Um ambiente de simulação de algoritmos anticolisão para apoio ao desenvolvimento de aplicações de RFID

José Luís Sorokin[1]
Flavius Portella Ribas Martins[2]

Resumo: Dentre os aspectos técnicos a serem considerados no projeto de um sistema baseado em *Radio Frequency Identification* (RFID) um dos mais importantes é a análise do processo de colisão de informações, ou seja, a sucessão de eventos que ocorrem quando etiquetas RFID adentram a zona de abrangência de um dispositivo leitor e transmitem suas informações simultaneamente, dificultando sua correta identificação. Para evitar os efeitos danosos que seriam causados por tais eventos é necessário integrar aos dispositivos transmissores e receptores um algoritmo – denominado 'anticolisão' – responsável pelo isolamento da comunicação entre cada etiqueta e o leitor. Neste trabalho propõe-se o desenvolvimento de um ambiente de simulação de algoritmos anticolisão para sistemas RFID que permite sua análise comparativa em situações variadas, facilitando a avaliação de desempenho desses algoritmos e sua subsequente seleção, de modo a atender às características da aplicação pretendida. Também são apresentados alguns experimentos realizados com os algoritmos anticolisão implantados no ambiente de simulação e analisados à luz das métricas de desempenho adotadas.

Palavras-chave: *Algoritmo anticolisão. ALOHA. Árvore binária. RFID.*

Abstract*: A crucial task in the project of Radio Frequency Identification (RFID) systems is to cope with information collision, an undesirable event caused by the attempt of two or more RFID tags simultaneously respond to an interrogation command issued by a data collector. To avoid the deleterious effects that such events could provoke, it is necessary to include an anticollisionalgorithm in the transmitters and receivers communication softwares. Considering the impact of those algorithms in the performance of an RFID system, we present, in this article, a simulation environment which allows a comparative analysis of different anticollision algorithms, easing the selection and subsequent setting up of the ones that meet the application demands. Some experiments performed with the implemented anticollision algorithms are also presented and analyzed according to the adopted performance metrics.*

Keywords: *Anticollision algorithm. ALOHA. Binary tree. RFID.*

1 Introdução

Um sistema baseado em *Radio Frequency Identification* (RFID) possui dois componentes essenciais – o dispositivo móvel leitor e as etiquetas identificadoras dos itens de interesse. Dessa forma, ao entrar na zona de interrogação de uma etiqueta e realizar a leitura de seu código e das demais informações nela eventualmente armazenadas, o leitor obtém apontadores que são utilizados para o acesso a uma base de dados central de onde se

[1] Instituto de Pesquisas Tecnológicas do Estado de São Paulo, Av. Prof. Almeida Prado 532, São Paulo (SP) - Brasil
{jls@ipt.br}
[2] Departamento de Engenharia Mecânica, Escola Politécnica da Universidade de São Paulo, Av. Prof. Mello Moraes, 2231, São Paulo (SP) - Brasil
{flavius.martins@poli.usp.br}

http://dx.doi.org/10.5335/rbca.2012.2233

extraem informações que possam alimentar os mecanismos de decisão da aplicação relativos ao item identificado.

Os sistemas baseados em RFID caracterizam-se por possibilitarem a identificação do objeto sem necessidade de contato mecânico e a uma distância que pode variar de alguns centímetros a vários metros. Todavia, quando várias etiquetas presentes na área de interrogação do leitor transmitem suas informações na mesma faixa de frequências e no mesmo intervalo de tempo, ocorrem interferências que ocasionam a perda de informações. Tal fenômeno é denominado 'colisão' [1] e as dificuldades dele decorrentes são contornadas pelos assim denominados algoritmos anticolisão.

De acordo com o método de processamento de sinais adotado, esses algoritmos podem ser classificados em algoritmos de domínio espacial, de domínio de frequências e de domínio temporal [2].

Os algoritmos de domínio espacial são baseados no SDMA (*Space Division Multiple Access*), um método de acesso que permite o uso simultâneo da mesma faixa de frequências em diferentes células, de acordo com o raio de ação da onda emitida. Para tanto, é necessário utilizar uma antena direcional e dispor de um método para determinação da distância entre a etiqueta e o leitor a partir da variação da energia transferida. Alternativamente, pode-se restringir a zona de abrangência do leitor e aumentar o número de leitores, solução essa que possui como desvantagem o elevado custo de implantação [3].

Os algoritmos de domínio da frequência podem utilizar tanto o *Frequency Division Multiple Access* (*FDMA*) quanto o *Code Division Multiple Access* (CDMA). No primeiro caso, divide-se o espectro de frequências em canais de largura fixa e relativamente estreita, permitindo-se que cada dispositivo tenha acesso a um canal separado para efetuar sua comunicação. O segundo método utiliza o mesmo canal de transmissão com um código associado a cada dispositivo, de modo que somente os componentes que utilizam o mesmo código conseguem se comunicar.

Os algoritmos de domínio do tempo baseiam-se no método *Time Division Multiple Access* (TDMA), que consiste no compartilhamento de um único canal de rádio dividido em pequenos intervalos de tempo, denominados *slots*. Devido à sua simplicidade, esse é o método de processamento mais utilizado em sistemas baseados em RFID.

Nas aplicações mais comuns de RFID existem poucas etiquetas na área de interrogação do leitor. Todavia, observa-se uma tendência de que as aplicações requeiram a leitura de um número cada vez maior de etiquetas. Tal é o caso, por exemplo, da identificação de produtos em cadeias de lojas de vendas a varejo (supermercados, tipicamente) [4] ou da extração de informações de equipamentos de rede de distribuição elétrica (transformadores, chaves de religação etc) durante os eventos de expedição, transporte, instalação, remoção e manutenção [5]. Naturalmente, nessas situações, é necessário que sejam embarcados algoritmos anticolisão tanto nas etiquetas quanto nos coletores de dados.

As razões apresentadas acima mostram que a escolha de um algoritmo anticolisão eficiente é uma etapa crucial para se garantir o bom desempenho de uma aplicação de RFID. Todavia, graças ao grande número de propostas existentes na literatura para esse tipo de algoritmo, e das naturais dificuldades logísticas para se testá-los durante a fase de desenvolvimento da aplicação, a construção de um ambiente de simulação que permita ao projetista analisar seu desempenho face à escolha de variados tipos de algoritmo anticolisão é uma necessidade concreta do segmento de desenvolvimento de software para sistemas baseados em RFID. Tal é, portanto, a motivação do presente artigo, cuja estrutura é a seguinte: na sessão 2 apresentam-se, sumariamente, os algoritmos anticolisão baseados no método TDMA mais utilizados em aplicações RFID; na sessão 3, descreve-se a metodologia adotada para a construção de uma ferramenta de simulação virtual de algoritmos anticolisão; na sessão 4, apresentam-se as principais características da ferramenta projetada; na sessão 5, aplica-se essa ferramenta à seleção de um algoritmo anticolisão para um sistema RFID de identificação de equipamentos elétricos; na sessão 6 apresentam-se as conclusões finais do trabalho.

