, October 2002.

[10] D. Klair, K.-W. Chin, and R. Raad, "A Survey and Tutorial of RFID Anti-Collision Protocols," *IEEE Communications Surveys Tutorials*, vol. 12, no. 3, pp. 400 –421, Third Quarter 2010.

[11] J. Ho, D. W. Engels, and S. E. Sarma, "HiQ: A Hierarchical Q-Learning Algorithm to Solve the Reader Collision Problem," in *Proceedings of the International Symposium on Applications on Internet Workshops (SAINT-W)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 88–91.

[12] J. Waldrop, D. W. Engels, and S. E. Sarma, "Colorwave: An Anticollision Algorithm for the Reader Collision Problem," in *Proceedings of the IEEE International Conference on Communications (ICC)*, vol. 2. Cambridge, MA, USA: IEEE Computer Society, 2003, pp. 1206–1210.

[13] S.-R. Lee, S.-D. Joo, and C.-W. Lee, "An Enhanced Dynamic Framed Slotted ALOHA Algorithm for RFID Tag Identification," *Annual International Conference on Mobile and Ubiquitous Systems*, pp. 166–174, 2005.

[14] Y.-H. Chen, S.-J. Horng, R.-S. Run, J.-L. Lai, R.-J. Chen, W.-C. Chen, Y. Pan, and T. Takao, "A Novel Anti-Collision Algorithm in RFID Systems for Identifying

Passive Tags," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 1, pp. 105 –121, February 2010.

[15] L. Zhu and T.-S. Yum, "Optimal Framed Aloha Based Anti-Collision Algorithms for RFID Systems," *IEEE Transactions on Communications*, vol. 58, no. 12, pp. 3583 –3592, December 2010.

[16] B. Li and J. Wang, "Efficient Anti-Collision Algorithm Utilizing the Capture Effect for ISO 18000-6C RFID Protocol," *IEEE Communications Letters*, vol. 15, no. 3, pp. 352 –354, March 2011.

[17] C. Piao, Z. Fan, and X. Han, "Research on RFID-Based Mobile Logistics Monitoring Solution and Anti-Collision Algorithm for Grouped RFID Tags Identification," *IEEE International Conference on E-Business Engineering*, pp. 190–197, 2010.

[18] J.-B. Eom and T.-J. Lee, "Accurate Tag Estimation for Dynamic Framed-slotted ALOHA in RFID Systems," *IEEE Communications Letters*, vol. 14, pp. 60–62, January 2010.

[19] Q. Tong, X. Zou, and H. Tong, "Dynamic Framed Slotted ALOHA Algorithm Based on Bayesian Estimation in RFID System," in *Proceedings of the WRI World Congress on Computer Science and Information Engineering*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 384–388.

[20] W.-T. Chen, "An Accurate Tag Estimate Method for Improving the Performance of an RFID Anticollision Algorithm Based on Dynamic Frame Length ALOHA," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 1, pp. 9 –15, January 2009.

[21] S. Liu and X. Peng, "Improved Dynamic Frame Slotted ALOHA Algorithm for Anti-collision in RFID Systems," in *Knowledge Discovery and Data Mining*, ser. Advances

in Intelligent and Soft Computing, H. Tan, Ed. Springer Berlin / Heidelberg, 2012, vol. 135, pp. 423–430, 10.1007/978-3-642-27708-5_58.

[22] D.-J. Deng and H.-W. Tsao, "Optimal Dynamic Framed Slotted ALOHA Based Anti-collision Algorithm for RFID Systems," *Wireless Personal Communications*, vol. 59, pp. 109–122, 2011, 10.1007/s11277-010-0193-3.

[23] S.-C. Kim and S. K. Kim, "An Enhanced Anti-collision Algorithm for EPC Gen2 RFID System," in *5th FTRA International Conference on Multimedia and Ubiquitous Engineering (MUE)*, June 2011, pp. 293 –296.

[24] J. Vales-Alonso, V. Bueno-Delgado, E. Egea-Lopez, F. Gonzalez-Castano, and J. Alcaraz, "Multiframe Maximum-Likelihood Tag Estimation for RFID Anticollision Protocols," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 3, pp. 487 – 496, August 2011.

