FBL - Filtro de Bloom Linear

Rui Lima[1] , Carlos Baquero[2] * e Hugo Miranda[3]

[1] CLEGI, Universidade Lusíada - Norte, V. N. Famalicão
rml[at]fam.ulusiada.pt
[2] HASLab / INESC tec, Universidade do Minho, Braga
cbm[at]di.uminho.pt
[3] LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
hmiranda[at]di.fc.ul.pt

Resumo As estruturas de dados que permitem o armazenamento de informação de forma probabilística (em particular, os filtros de Bloom) caracterizam-se por permitir a regulação do equilíbrio entre a eficiência na gestão do espaço de armazenamento e a precisão das respostas. Esta possibilidade tem motivado a sua utilização em cenários adversos, por exemplo em redes de sensores, onde os dispositivos apresentam recursos limitados (memória, cpu, energia). Este artigo apresenta um mecanismo de Filtro de Bloom Linear (FBL), que permite associar a cada um dos elementos uma probabilidade quantizada no intervalo real [0,1], ultrapassando as limitações impostas pela característica binária dos filtros de Bloom. Os resultados mostram que é possível parametrizar um FBL de modo a manter um erro aceitável em função da variação do número de bits usados na quantização e do número de funções de *hash* usadas na indexação. O artigo discute a aplicação dos FBLs em mecanismos de disseminação e descoberta de recursos em redes de sensores, mostrando como contribuem para manter uma dimensão constante das mensagens trocadas pelos sensores, independentemente da dimensão da rede.

Keywords: Filtros de Bloom, Difusão, Redes de Sensores

1 Introdução

Um Filtro de Bloom (FB) [2] é uma estrutura de dados que permite o armazenamento eficiente de informação de uma forma probabilística. Estas estruturas de dados têm muitas aplicações [16] como por exemplo em dispositivos com baixos recursos de memória [12]. Uma das suas características fundamentais é a elevada eficiência na gestão do espaço de armazenamento de dados. Contudo, a sua natureza não determinística tem sempre associado um grau de incerteza, permitindo que na consulta da informação inserida (interrogação ao filtro) possam surgir erros na resposta, erros estes que se manifestam como falsos positivos. Os FBs

* O trabalho descrito neste artigo foi parcialmente suportado pela FCT - Fundação para a Ciência e Tecnologia, Portugal e desenvolvido no âmbito do projeto UID/EEA/50014/2013 (HASLab - INESC TEC).

permitem ajustar alguns dos seus parâmetros assegurando um limite máximo para o erro esperado, de forma que cada aplicação possa estabelecer um nível de erro que seja comportável.

A motivação imediata para a proposta de um Filtro de Bloom Linear (FBL), resulta da necessidade de gestão eficiente de informação difundida e armazenada para a pesquisa de recursos em redes WSN (Wireless Sensor Network). Atendendo que os sensores podem encaminhar as mensagens uns dos outros (*multi-hop*), os FB podem ser incluídos nas mensagens trocadas entre os sensores, uma aproximação já utilizada em [7,4] para identificar quais os recursos detidos por cada nó da rede. No entanto, a informação mantida pelos FB tradicionais é limitada a uma indicação binária da presença/ausência de cada recurso (elemento do conjunto).

O FBL estende o FB tradicional ao permitir associar um parâmetro adicional a cada elemento que é inserido no filtro. O que permite ao FBL responder a interrogações sobre a presença de um elemento e ainda obter uma indicação sobre o nível de confiança dessa resposta. Uma característica interessante do parâmetro de confiança é a possibilidade de ajustar o seu valor em função dos diferentes nós em que é registado. No caso das redes multi-hop, este valor pode ser usado para inferir sobre a distância a que se encontra um determinado recurso, um aspeto muito relevante para redes sem infraestrutura como as redes de sensores.

O artigo discute o equilíbrio entre a resolução dos parâmetros de configuração e o número de elementos do FBL, demonstrando que a calibração do FBL permite obter um erro aceitável em função da variação do número de bits usados na quantificação e do número de funções de *hash* usadas na indexação. O FBL utiliza uma técnica bastante geral, potencialmente aplicável a outros contextos.