É possível encontrar na literatura diversos exemplos de ambientes de simulação. Nos "proceedings of the 9th International Conference on Advanced Communication Technology" [1], o autor desenvolveu um simulador utilizando a linguagem C# para realizar a análise de algoritmos árvore binária e ALOHA e propor uma variação mais eficaz desse tipo de algoritmo. Em "RFID-Env: methods and software simulation for RFID environments" [18] é apresentado um simulador que contempla uma biblioteca de algoritmos abrangendo o padrão ISO 18000, desenvolvida em linguagem JAVA. A despeito de compartilhar certas características desses ambientes,

apresenta-se neste artigo uma biblioteca de componentes, que facilita a implementação de um novo algoritmo mediante a reutilização de códigos previamente cadastrados.

2 Algoritmos anticolisão de domínio temporal

Esses algoritmos asseguram que as etiquetas efetuem a transmissão em intervalos de tempos previamente definidos, de modo a diminuir o número de colisões [6]. Dentro dessa categoria, existem duas classes de algoritmos – os determinísticos, baseados em busca em árvores binárias, e os probabilísticos, baseados no protocolo ALOHA [7]. Todavia, pesquisas recentes sobre esse tema têm acrescentado importantes melhorias a esses algoritmos básicos, dando origem a variações como, por exemplo, o algoritmo com retrocesso baseado em busca binária [8] e o algoritmo probabilístico baseado em separação em grupos [9].

Assim que uma etiqueta entra na zona de leitura, o algoritmo *ALOHA* transmite os seus dados ao leitor e, após um intervalo aleatório de tempo, realiza a retransmissão. Se não houver colisão, o leitor envia um comando à etiqueta confirmando a identificação e evitando, com isso, posteriores retransmissões. Esse processo é repetido até que todas as etiquetas que se encontram na zona de leitura sejam identificadas. Devido à sua simplicidade, é relativamente grande a probabilidade de que ocorram intersecções temporais nos processos de transmissão de duas ou mais etiquetas. Por tal motivo propuseram-se versões mais aperfeiçoadas desse algoritmo – *Slotted ALOHA, Framed Slotted ALOHA e Dynamic Framed Slotted ALOHA* – em que se busca diminuir a probabilidade de ocorrência de colisões mediante a criação de *slots*, ou seja, intervalos fixos e pré-estabelecidos de tempo para a conclusão de um processo de comunicação.

No algoritmo *Slotted ALOHA,* as etiquetas somente podem iniciar a transmissão após a passagem de um número inteiro de *slots*, definido a priori pelo leitor. Embora essa versão apresente desempenho superior ao do algoritmo *ALOHA* original, ainda assim permite a ocorrência de muitas colisões. Em uma versão subsequente – o *Framed Slotted ALOHA*, os *slots* são organizados em intervalos denominados *frames*, meio aos quais as etiquetas podem realizar uma única transmissão.

O grande desafio dessa classe de algoritmos é determinar o número ideal de *slots* para um dado número de etiquetas. Com poucos *slots* gera-se um grande número de colisões, pois as etiquetas na zona de leitura têm poucas oportunidades para a transmissão; por outro lado, se for grande o número de *slots*, diminui-se a probabilidade de colisões, mas aumenta-se significativamente o tempo de espera. Visando à solução dessas dificuldades, desenvolveu-se o algoritmo *Dynamic Framed Slotted* ALOHA, no qual o tamanho do *frame* é modificado dinamicamente a cada ciclo, de acordo com vários métodos propostos na literatura [10-11], de modo a maximizar a taxa de identificação correta de etiquetas.

É importante salientar que existe uma grande variedade de outros algoritmos anticolisão probabilísticos, como, por exemplo, o algoritmo *Btree*, proposto na norma ISO 18000-6 B [12], e que utiliza conceitos similares aos do ALOHA, como *slots* e seleção aleatória do momento da transmissão.

Os algoritmos anticolisão determinísticos, por sua vez, descrevem os códigos de identificação das etiquetas segundo uma árvore binária: no nível zero armazenam-se os bits de ordem *n* do identificador; no nível 1 os bits de ordem *n*-1 e assim por diante, até o nível *n*, onde se armazenam os bits de ordem 0. Inicialmente, o leitor envia um prefixo a todas as etiquetas – por exemplo, '0'. Cada etiqueta verifica se esse prefixo corresponde ao seu bit de ordem *n*; em caso afirmativo, seu código completo é enviado ao leitor. Se apenas uma etiqueta enviar a resposta, o processo de identificação estará concluído, mas se ocorrer colisão (ou, seja, duas ou mais etiquetas possuírem o mesmo prefixo e tentarem enviar sua identificação), o leitor acrescenta um novo bit ao prefixo anteriormente enviado. Esse processo se repete até que uma única etiqueta responda ao comando. Durante a travessia da árvore, o algoritmo pode identificar os nós abaixo dos quais não existe nenhuma etiqueta, e, assim, diminuir o tempo de busca, melhorando significativamente o seu desempenho [13].

3 Materiais e métodos

Para que o processo de construção da ferramenta de apoio à seleção de algoritmos anticolisão atendesse aos requisitos estabelecidos em sua proposição, adotou-se o modelo sequencial linear da engenharia de software [14], que se inicia com a 'definição de requisitos', passa às etapas de 'projeto de sistema e software',

‘implementação e teste de unidade’, ‘integração e teste de sistema’ e finaliza com a etapa de ‘operação e manutenção’.

Nos tópicos que seguem, descrevem-se os procedimentos e resultados associados às etapas do processo de desenvolvimento de software mencionadas acima.

3.1 Etapa de definição de requisitos

Devido ao fato de não haver um cliente específico para a realização do levantamento de requisitos, essa etapa foi realizada através de pesquisas em trabalhos similares, observando-se as entradas e as saídas de dados bem como as funcionalidades necessárias para a realização das simulações.

Estabeleceu-se que o ambiente de simulação deveria atender aos seguintes requisitos: 1) possuir uma biblioteca de algoritmos anticolisão passível de ser modificada pelo usuário mediante inclusão, alteração ou exclusão de algoritmos; 2) simular a execução de algoritmos anticolisão a partir de valores de parâmetros de entrada fornecidos pelo usuário; 3) cadastrar características de hardware dos componentes (leitores e etiquetas RFID); 4) cadastrar normas técnicas associadas aos respectivos algoritmos anticolisão; 5) efetuar comparações entre os ensaios realizados baseadas nos índices de desempenho dos algoritmos cadastrados; 6) gerar o pseudocódigo de cada algoritmo cadastrado, com o objetivo de facilitar sua implantação em leitores e etiquetas RFID.

No diagrama de casos de uso da Fig. 1, as principais missões da ferramenta de apoio ao projeto de algoritmos anticolisão são explicitadas.