[25] H. Vogt, "Efficient Object Identification with Passive RFID Tags," in *Proceedings of the First International Conference on Pervasive Computing*. London, UK: Springer-Verlag, 2002, pp. 98–113.

[26] F. C. Schoute, "Dynamic Frame Length ALOHA," *IEEE Transactions on Communications*, vol. 31, pp. 565–568, April 1983.

[27] R. Want, "The Magic of RFID," *ACM Queue*, vol. 2, pp. 40–48, October 2004.

[28] B. A. Jesus, "Um Protocolo Híbrido de Anti-colisão de Etiquetas para Sistemas RFID," Master's thesis, Universidade Federal de Pernambuco, 2010.

[29] M. Kodialam and T. Nandagopal, "Fast and Reliable Estimation Schemes in RFID Systems," in *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom)*. New York, NY, USA: ACM, 2006, pp. 322–333.

[30] F. Shad, T. Todd, V. Kezys, and J. Litva, "Dynamic Slot Allocation (DSA) in Indoor SDMA/TDMA Using a Smart Antenna Basestation," *IEEE/ACM Transactions on Networking*, vol. 9, no. 1, pp. 69 –81, February 2001.

[31] P. Jung, P. Baier, and A. Steil, "Advantages of CDMA and Spread Spectrum Techniques Over FDMA and TDMA in Cellular Mobile Radio Applications," *IEEE Transactions on Vehicular Technology*, vol. 42, no. 3, pp. 357 –364, August 1993.

[32] R. Nelson and L. Kleinrock, "Spatial TDMA: A Collision-Free Multihop Channel Access Protocol," *IEEE Transactions on Communications*, vol. 33, no. 9, pp. 934 – 944, September 1985.

[33] D. R. Rush and C. Wood, "Analysis of Tree Algorithms for RFID Arbitration," in *IEEE International Symposium on Information Theory*. IEEE, 1998, p. 107.

[34] C. Law, K. Lee, and K.-Y. Siu, "Efficient Memoryless Protocol for Tag Identification," in *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*. New York, NY, USA: ACM, 2000, pp. 75–84.

[35] J.-D. Shin, S.-S. Yeo, T.-H. Kim, and S. K. Kim, "Hybrid Tag Anti-collision Algorithms in RFID Systems," in *Proceedings of the 7th International Conference on Computational Science (ICCS), Part IV*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 693–700.

[36] J. Ryu, H. Lee, Y. Seok, T. Kwon, and Y. Choi, "A Hybrid Query Tree Protocol for Tag Collision Arbitration in RFID systems," in *Proceedings of the IEEE International Conference on Communications (ICC)*, June 2007, pp. 5981 –5986.

[37] T. H. Kim and S. J. Lee, "A Hybrid Hyper Tag Anti-Collision Algorithm in RFID System," in *Proceedings of the 11th International Conference on Advanced Communication Technology (ICACT)*, vol. 02, February 2009, pp. 1276 –1281.

[38] N. Abramson, "THE ALOHA SYSTEM: Another Alternative for Computer Communications," in *Proceedings of the Fall Joint Computer Conference (AFIPS)*. New York, NY, USA: ACM, November 1970, pp. 281–285.

[39] J. F. Kurose and K. W. Ross, *Redes de Computadores e a Internet. Uma Abordagem Top-down*, 5th ed. Pearson Addison Wesley, 2010.

[40] D. K. Klair, K. wu Chin, and R. Raad, "An Investigation Into the Energy Efficiency of Pure and Slotted Aloha Bbased RFID Anti-Collision Protocols," in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2007, pp. 18–21.

[41] J.-R. Cha and J.-H. Kim, "Novel Anti-collision Algorithms for Fast Object Identification in RFID System," in *Proceedings of the 11th International Conference on Parallel and Distributed Systems*, vol. 2, July 2005, pp. 63 –67.

[42] H. Wu and Y. Zeng, "Bayesian Tag Estimate and Optimal Frame Length for Anti-Collision Aloha RFID System," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 4, pp. 963 –969, October 2010.