2 Trabalho Relacionado

Os Filtros de Bloom (FB) [2] e as suas variantes têm sido muito usados em vários domínios [16]. Um FB permite representar sucintamente um conjunto, à custa de uma dada probabilidade de falsos positivos ao testar a pertença de elementos nesse conjunto. A versão mais simples de um FB pode ser implementado recorrendo a um simples vetor de elementos binários. Ao inserir um dado elemento, são calculadas k coordenadas no vetor pela aplicação de igual número de funções de *hash*. Os bits nestas coordenadas são passados a 1 para sinalizar a inserção. Dualmente, a interrogação da pertença verifica se todas as coordenadas do elemento a testar estão a 1. A calibração do k ótimo depende no tamanho do vetor e do número de elementos a inserir. Os falsos positivos resultam da possibilidade de um elemento não inserido aparentar estar presente, se as suas coordenadas coincidirem todas com outros elementos inseridos no filtro.

Sendo os FB representações de conjuntos, a necessidade de noutros contextos de aplicação representar multi-conjuntos permitindo remoções, conduziu a uma variante designada por *Counting Bloom Filters* (CBF) [6]. No CBF o vetor deixa de ter valores com um único bit passando a ter um pequeno contador, permitindo operações de eliminação ao incorporar a possibilidade de decrementar o

respetivo contador. O processamento de grandes quantidades de dados necessita de otimizações das operações principais, reduzindo o tempo de execução para a inserção e remoção de elementos [9]. Uma outra forma de otimização consiste em melhorar as funções de *hash*, permitindo aumentar a eficiência de execução e reduzir o espaço de armazenamento [3]. A variante Bloomier [4] permite codificar funções estendendo as tradicionais interrogações para testar se um elemento é membro de um conjunto, ao permitir associar atributos a cada um dos elementos disponíveis. A sua estrutura resulta numa utilização em paralelo de múltiplos FB. Ao ser adicionado a mensagens de pesquisa em redes sem fios, permitiu desenvolver aplicações de aquisição de contexto e descoberta de recursos [16].

Para evitar filtros sobredimensionados surgiu a variante *Scalable Bloom Filters* (SBF) [1] que permite adaptar dinamicamente a qualidade do filtro, sem saber previamente o número de elementos a inserir. Desta forma foi possível ultrapassar uma das principais limitações dos FBs, que consistia na necessidade de definir o tamanho do filtro em função do número de elementos a inserir e da qualidade do filtro. A utilização de FBs em aplicações distribuídas despoletou uma técnica que permite implementar FB de crescimento dinâmico [8], em que a dimensão do filtro é ajustada (durante a execução) em função do número de elementos inseridos.

Ao transmitir um FB em mensagens através da rede é necessário considerar aspetos de segurança, tomando ações simples como por exemplo aumentar a probabilidade de esquecer elementos antigos face a elementos mais recentes, quando existe sobreposição de elementos [11] no filtro. Numa WSN os sensores podem adquirir informação de contexto através da escuta das mensagens dos seus vizinhos. Os FB podem resumir essa informação de contexto para melhorar os protocolos de encaminhamento [10] em redes 6LoWPAN (*IPv6 over Low-Power Wireless Personal Area Networks*). Em vez de processar e armazenar toda a informação das tabelas de encaminhamento sobre redes IP, é possível obter bons resultados apenas com resumos destas tabelas armazenados em FB, recolhendo essa informação recorrendo à troca de mensagens [17]. Começam a surgir propostas alternativas [14] ao FB tradicional, como por exemplo o filtro Cuckoo [5] que permite a remoção de elementos dinamicamente, melhorando a performance do filtro em espaço de armazenamento e tempo de consulta.