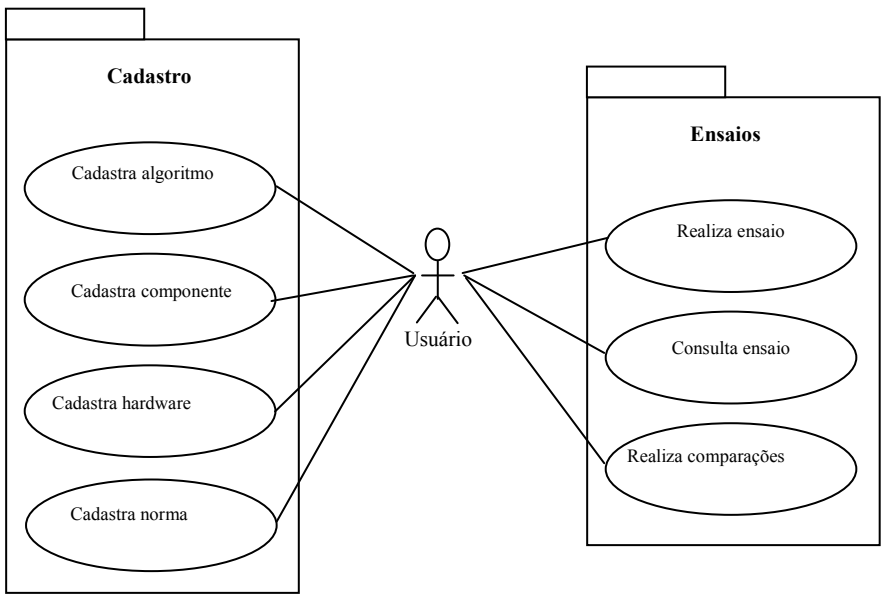


Figura 1: Diagrama de casos de uso.

3.2 Etapa de projeto de sistema e software

A partir dos requisitos levantados na etapa anterior realizaram-se as ações de projeto voltadas à representação estática e dinâmica do sistema, utilizando-se, para a primeira, o diagrama de classes (Fig. 2) e o modelo entidade-relacionamento (Fig. 3) e, para a segunda, os diagramas de sequência (Fig.4), que identificam quais métodos devem ser disparados entre os atores e os objetos envolvidos e em qual ordem.

É importante enfatizar que os objetos utilizados na base de dados do sistema foram representados de acordo com o modelo entidade-relacionamento elaborado e adotando-se a abordagem IDEF1X (*Integration Definition for Information Modeling*) [15] que utiliza uma linguagem e um método para modelagem de informação a partir de modelos entidade-relacionamento.

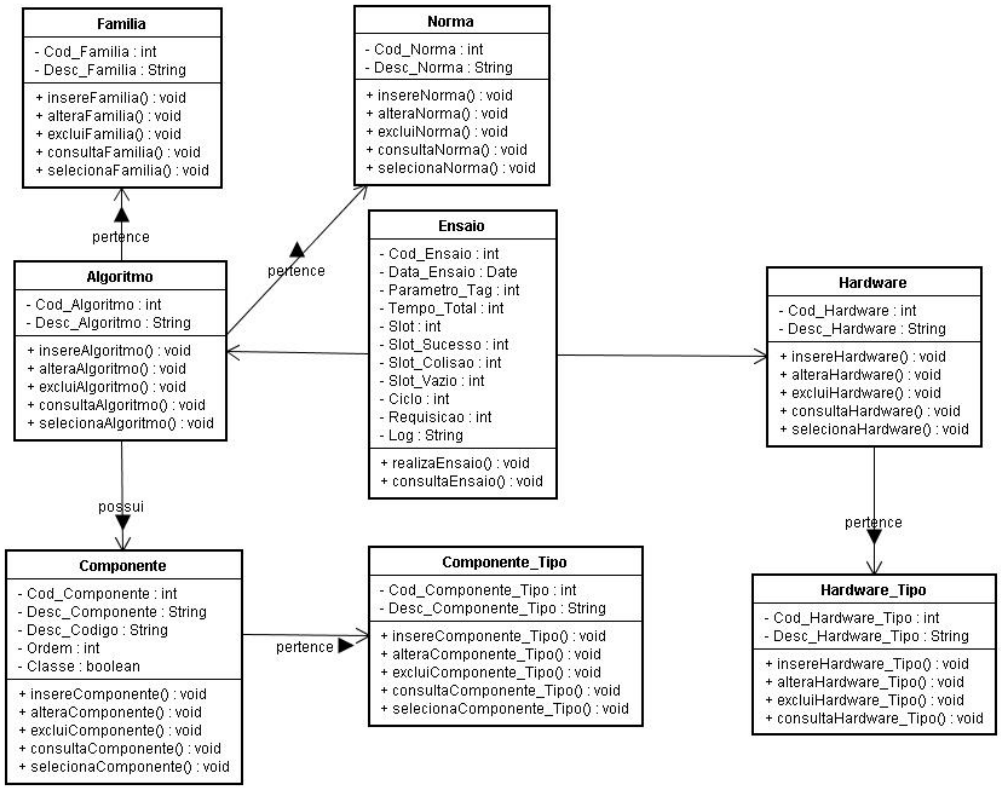


Figura 2: Diagrama de classes.

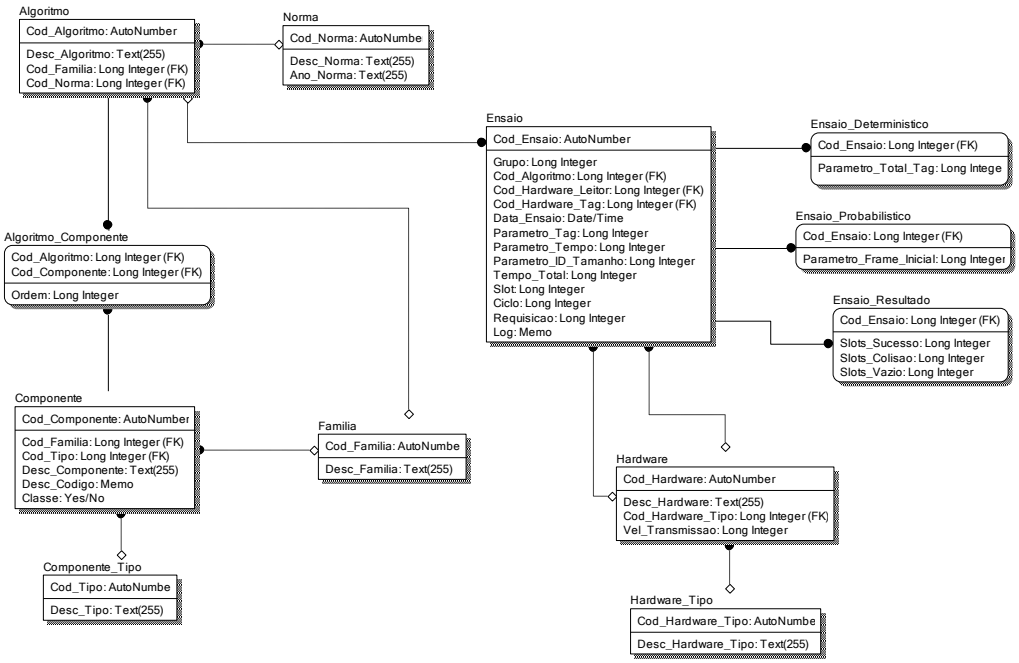


Figura 3: Modelo entidade-relacionamento.