[43] EPC Global Inc., *EPC Radio-Frequency Identity Protocols Class-1 Generation 2 UHF RFID Protocol for Communications at 860 MHz - 960 MHz*, 1st ed., 2008.

[44] J. Andrade and P. Gonçalvez, "Uma Função de Cálculo de Tamanho de Frames para o Protocolo DFSA em Sistemas RFID," in *XVI Workshop de Gerência e Operação de Redes e Serviços (WGRS)*, Campo Grande, MS, Brazil, May 2011, pp. 61–74.

[45] S. K. Park and K. W. Miller, "Random Number Generators: Good Ones Are Hard to Find," *Communications of the ACM*, vol. 31, pp. 1192–1201, October 1988.

[46] P. L'Ecuyer, "Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure," *Mathematics of Computation*, vol. 68, pp. 249–260, January 1999.

[47] D. E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997, vol. 2.

[48] G. Marsaglia and W. W. Tsang, "The 64-bit Universal RNG," *Statistics & Probability Letters*, vol. 66, no. 2, pp. 183 – 187, 2004.

[49] G. Marsaglia, "The Marsaglia Random Number CDROM Including the DIEHARD: A Battery of Tests of Randomness," 1996, see http://stat.fsu.edu/∼geo/diehard.html.

APÊNDICE A

FERRAMENTA DE SIMULAÇÃO

Para a realização dessa dissertação, conforme mencionado, foi desenvolvida uma ferramenta de simulação para sistemas RFID em C++. A linguagem C++ foi escolhida por prover um bom desempenho em termos computacionais e utilizar o paradigma de orientação a objeto. O simulador desenvolvido é multiplataforma, com suporte a *threads* e depende das bibliotecas *boost*[1]. A ferramenta, de modo geral, simula um dado protocolo anticolisão para sistemas RFID, um estimador (se for aplicável), para um intervalo de etiquetas.

Genericamente, o simulador define as seguintes opções de configuração: tamanho do identificador das etiquetas em *bits*; quantidade de *threads* que serão utilizadas na simulação, otimizando assim o tempo de simulação; se o canal irá possuir ruído, ou seja, se o canal é ideal ou se existe probabilidade de a mensagem enviada se perder tanto na transmissão leitor-etiqueta quanto na transmissão etiqueta-leitor; probabilidade de ruído; banda disponível no sentido *downstream* (leitor-etiqueta) e *upstream* (etiqueta-leitor); quantidade de vezes que cada caso de simulação será executado; quantidade inicial de etiquetas, quantidade final de etiquetas e o passo que será incrementado à quantidade de etiquetas, gerando assim um intervalo de simulação.

Também é possível estender com relativa simplicidade quais os protocolos podem ser simulados que, por sua vez, adicionam novas funcionalidades e configurações ao simulador, como por exemplo, no caso do DFSA, foi adicionada a capacidade de escolher estimadores,

[1]Biblioteca C++ BOOST, versão 1.43.0, encontrada em http://www.boost.org e desenvolvida por BOOST.ORG

tamanho do *frame* inicial e restringir o tamanho do *frame* que os estimadores fornecem sem ser necessário alterar o "código base".

Ao se trabalhar com simulações, é de vital importância considerar como os números aleatórios são gerados. Note que os números aleatórios contidos na maioria das linguagens de programação são na verdade números pseudo-aleatórios, pois dada as sementes e a fórmula de geração, o próximo número pode ser previsto. Apesar disso, esses números atendem a critérios estatísticos e podem ser utilizados como se fossem verdadeiramente aleatórios. Ressaltados esses fatos, os geradores de números pseudo-aleatórios são referenciados simplesmente por geradores de números aleatórios.

A linguagem C++ utiliza um gerador clássico conhecido com GLC (Gerador de Congruência Linear). O GLC é simples de ser implementado e possui baixo custo computacionai, mas é um gerador considerado fraco pela literatura [45, 46, 47] e o padrão C++ em vigor, atualmente, garante apenas um período de 32767 (2^{15}).