A variante *attenuated Bloom filter* [15], foi aplicada num cenário de pesquisa de documentos sobre redes *peer-to-peer*. Este tipo de filtro é formado por uma matriz de FBs, em que na posição i da matriz está armazenado o FB resultante da contribuição de todos os servidores que se encontram à distância i, de modo a estimar a distância em *hops* entre servidores com documentos relacionados. Este mecanismo permite adquirir contexto dos servidores vizinhos de modo a obter um algoritmo de localização probabilístico. Esta abordagem revela um potencial elevado de aplicação em WSNs, mas o facto da dimensão desta estrutura de dados depender do diâmetro da rede, é uma condição demasiado exigente para sensores com recursos (memoria, cpu, energia) limitados. Existe ainda um mecanismo de encaminhamento que propõem o decaimento da informação contida num FB, para encontrar um gradiente que corresponda à rota entre dispositivos numa

rede WSNs [7]. Contudo o mecanismo de difusão é demasiado pesado e não prevê a sua aplicação de forma eficiente em redes com topologia dinâmica.

3 Filtro de Bloom Linear

Face a estas limitações, este artigo vem propor um Filtro de Bloom Linear (FBL), que inspirado nas características das variantes anteriores, permite atender às limitações impostas pelos sensores mantendo uma dimensão constante da estrutura de dados.

3.1 Filtro de Bloom

A estrutura de dados mais simples para um FB é a sua implementação como um vetor binário, onde cada posição pode armazenar um único bit $\{0, 1\}$. No FB a informação é armazenada com uma técnica probabilística, para permitir operar sobre grandes quantidades de dados, fazendo uma gestão eficiente do espaço de memória disponível. A dimensão do filtro é representada por m e corresponde ao número de posições disponíveis no vetor binário. O número de elementos inseridos no FB é representado por n.

Sobre um FB podem-se realizar duas operações principais: **inserir** (armazenar elementos no filtro) e **interrogar** (questionar o filtro para saber se este contém um determinado elemento). A operação de inserção de elementos corresponde a atribuir a cada uma das k posições do vetor (obtidas usando k funções de *hash* independentes) o valor 1. A operação de interrogação começa por aplicar as k funções de *hash* aos elementos, retornando uma resposta afirmativa quando verifica que todas essas k posições do vetor tem o seu valor definido a 1. A natureza probabilística do FB permite falsos positivos, ou seja o resultado da interrogação pode devolver uma indicação errada sobre a pertença de um elemento, embora esse elemento nunca tenha sido inserido. A probabilidade de falsos positivos é dada por p^k, sendo p a probabilidade de um determinado bit não ser 1. A análise combinatória resulta que após a inserção de n elementos o valor de p pode ser calculado pela Eq.(1).

$$p = \left(1 - (1 - 1/m)^{nk}\right) \approx \left(1 - e^{-nk/m}\right) \tag{1}$$

É possível calcular analiticamente o valor de k que minimiza a probabilidade de falsos positivos usando a Eq.(2) (mas na prática usa-se sempre k inteiro). A probabilidade de falsos positivos f é dada pela Eq.(3).

$$k = \frac{m}{n} \ln 2 \tag{2}$$

$$f \approx (1/2)^k \approx (0.6185)^{m/n} \tag{3}$$

As equações (2) e (3) permitem escolher k em função de f, contudo é possível enunciar também uma **regra prática** para o dimensionamento de um filtro: Caso

se conheça a ordem de grandeza dos elementos a inserir no filtro, o FB deve ter uma dimensão $m \approx 10 \times n$ e para assegurar uma probabilidade de falsos positivos que seja inferior a 1%, então resulta $k = 7$.

Inicialmente os filtros arrancam com todas as posições a zero. A Figura 1 representa a fase de inserção dos elementos $\{a, e\}$, sobre um FB previamente carregado com os elementos $\{b, c, d\}$, usando $k = 4$ funções de hash_k independentes para gerar os índices do vetor.