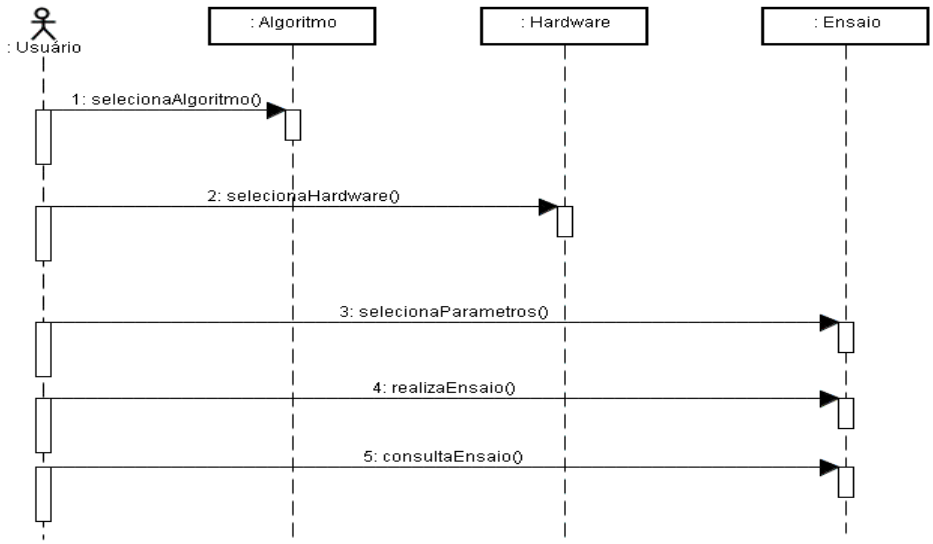


Figura 4: Diagrama de sequência 'Realiza ensaio'.

3.3 Etapa de implementação

O projeto do sistema protótipo do ambiente de simulação foi desenvolvido utilizando-se a ferramenta *Microsoft Studio* 2008 e a linguagem C# e adotando-se como requisito a necessidade de se compilar e gerar pseudocódigo a partir do trechos de código cadastrados na biblioteca de algoritmos anticolisão. Para que isso fosse viável fez-se necessário utilizar o componente *CodeDOM*, do *Microsoft Framework*, que possui funções para o gerenciamento da compilação e execução de código escrito em C#, *Visual Basic*, C++, J# e *JavaScript*.

Foram incluídas duas classes responsáveis pelo provimento dos recursos de infraestrutura do aplicativo – a classe <dbRFID> para o estabelecimento da comunicação de todas as demais classes do aplicativo com a base de dados e a classe <cpRFID> para a compilação e execução do código dos algoritmos anticolisão cadastrados.

Na Figura 5 apresenta-se o diagrama de componentes da ferramenta de apoio à seleção e configuração de algoritmos anticolisão, onde se observam os componentes utilizados na construção do software e seus relacionamentos.

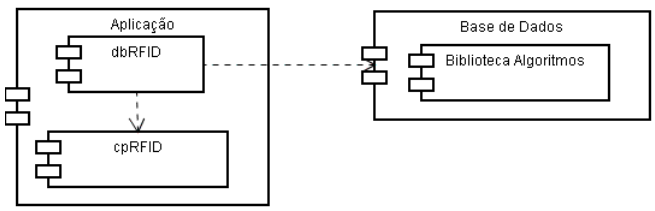


Figura 5: Diagrama de componentes.

O sistema foi dividido em quatro módulos: "simulação", "cadastro", "relatório de resultados" e "geração de pseudocódigo".

O "módulo de simulação" permite que o usuário selecione um algoritmo previamente cadastrado, o leitor e a etiqueta a serem utilizados na simulação e os valores para os seguintes parâmetros: número inicial e final de etiquetas, incremento e tamanho inicial do *frame* (no caso de algoritmos que utilizam esse parâmetro). Isso feito, o sistema compila o algoritmo selecionado e simula sua execução ao longo do número de ciclos fixado pelo usuário. Os resultados obtidos são armazenados na base de dados para futura verificação e comparação.

O "módulo de cadastro"permite a inclusão, alteração e exclusão de algoritmos, componentes, normas técnicas e dispositivos de hardware (leitores e etiquetas). O cadastro de algoritmos é realizado mediante a seleção de componentes, sendo cada um deles definido no sistema como um segmento de código (classe, função etc.), em linguagem C#, que pode ser reutilizado por um ou mais algoritmos. Cabe ressaltar que a codificação em

C# dos algoritmos anticolisão é, naturalmente, realizada de forma manual, a partir da análise de especificações disponíveis na literatura especializada; uma vez codificado, o referido algoritmo é incluído em uma base de dados – a biblioteca de componentes. Utilizando-se, portanto, o acervo dessa biblioteca, a montagem de um novo algoritmo anticolisão é facilmente executada por meio da seleção e encadeamento organizado dos componentes nela cadastrados bem como de sua eventual edição, de modo a que se comporte conforme o estabelecido nas especificações.

O "módulo de relatório de resultados" permite consulta aos resultados das simulações e comparações de desempenho (vide Fig. 5), o que pode ser feito para todos os ensaios realizados ou apenas para aqueles abrangendo um dado tipo de algoritmo, norma técnica, leitor ou etiqueta.

Relatório de Ensaios de Protocolo Anti-Colisão

Algoritmo: Arvore Binária **Data Ensaio:** 14/08/2010 14:28:17

Leitor: Motorola RD5000 - 40(kpbs) **Etiqueta:** FastTrack LRP820

ID (bits): 8

Nº de Tags	Tempo (s)	*Slots*	*Slots* Colisão	Taxa Eficiência	Taxa Colisão
10	0,006	28	13	35,71	46,43
20	0,009	46	22	43,48	47,83
30	0,013	64	31	46,88	48,44
40	0,017	84	41	47,62	48,81
50	0,021	106	52	47,17	49,06
60	0,024	122	60	49,18	49,18
70	0,029	146	72	47,95	49,32
80	0,032	162	80	49,38	49,38
90	0,037	184	91	48,91	49,46
100	0,041	204	101	49,02	49,51
			Média	**46,53**	**48,74**

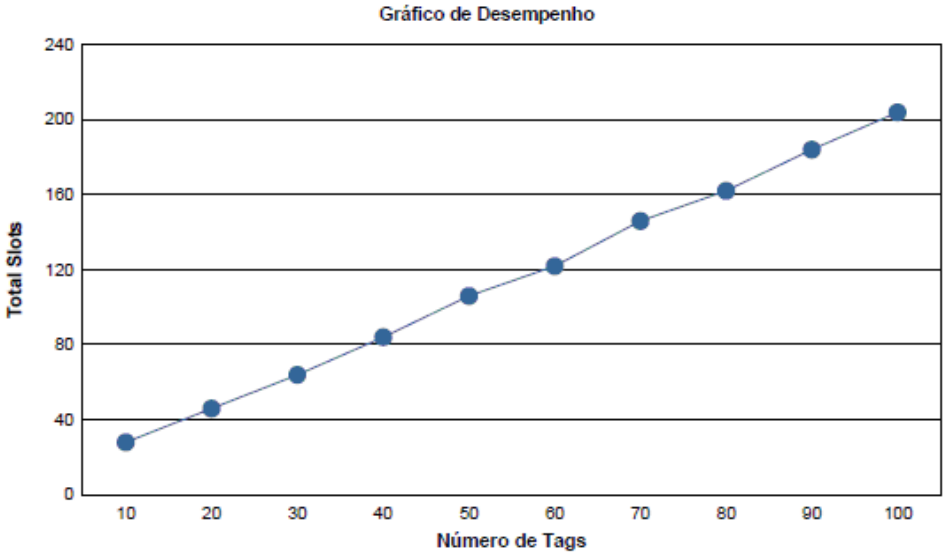


Figura 5: Resultados da simulação.