Considerando a importância de um bom gerador de números aleatórios e as fraquezas do GLC, foi implementado, no simulador, um gerador de números aleatórios diferente: o gerador de Marsaglia de 64 *bits* [48]. O gerador de Marsaglia é um gerador que fornece números entre $[0, 1)$ com período máximo de, aproximadamente 2^{202} ou 10^{61}. O gerador de Marsaglia é um gerador comprovadamente bom, passando em todos os testes estatísticos *DIEHARD* [49] com um custo computacional próximo ao GLC.

Para garantir uma melhor inicialização do gerador de números aleatórios, foram implementados *timers* com precisão de microssegundos. Esse nível de precisão ainda não faz parte do padrão atual da linguagem C++ e depende diretamente do sistema operacional que está sendo utilizado. Desse modo, esses *timers* foram implementados utilizando as APIs do Windows *QueryPerformanceFrequency() e QueryPerformanceCounter()* e a API do Linux *gettimeofday()*. Note que todos os sistemas operacionais modernos fornecem uma API com esse nível de precisão, logo, esses *timers* podem ser facilmente portados para outros sistemas operacionais.

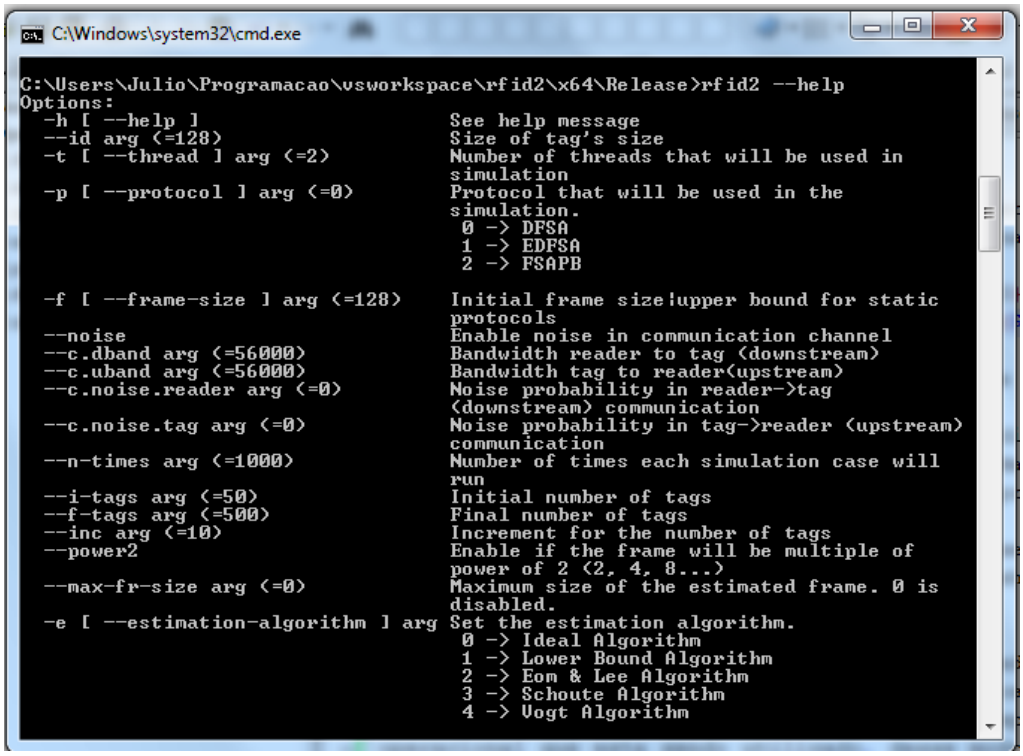


Figura A.1: Tela de ajuda do simulador.

Nas Seções A.1 e A.2 serão mostradas, respectivamente, como um usuário pode utilizar o simulador e quais os resultados gerados e como um desenvolvedor pode estender as funcionalidades existentes.

A.1 VISÃO DO USUÁRIO

A iteração com o simulador, no momento, é feita através de uma CLI (Command Line Interface - Interface por Linha de Comando), que pode ser vista na Figura A.1. Esse método de entrada foi escolhido para facilitar simulações em *batch*, através de *scripts*. Toda a interface está em inglês. Note que essa interface pode ser facilmente substituída, como será mostrado na Seção A.2.