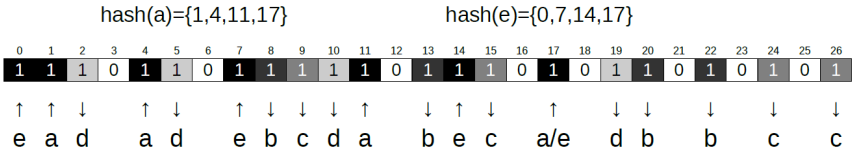


Figura 1: FB com os elementos {a,b,c,d,e}

Ao inserir $\{a\}$ as posições $\{1, 4, 11, 17\}$ são colocadas a 1, mas ao inserir o elemento $\{e\}$ nas posições $\{0, 7, 14, 17\}$ surge uma sobreposição, que por si só não perturba o resultado da interrogação. Contudo, dada a densidade do filtro apresentado na Figura 1, caso sejam inseridos mais elementos o filtro tende a saturar. Neste caso, a taxa de falsos positivos aumenta até ao ponto em que o filtro fica completamente saturado, ficando o vetor correspondente com todas as posições definidas a 1 e o filtro deixa de ter utilidade. Estudos anteriores [15,3,4] exploraram este efeito, de modo a ter uma regulação do equilíbrio entre a eficiência na gestão do espaço de armazenamentos e a precisão das respostas, concluindo que o número de bits a 1 deve ser aproximadamente metade da dimensão do filtro.

3.2 FBL: Estrutura e Operadores

O Filtro de Bloom Linear (FBL) é uma estrutura semelhante a um Filtro de Bloom que permite armazenar junto a cada elemento uma grandeza totalmente ordenada (dando um resultado único perante as funções binárias *max* e *min*). Assim, já não se trata de representar um conjunto, mas sim uma associação que faz o mapeamento entre elementos e grandezas numéricas. Este artigo considera uma representação das grandezas numéricas em números reais limitados ao intervalo $[0, 1]$. A cada elemento fica associado um parâmetro que indica o grau de confiança c com que o elemento é adicionado ao conjunto. Por omissão o FBL é inicializado com o valor 0 em todas as posições do vetor (tal como usual, 0 indica a não pertença). Caso seja usado $c = 1$ para todos os elementos inseridos, então o FBL comporta-se como um FB tradicional. Nos restantes casos é possível considerar um grau de incerteza considerando valores intermédios para o parâmetro de confiança $0 < c < 1$. Para associar um grau de confiança a cada elemento inserido no FBL, é formado o par $(elem_n, c_n)$. A Figura 2 ilustra uma inserção análoga à apresentada na Figura 1 mas realizada sobre um FBL.

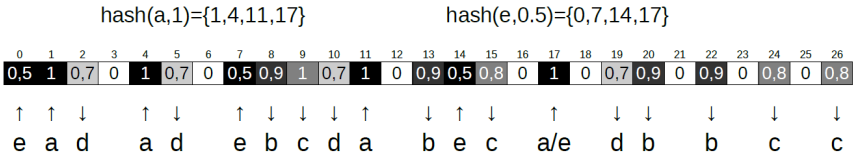


Figura 2: FBL com os elementos {a,b,c,d,e} e respetivo parâmetro de confiança

A operação de **interrogação** ao FBL é realizada pela função *conFBL*(*elem*) apresentada no Alg. 1. Caso a consulta retorne $c = 0$, então *elem* não faz parte do conjunto armazenado no FBL. Caso a consulta retorne $c \in]0, 1]$, então o filtro funciona como um estimador, retornando o grau de confiança estimado $\hat{c}$ para o elemento *elem*.

A operação *min* da função *conFBL*(*elem*) assegura que em casos de sobreposição parcial, como acontece no FBL apresentado na Figura 2 na posição 17, a consulta do parâmetro c não é afetada. Contudo, o resultado da interrogação pode sofrer uma **sobreestimação de pertença** resultante de um erro de estimação por excesso, caso a sobreposição seja completa (sobreposição em todas funções de *hash*). A sobreestimação de pertença é uma generalização do conceito de falso positivo no contexto dos FBs, que se deteta quando o estimador devolve um grau de confiança c superior ao inserido para um dado elemento, ou seja $\hat{c} > c$. Este comportamento pode resultar de um mau dimensionamento do filtro, podendo ser um indicador da entrada em saturação do FBL.