Com o propósito de facilitar a migração dos algoritmos configurados para os componentes reais de um sistema RFID, o sistema possui um "módulo de geração de pseudocódigo" que contém a descrição das funções de alto nível que devem ser implantadas nos componentes RFID específicos. Nos casos de algoritmos baseados em normas, essas funções vêm incorporadas aos componentes fabricados. Esse pseudocódigo (vide Fig. 6) serve como base para o projetista validar as funções necessárias para o correto funcionamento do algoritmo, não sendo necessário detalhar o código para a realização dos experimentos.

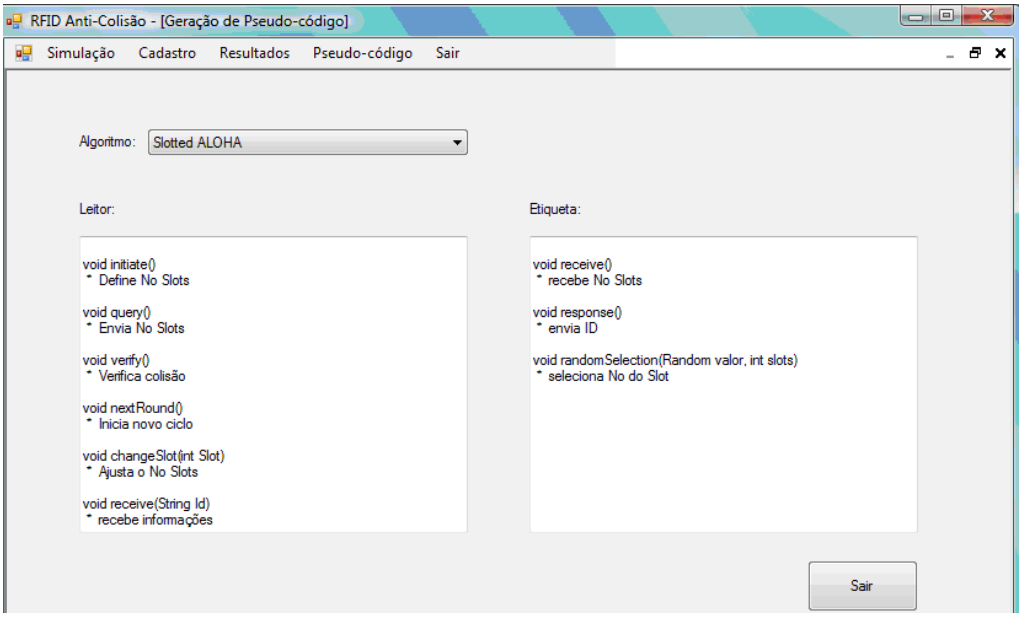


Figura 6: Pseudocódigo gerado.

4 Resultados experimentais

Após o cadastro de algoritmos anticolisão no ambiente de simulação, pode-se simular sua execução por meio de experimentos virtuais, e, assim, investigar o efeito de certos parâmetros característicos sobre seu desempenho. Para tanto, são adotadas duas métricas de desempenho: o número total de *slots* necessários à identificação das etiquetas e o *throughput* – razão entre o número de slots em que a transmissão foi bem sucedida (*nt*) e o número *n* total de *slots,* abrangendo o número de slots vazios (*nv*) e o de *slots* em que ocorreram colisões (*nc*).

Nos experimentos ilustrados neste artigo admitiu-se o uso de um leitor Motorola® RD5000 e de etiquetas RFID *FastTrack*® modelo LRP820. É importante ainda destacar que as estimativas de tempo de leitura referidas no texto basearam-se tão somente na velocidade nominal de transmissão do leitor (bits/s). Portanto, foram ignorados os efeitos de: 1) eventuais operações não diretamente relacionadas com o processo de identificação; 2) interferências causadas pelas condições ambientais; 3) disposição geométrica das etiquetas em relação ao leitor.

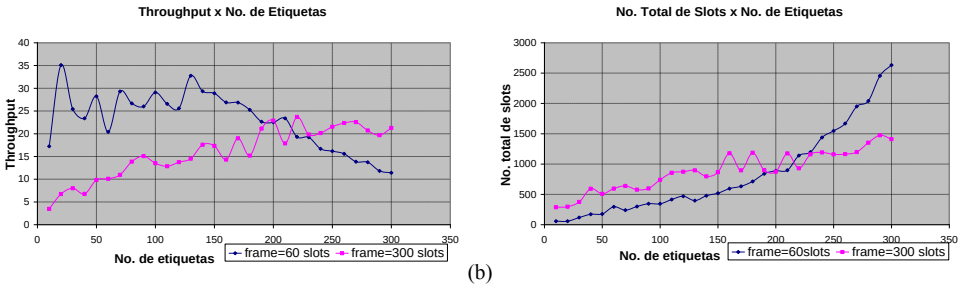
A título de ilustração dos recursos oferecidos pelo ambiente de simulação de algoritmos anticolisão, descrevem-se nos próximos tópicos experimentos realizados com alguns dos algoritmos anticolisão cadastrados no sistema.

4.1 Experimentos com o algoritmo Framed Slotted ALOHA

O principal desafio para a configuração desse algoritmo é a escolha adequada do número de *slots* disponíveis em cada *frame*. Utilizando-se o ambiente de simulação, foram realizados dois ensaios de identificação visando à comprovação da importância desse parâmetro no desempenho do algoritmo. Em ambos os ensaios, variou-se o número de etiquetas simultâneas presentes na área de varredura do leitor entre 10 e 300 exemplares, mas no primeiro ensaio estabeleceu-se em 60 o número f inicial de *slots* disponíveis em cada *frame*, enquanto que no segundo ampliou-se esse número para 300, igualando-o ao número máximo de etiquetas a serem identificadas.

Conforme se pode observar nos gráficos da Fig.7a, quando existem poucas etiquetas na área de leitura, o desempenho obtido no segundo ensaio (f=300) é inferior ao do primeiro ensaio (f=60), fato que se atribui ao elevado número de *slots* vazios que ocorrem para aquela configuração do algoritmo. À medida, porém, que o número de etiquetas se aproxima do valor máximo estabelecido, o desempenho do algoritmo para f=300 supera o da configuração do primeiro ensaio. É interessante também observar (Fig 7b) que, à medida que o número de

etiquetas se aproxima do valor máximo (300), a inclinação da curva *nº total de slots* × *nº de etiquetas* sofre um incremento considerável para f=60.