As opções de configuração são as seguintes, onde *(número)* representa um número inteiro, *(float)* um número de ponto flutuante e *(flag)* habilita certa característica na

simulação:

–**help** Ajuda;

–**id** Tamanho do identificador das etiquetas, em *bits* (número);

-t Quantidade de *threads* que o programa criará, é recomendável que esse número não seja maior que a quantidade de *threads* de execução do processador (número);

-p Protocolo a ser simulado (número);

-f Tamanho do *frame* inicial (número);

–**noise** Se a simulação irá considerar ruído no canal (flag);

–**c.dband** Banda *downstream*, em bps (número);

–**c.uband** Banda *upstream*, em bps (número);

–**c.noise.reader** Probabilidade de ruído para comunicação *downstream* (float no intervalo $[0, 1]$);

–**c.noise.tag** Probabilidade de ruído para comunicação *upstream* (float no intervalo $[0, 1]$);

–**i-tags** Quantidade inicial de etiquetas (número);

–**f-tags** Quantidade final de etiquetas (número);

–**inc** Passo que será incrementada a quantidade inicial de etiquetas até que seja atingido a quantidade final de etiquetas (número);

–**power2** Restringe o tamanho do *frame* gerado pelos estimadores a ser sempre potência de dois, sendo o tamanho do *frame* sendo "arredondado" para a maior potência de dois mais próxima (flag);

–**max-fr-size** Restringe o tamanho do *frame* gerado pelos estimadores a um tamanho máximo (número);

-e Código do algoritmo de estimativa a ser utilizado (número).

Uma vez que a simulação foi configurada, o simulador gera um arquivo de saída com o seguinte padrão: "**nome do protocolo-itamanho do id_quantidade de etiquetas inicial-quantidade de etiquetas final-passo_nome do estimador**.rfidata". Por exemplo, se for feita uma simulação utilizando o DFSA, com 128 *bits* de ID para as etiquetas, com uma quantidade inicial de 100 etiquetas, quantidade final de 1000 etiquetas, passo de 200 etiquetas e o estimador de Vogt, se obtém o seguinte nome de saída: **dfsa-i128_100-1000-200_vogt.rfidata**.

Apesar de criar uma nova extensão para o arquivo, esse é um arquivo de texto, que pode ser aberto pro programas como o bloco de notas no Windows ou o gedit no Linux. Esse arquivo possui o resultado da simulação, onde cada linha representa uma simulação e as colunas representam, respectivamente, a quantidade de etiquetas, quantidade de *frames* utilizados na simulação, quantidade de *slots* vazios, quantidade de *slots* bem-sucedidos, quantidade de *slots* em colisão, tempo de identificação, eficiência, menor erro de estimativa normalizado, média do erro de estimativa normalizado, maior erro de estimativa normalizado, erro de estimativa absoluto do primeiro *frame*, menor erro de estimativa absoluto, média do erro de estimativa absoluto, maior erro de estimativa absoluto e eficiência temporal.

O arquivo de saída contém todos os dados da simulação. Desse arquivo, é possível fazer as análises necessárias.

A.2 VISÃO DO DESENVOLVEDOR

Um desenvolvedor pode adicionar ou estender o conteúdo na ferramenta com relativa facilidade. No momento, é possível estender, alterar ou adicionar as seguintes opções: UI (User Interface, Interface do Usuário), protocolos e estimadores. Protocolos, por sua vez, envolvem estender etiquetas, leitor, métodos de comunicação e controle.

A.2.1 Interface de Usuário

A criação de uma UI envolve criar e interagir com um *SimulationEnviroment*. Um *SimulationEnviroment* é a classe que gerencia as *threads*, isto é, define que *thread* irá executar que condição de simulação, bem como alocação de espaço para os resultados. Assim que essa classe é criada, praticamente toda a memória necessária é alocada imediatamente e a simulação é iniciada através do método *start()*. Uma vez que a simulação iniciou, é possível interagir com a classe através de dois métodos: *wait_simualtion_end()*, que bloqueia a *thread* da UI até que a simulação encerre ou *get_simulation_missing_steps()*, onde esse método retorna quantas simulações estão faltando para que a simulação encerre. Note que a utilização deste passa a responsabilidade de evitar que a instância criada não seja removida previamente para o desenvolvedor. O exemplo da arquitetura pode ser visto na Figura A.2. Após o final da simulação, é possível chamar o método *record_results()*, que grava os resultados num arquivo passado como parâmetro.

A.2.2 Protocolos

A extensão de um protocolo envolve, como mencionado, etiquetas, leitor, controle e comunicação.

Estender controles está diretamente relacionado a como controlar a etiqueta. Esse controle se dá através de *flags* da etiqueta. Para que se estenda essas *flags* é necessário

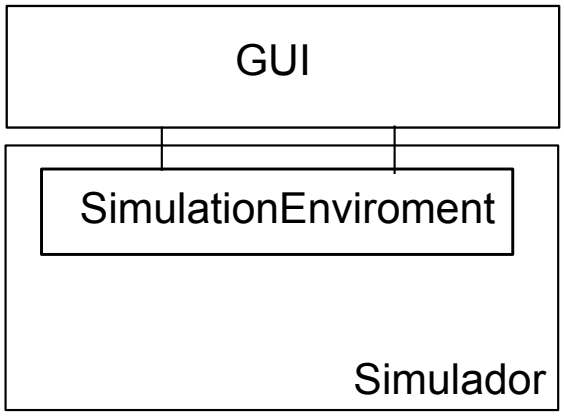


Figura A.2: Arquitetura para extensão da GUI.

implementar uma classe que herda da classe *TagFlags*, onde a classe que está herdando precisa registrar as *flags* através do método *register_flag()*, que recebe um identificar da *flag* e um valor inicial que pode ser ativada ou desativada.

Uma vez que os *flags* da etiqueta estão completos, é necessário criar uma classe que mapeie como as etiquetas devem se comportar ao receber mensagens do leitor. Essa classe precisa implementar o seguinte método: *void process_message(rfid::Event &event_recorder, rfid::RandomNumber *rng, rfid::TagFlags &flags, const rfid::Id &id, const rfid::Parameters *parameters)*, onde *event_recorder* é onde devem ser "gravadas" as respostas da etiqueta ao leitor; *rng* é o gerador de números aleatórios, para sortear um *slot* num *frame*, por exemplo; *flags* é onde as etiquetas que foram registradas, previamente, podem ser consultadas ou alteradas; *id* é o identificador da etiqueta, que pode ser utilizado para verificar se uma etiqueta pertence a algum grupo e *parameters* são os parâmetros que foram passados do leitor para a etiqueta, como por exemplo, o tamanho do *frame* para que as etiquetas possam escolher um *slot*.

Quando essas duas classes estão concluídas, elas são passadas como parâmetro para a classe *Tag*, conforme a seguinte construção: *template<class ProcessBehavior, class FlagsType> class Tag*, onde *ProcessBehavior* é a classe que implementou o método *process_message()* e *FlagsType* é a classe que herdou de *TagFlags*.