(a) (b)

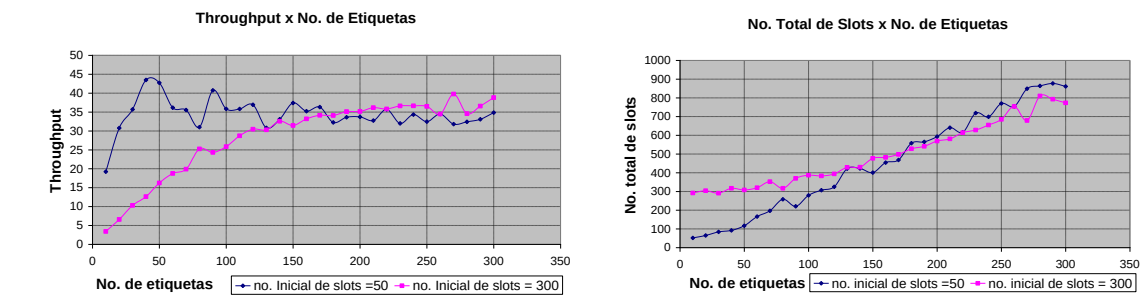
Figura 7: Algoritmo Frame-Slotted ALOHA: (a) *Throughput*; (b) Nº total de *slots*.

Os resultados fornecidos pelos dois experimentos apresentados acima corroboram com o disposto em "An improved anti-collision algorithm in RFID system" [16] acerca do algoritmo *Framed Slotted ALOHA*. Quando o número reservado de *slots* é maior que o número de etiquetas a serem lidas, o tempo de identificação é grande devido ao grande número de *slots* vazios; mas se esse número de *slots* é menor que o de etiquetas a serem lidas, o tempo aumenta devido ao grande número de colisões. Portanto, a adequada definição do número de *slots* por *frame* é fundamental para o bom desempenho desse algoritmo.

4.2 Experimentos com o algoritmo *Dynamic Framed Slotted ALOHA*

Nesses experimentos, o número ideal de *slots* em um *frame* foi calculado a cada instante de acordo com o algoritmo apresentado em "An improved anti-collision algorithm in RFID system" [16]. Dois ensaios foram realizados admitindo-se os valores 50 e 300 para o número inicial de *slots* utilizado no referido algoritmo.

Os gráficos da Fig. 8a mostram que, apesar de o algoritmo possuir a capacidade de calcular o número *f* de *slots* por *frame* de forma adaptativa (vide Fig. 8b), o máximo *throughput* é atingido quando o número de etiquetas presentes na área de interrogação é próximo ao do valor inicial admitido para *f* .



(a) (b)

Figura 8: Algoritmo Dynamic Slotted ALOHA: (a) *Throughput* ; (b) Nº total de *slots*.

A primeira requisição do leitor tem, portanto, grande importância para se obter o máximo *throughput*. Logo, se, para uma dada aplicação, o número de etiquetas que estão na área de interrogação do leitor é conhecido *a priori* (é o caso, por exemplo, de um caminhão que transporta sempre o mesmo número aproximado de itens), é possível configurar o algoritmo para que opere com máximo *throughput*. Isso pode ser reproduzido no ambiente de simulação desenvolvido, conforme bem o ilustra o gráfico da Fig. 9.

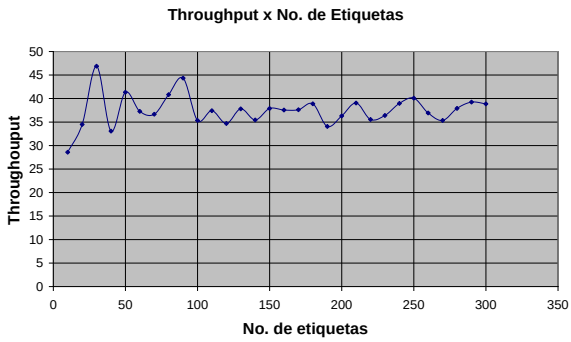
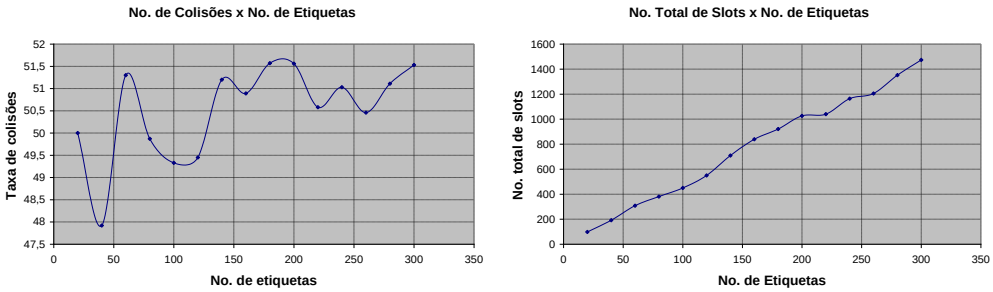


Figura 9: Dynamic Frame Slotted ALOHA otimizado.

4.3 Experimentos com o algoritmo *Btree*

Btree disponibiliza sempre um único *slot* para a realização da transmissão, o *slot* de número zero. As etiquetas iniciam o processo de identificação com o valor zero atribuído ao seu contador, portanto, no primeiro ciclo, todas as etiquetas realizam a transmissão simultaneamente.

Conforme ilustrado nas Figs.10a-b, o número total de *slots* cresce de forma aproximadamente linear com o número de etiquetas e a taxa de colisões (razão entre o número de *slots* em que ocorre colisão e o número total de *slots*) do algoritmo mantém-se em torno de 50%.



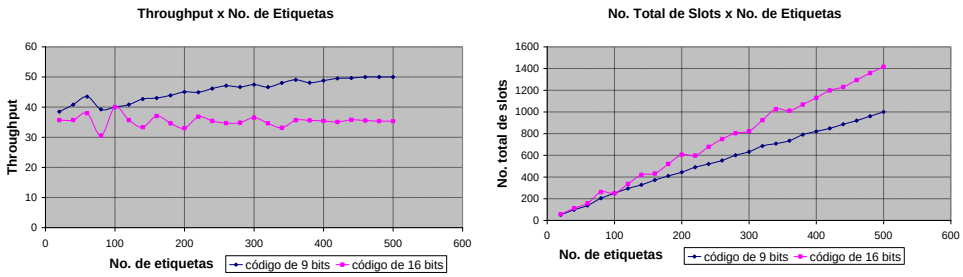
(a) (b)

Figura 10: Algoritmo Btree: (a) Taxa de colisões. (b) Nº total de *slots*.

4.4 Experimentos com o algoritmo Árvore Binária

De acordo com Sommerville [17], esse algoritmo apresenta queda de desempenho quando o número de requisições e respostas aumenta, a qual é atribuída à extensão do código de identificação da etiqueta e à característica de busca em árvore. Observe-se que o leitor necessita construir a árvore binária de acordo com a extensão do código de identificação para executar a busca em profundidade e enviar as requisições, portanto, a árvore binária deve ter altura igual a essa extensão.