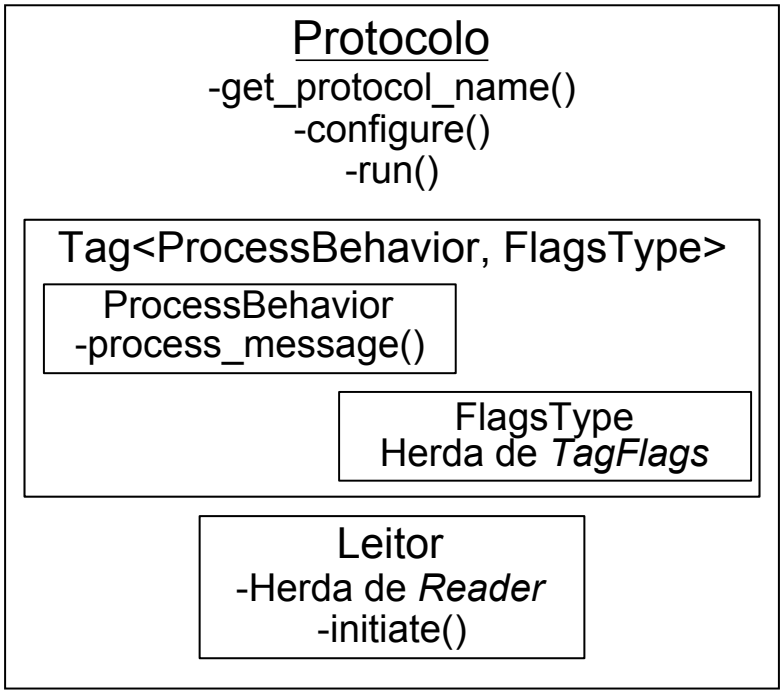


Figura A.3: Arquitetura para extensão de protocolos.

A partir desse momento, é necessário estender o leitor. Note que os protocolos são controlados pelo leitor, logo, é aqui onde vai existir o funcionamento do protocolo, como critérios de parada da simulação, quando e quais as mensagens devem ser enviadas às etiquetas, etc. A extensão do leitor é conseguida através da herança da classe *Reader*. A classe que herda precisa implementar o método *initiate()*, que inicia a execução do protocolo e deve ser finalizado apenas quando um critério de parada previamente definido é atingido. Vale salientar que a comunicação com as etiquetas deve ser feita através dos métodos *send_unicast* e *send_broadcast* providos pela classe *Reader*, pois desse modo a execução em um canal com ruído e um canal sem ruído se torna transparente para o desenvolvedor.

Finalmente, é necessário criar mais uma classe que implementa os seguintes métodos:

- const std::string get_protocol_name(void) const;
- void configure(const int &number_tags);

- void run(Result &result)

Esses métodos fazem uma interface com a classe *SimulationEnviroment* e servem, respectivamente, para consultar o nome do protocolo, informar quantas etiquetas estão participando do processo de identificação e informar a instância onde os resultados devem ser gravados. O exemplo da arquitetura pode ser visto na Figura A.3.

APÊNDICE B

PROVA DE VOGT

Como dito anteriormente, quando a quantidade de *slots* em colisão é igual ao tamanho do *frame* é necessário saber se a função, mantém o comportamento encontrado, isto é, a medida que a quantidade de etiquetas aumenta, a função decresce até o valor 0, ou seja, é impossível encontrar a quantidade de etiquetas que minimiza a função.

Desse modo, temos que encontrar o limite da função, para o caso de todos os *slots* estarem em colisão - $s_v = 0$, $s_s = 0$ e $s_c = L$ - quando n tende a infinito. Logo, temos que achar o resultado da seguinte equação:

$$\lim_{n\to\infty} \left(\varepsilon(L, s_v, s_s, s_c) = \left| \begin{pmatrix} a_0^{L,n} \\ a_1^{L,n} \\ a_{\geq 2}^{L,n} \end{pmatrix} - \begin{pmatrix} s_v \\ s_s \\ s_c \end{pmatrix} \right| \right) \quad . \tag{B.1}$$

Expandindo a equação (B.1), temos:

$$\varepsilon(L, 0, 0, L) = \sqrt{\left(a_0^{L,n} - 0\right)^2 + \left(a_1^{L,n} - 0\right)^2 + \left(a_{\geq 2}^{L,n} - L\right)^2} \quad , \tag{B.2}$$

$$\lim_{n\to\infty} (\varepsilon) = \lim_{n\to\infty} \left(\sqrt{\left(a_0^{L,n}\right)^2 + \left(a_1^{L,n}\right)^2 + \left(a_{\geq 2}^{L,n} - L\right)^2} \right) \quad . \tag{B.3}$$

Aplicando as propriedades de limites na Equação (B.3), é obtido:

$$\lim_{n\to\infty}(\varepsilon) = \sqrt{\left(\lim_{n\to\infty}\left(a_0^{L,n}\right)\right)^2 + \left(\lim_{n\to\infty}\left(a_1^{L,n}\right)\right)^2 + \left(\lim_{n\to\infty}\left(a_{\geq 2}^{L,n} - L\right)\right)^2} \quad . \tag{B.4}$$