Utilizando-se o ambiente de simulação, foram realizados dois ensaios que demonstram o impacto da extensão do código no processo de identificação das etiquetas. No primeiro ensaio, utilizou-se um código composto por 9 bits; no segundo, um código de 16 bits, conforme adotado no processo de identificação do padrão *Electronic Product Code* (EPS). Em ambos os ensaios, os códigos foram gerados a partir de chamadas a uma função geradora de números aleatórios. Conforme se pode observar nos gráficos da Fig. 11a-b, nos dois ensaios as curvas de variação do número total de *slots* em função do número de etiquetas são aproximadamente lineares; nota-se, porém, que o uso de código de 16 bits redunda em *throughput* médio inferior ao que se obtém com um código de 9 bits.



(a) (b)

Figura 11: Algoritmo Árvore Binária: (a) *Throughput*; (b) Nº total de *slots*.

Sempre é possível, no entanto, construir um ambiente computacional em que o leitor tenha acesso à base de dados da aplicação RFID e que essa base contenha as informações de todas as etiquetas cadastradas. Dessa forma, a travessia da árvore binária pode ser aprimorada, não havendo a necessidade de se construir todas as ramificações inerentes ao número de bits do código identificador [13], mas apenas aquelas que constam da base de dados do leitor, o que melhora bastante o desempenho do sistema, principalmente em aplicações com um número limitado de etiquetas.

O comportamento dessa versão do algoritmo árvore binária – denominada 'árvore binária com *cut through'* é ilustrado na Fig. 12, onde se pode observar um pequeno incremento no *throughput* do sistema relativamente ao do algoritmo tradicional.

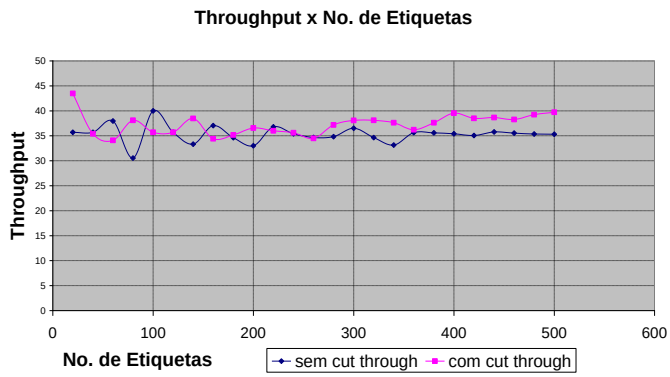


Figura 12: Algoritmo Árvore Binária com e sem "cut through" .

4.6 Exemplo de aplicação do ambiente de simulação de algoritmos anticolisão

De posse da ferramenta de simulação dos algoritmos anticolisão, é possível demonstrar a sua aplicabilidade à solução de um problema prático do âmbito de um projeto de sistemas distribuídos baseados em RFID.

Tomando-se como exemplo o problema de manutenção de equipamentos da rede de distribuição elétrica, seria desejável que os espécimes danificados e identificados por etiquetas RFID, no momento em que cruzassem o portal de entrada da empresa de manutenção, embarcados em caminhões transportadores, fossem identificados por uma antena, de modo a facilitar o subsequente processo de expedição. A título de ilustração, admitem-se as seguintes pré-condições: 1) o caminhão transporta, em média, 70 equipamentos diversos, como, por exemplo, chaves de religação e transformadores, todos eles identificados por uma etiqueta RFID proprietária de 16 bits; 2) a velocidade média do caminhão é de 20Km/h; 3) o raio de ação do leitor é de, aproximadamente, 3m.

Assim, são realizados ensaios de acordo com as especificações, utilizando-se os algoritmos *Btree*, Árvore Binária com *Cut-Through e Dynamic Slotted ALOHA*.

Pode-se estimar o tempo de exposição das etiquetas na área de abrangência do leitor de acordo com a velocidade média do caminhão – cerca de 0,54 segundos. O gráfico comparativo da Fig. 13 exibe os tempos aproximados de identificação total das etiquetas. Para tanto somente são consideradas as operações relacionadas à identificação das etiquetas, desprezando-se as relacionadas à segurança ou à leitura ou gravação de informações adicionais. Com base no gráfico citado, é possível deduzir que o algoritmo Btree não consegue identificar todas as etiquetas quando o caminhão transportar mais do que 90 equipamentos; todos os demais algoritmos atendem ao requisito de tempo de leitura.

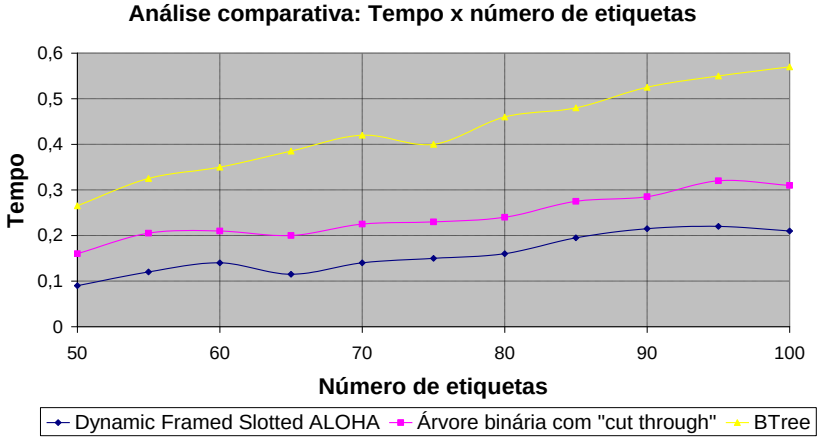


Figura 13: Análise comparativa de algoritmos: Tempo × nº de etiquetas.

Conforme se pode observar no gráfico da Fig. 14, o algoritmo *Dynamic Slotted ALOHA* é o que necessita de um menor número de requisições; já no gráfico da figura 15, pode-se constatar a sua superioridade no que tange ao *throughput*. Portanto, conclui-se que, para a aplicação ilustrada no exemplo, o algoritmo *Dynamic Slotted ALOHA* seria o mais recomendável.

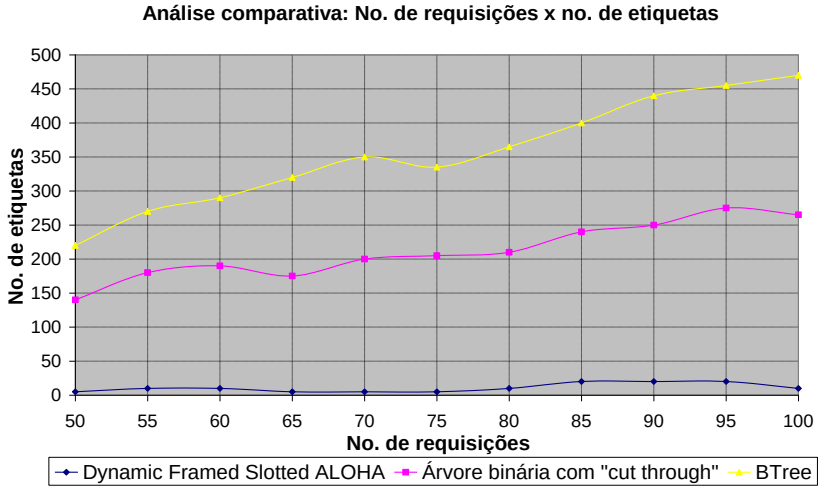


Figura 14: Análise comparativa de algoritmos: total de requisições × nº de etiquetas.

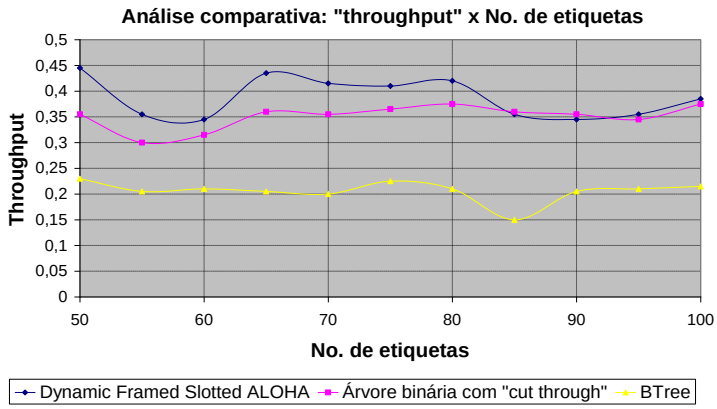


Figura 15: Exemplo de aplicação prática: *Throughput* × nº de etiquetas.

5 Conclusões

Neste artigo apresentou-se um ambiente de simulação de algoritmos anticolisão que permite ao projetista de sistemas RFID realizar a seleção e ajuste de parâmetros dos algoritmos mais adequados à sua aplicação. O sistema desenvolvido permite a inclusão ou alteração de algoritmos anticolisão, de acordo com a necessidade do usuário e em harmonia com as boas regras de reutilização de código, o que facilita a implementação de uma nova versão de um algoritmo já cadastrado.

Para ilustrar os recursos de uso oferecidos pela ferramenta, foram analisados alguns importantes algoritmos anticolisão citados na literatura mediante a realização de ensaios de simulação envolvendo a sua execução sob variadas condições de operação; e os resultados desses experimentos corroboraram com os publicados na literatura. Ao final pôde-se demonstrar a utilidade do sistema desenvolvido mediante a sua aplicação à seleção de um algoritmo anticolisão para um sistema RFID voltado à identificação de equipamentos transportados em caminhões trafegando à baixa velocidade.

Referências

[1] CHENG, T.; JIN, L. Analysis and simulation of RFID anti-collision algorithms. In: 9th INTERNATIONAL CONFERENCE ON ADVANCED COMMUNICATION TECHNOLOGY, Phoenix Park, Korea, February 12-14, 2007. *Proceedings* of the 9th International Conference on Advanced Communication Technology, p. 697-701, 2007. Phoenix Park, Korea: 2007. 5p.

[2] BANKS, J. *RFID Applied*. John Willey and Sons, 2007, 509p.

[3] SHIH, D.H.; SUN, P.L.; YEN, D.; HUANG, S.M. Taxonomy and Survey of RFID anti-collision protocols. *Computer Communications*, v. 29, no.11, 2006, p. 2150-2166.

[4] SHU, L.; LIAO, G. Event-driven RFID system development for retailer supermarket. In: THE 1ST INTERNATIONAL CONFERENCE ON INFORMATION SCIENCE AND ENGINEERING, Nanging, China, December 26-28, 2009. *Proceedings* of the 1st International Conference on Information Science and Engineering (ICISE2009). Nanging: China: 2009. 4p.

[5] IPT. *Controle logístico inteligente de equipamentos de distribuição em estoque e instalados em campo: especificações iniciais do software e do middleware*. Instituto de Pesquisas Tecnológicas do Estado de São Paulo: Relatório técnico no. 94981-205. 2007.

[6] LIU, L.; LAI, S. ALOHA-based anti-collision algorithms used in RFID system. In: INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS, NETWORKING AND MOBILE COMPUTING, Wuhan, China, September 22-24, 2006. *Proceedings* of the International Conference on Wireless Communications, Networking and Mobile Computing. Wuhan, China: 2006. 4p.

[7] GLOVER, B.; BHATT, H. *Fundamentos de RFID*. Rio de Janeiro: Alta Books, 2006. 240p.

[8] SHI, X.L.; SHI, X.W.; HUANG, Q.L.; WEI, F. An enhanced binary anti-collision algorithm of backtracking in RFID system. *Progress in Electromagnetics Research B*, v.4, 2008, p. 263-271.

[9] PENG, Q.; ZHANG, M.; WU, W. Variant enhanced dynamic frame slotted ALOHA algorithm for fast object identification in RFID system. In: 2007 INTERNATIONAL WORKSHOP ON ANTI-COUNTERFEITING, SECURITY AND IDENTIFICATION, Xiamen, China, April, 16-17, 2007. *Proceedings* of the IEEE International Workshop on Anti-Counterfeiting, Security and Identification. Xiamen, China: 2007. 4p.

[10] LEE, S.R.; JOO, S.D.; LEE, C.W. An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification. In: THE 2nd ANNUAL INTERNATIONAL CONFERENCE ON MOBILE AND UBIQUITOUS SYSTEMS: NETWORKING AND SERVICES, San Diego, California, July 17-21, 2005. *Proceedings* of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services. San Diego, California: 2005. 7p.

[11] TONG, Q.; ZOU, X.; TONH, H. Dynamic framed slotted ALOHA algorithm based on Bayesian estimation in RFID system. In: 2009 WRI WORLD CONGRESS ON COMPUTER SCIENCE AND IFORMATION ENGINEERING, Los Angeles, California, March 31 – April 2, 2009. *Proceedings* of the WRI World Congress on Computer Science and Information Engineering. Los Angeles, California: 2009. p.384-388.

[12] ISO-IEC. *Information technology – Radio frequency identification for item management – Part 6:* Parameters for air interface communications at 860 MHz to 960 Mhz. In: ISO/IEC 18000-6, 2004.

[13] WANG, T.P. Enhanced binary search with cut-through operation for anti-collision in RFID systems. *IEEE Communications Letters*, v.10, n.4, 2006, p.236-238.

[14] SOMMERVILLE, I. *Engenharia de Software*. Pearson Education, 8ª edição. 2007. 568p.

[15] NIST. *Integration Definition for Information Modelling*. NIST Federal Information Processing Standards Publication 184, 1993, 200p. Disponível em: <www.itl.nist.gov/fipspubs/idef1x.doc>.

[16] LIU, L.; XIE, Z.; XI, J.; LAI, S. An improved anti-collision algorithm in RFID system. In: 2nd INTERNATIONAL CONFERENCE ON MOBILE TECHNOLOGY, APPLICATIONS AND SYSTEMS, Gouangzhou, China, November 15-17, 2005. *Proceedings* of the 2nd International Conference on Mobile Technology, Applications and Systems. Gouangzhou, China: 2005. 5p.

[17] QUAN C.H.; HONG W.K.; KIM H.C. Performance analysis of tag anti-collision algorithms for RFID systems. In: *Lecture Notes in Computer Science*, v.4097, 2006, p.382-391.

[18] AZAMBUJA, M., JUNG, C., CATEN, C., HESSEL, F., RFID-Env: methods and software simulation for RFID environments. *Business Process Management Journal*, v.16, Issue 6, p.1014-1038.