Logo, temos que resolver as seguintes equações:

$$\lim_{n\to\infty}\left(a_0^{L,n}\right) \quad , \tag{B.5}$$

$$\lim_{n\to\infty}\left(a_1^{L,n}\right) \quad , \tag{B.6}$$

$$\lim_{n\to\infty}\left(a_{\geq 2}^{L,n} - L\right) \quad . \tag{B.7}$$

Expandindo e aplicando novamente as propriedades de limites na Equação (B.5):

$$\lim_{n\to\infty}\left(a_0^{L,n}\right) = \lim_{n\to\infty} L\left(1 - \frac{1}{L}\right)^n = L\lim_{n\to\infty}\left(\frac{L-1}{L}\right)^n \quad . \tag{B.8}$$

Como pode ser visto na Equação (B.8), o denominador cresce mais rápido do que o numerador, logo, a parte do limite da Equação (B.8) possui o seguinte resultado:

$$\lim_{n\to\infty}\left(\frac{L-1}{L}\right)^n = 0 \quad . \tag{B.9}$$

Substituindo o valor encontrado na Equação (B.9) na Equação (B.8), temos que o resultado da Equação (B.5) é:

$$\lim_{n\to\infty}\left(a_0^{L,n}\right) = 0 \quad . \tag{B.10}$$

Aplicando mesmo conceito a Equação (B.6), é obtido:

$$\lim_{n\to\infty}\left(a_1^{L,n}\right) = \lim_{n\to\infty} n\left(1 - \frac{1}{L}\right)^{n-1} \quad . \tag{B.11}$$

Reescrevendo a Equação (B.11) e aplicando a regra de L'Hôpital é obtida a seguinte

equação:

$$\lim_{n\to\infty} n\left(1-\frac{1}{L}\right)^{n-1} = \lim_{n\to\infty} \frac{[n]'}{\left[\frac{1}{\left(1-\frac{1}{L}\right)^{n-1}}\right]'} = \lim_{n\to\infty} \frac{1}{-\left(1-\frac{1}{L}\right)^{1-n}\cdot \ln\left(1-\frac{1}{L}\right)} \quad . \tag{B.12}$$

Assim como na Equação (B.8), o denominador da Equação (B.12) cresce mais rápido do que seu numerador, logo, temos que o resultado do segundo limite é igual a 0, ou seja,

$$\lim_{n\to\infty}\left(a_1^{L,n}\right) = 0 \quad . \tag{B.13}$$

Finalmente, temos o último limite a ser calculado, na Equação (B.7). Expandindo esta equação e aplicando as propriedades dos limites, é obtido:

$$\begin{aligned}\lim_{n\to\infty}\left(a_{\geq 2}^{L,n} - L\right) &= \lim_{n\to\infty}\left(L - a_0^{L,n} - a_1^{L,n} - L\right) \\ &= -1\cdot\lim_{n\to\infty}\left(a_0^{L,n}\right) - 1\cdot\lim_{n\to\infty}\left(a_1^{L,n}\right)\end{aligned} \quad . \tag{B.14}$$

É possível notar que os limites necessários já foram calculados e podem ser encontrados nas Equações (B.10) e (B.13), respectivamente. Substituindo os valores encontrados na Equação (B.14), temos:

$$\lim_{n\to\infty}\left(a_{\geq 2}^{L,n} - L\right) = -1\cdot 0 - 1\cdot 0 = 0 \quad . \tag{B.15}$$

De posse de todos os valores calculados (Equações (B.10), (B.13), (B.15)) e substituindo-os na Equação (B.4), temos:

$$\lim_{n\to\infty}(\varepsilon) = \sqrt{(0)^2 + (0)^2 + (0)^2} = 0 \quad . \tag{B.16}$$

Desse modo, fica provado que, para o caso de de todos os *slots* em colisão, o estimador de Vogt tende a zero a medida que a quantidade de etiquetas tende a infinito.