Algorithm 1: Operações sobre FBL

```
1  const FBL[] ;                                   // Filtro de Bloom Linear
2  const K ;                        // Número de funções de hash independentes
3  Function conFBL (elem)
4      c ← 1;
5      for i ← 1 to K do
6          idx ← hash_i(elem);
7          c ← min (c,FBL[idx]);
8      end
9      return c;
10 Function insFBL (elem,c)
11     for i ← 1 to K do
12         idx ← hash_i(elem);
13         FBL[idx] ← max (c,FBL[idx]);
14     end
```

A operação de **inserção** é semelhante à realizado num FB tradicional, mas em vez se armazenar um único bit para cada uma das k posições do vetor, o valor c é adicionado por uma operação de majoração (*max*) em cada uma das k

células do FBL. A processo de inserção é realizada pela função $insFBL(elem, c)$ apresentada no Alg. 1. A Figura 2 apresenta um exemplo de um FBL sobre o qual estão a ser inseridos os elementos $\{(a, 1); (e, 0.5)\}$, através das operações $insFBL(a, 1)$ e $insFBL(e, 0.5)$ respetivamente.

Exemplo de aplicação: Uma operação de **atenuação**, que pode ser invocada por sensores de uma rede WSN, pode multiplicar todas as células do filtro por um dado fator (e.g. 0.9). O efeito é reduzir o grau de confiança associado a todos os elementos presentes no filtro, que no limite se aproxima de 0. O uso desta operação sobre a receção de FBL difundidos pelos sensores vizinhos, permite criar localmente gradientes sobre a localização de recursos em redes multihop [13].

3.3 FBL: Quantização Linear

Havendo um número infinito de reais em qualquer segmento de reais, em termos de implementação num FBL, seria necessário armazenar um número infinito de bits para fazer a representação binária de c. Existem várias formas para representar um número de vírgula flutuante na sua forma binária, como por exemplo a norma IEEE 754. Esta norma define os formatos adequados para representar números de vírgula flutuante com vários níveis de precisão, sendo os mais comuns: i) precisão simples (32 bits) e ii) precisão dupla (64 bits). O FBL não impõe nenhuma restrição sobre a técnica a ser utilizada para fazer a representação binária e respetiva compressão de um número de vírgula flutuante. Contudo, sugere-se a utilização de um mecanismo de quantização simples que permita a atribuição de valores discretos para uma grandeza contínua. Atendendo que no FBL o parâmetro a representar é $c \in [0, 1]$, são assumidas algumas simplificações para fazer a respetiva representação binária. Como c não tem sinal nem expoente, é suficiente fazer a representação binária da sua mantissa. A norma IEEE 754 de precisão simples aloca 23 bits para a representação da mantissa. Atendendo que o FBL poderá ser utilizado em redes de sensores, não faz sentido manter uma mantissa com esta precisão porque: i)o FBL é uma estrutura de dados probabilística (com incerteza); ii) os dispositivos sensores tem significativas restrições de memória e CPU.

O número de bits b usados no processo de quantização será um dos principais parâmetros de calibração do FBL. O espaço de memória alocado por cada FBL, i.e. a dimensão de FBL é $m \times b$ bits, sendo m número máximo de células disponíveis no FBL e b o número de bits usados em cada uma das células. Estes parâmetros devem ser ajustados em função da precisão aceitável pela aplicação final, sabendo que à medida que b aumenta o erro de quantização diminui, mas em contrapartida, aumenta linearmente a dimensão do FBL.

Sabendo que o maior número inteiro positivo representado por um conjunto b bits é (2^b-1), então é possível quantificar linearmente c dividindo o intervalo $[0, 1]$ em partes iguais, ou seja: $c \times (2^b-1)$. Contudo, o resultado desta operação poderá ser ainda um número real, que por uma operação de arredondamento pode ser convertido num inteiro, assumindo um erro máximo absoluto de quantização (FBL_{ERR}) dado pela Eq.(4).

$$FBL_{ERR} = \pm 1/(2^b - 1) \tag{4}$$

O arredondamento poderá ser parametrizado em função das necessidades das aplicações, mas considerando que tipicamente aplicações sobre WSNs associam $c = 1$ ao facto do recurso ser local ao sensor e que isso deve corresponder a ter todos os bits das respetivas células a "1", então é preferível fazer uma operação de truncatura implementada pela função $floor()$, para extrair apenas a parte inteira. Desta forma assegura-se que se existir uma atenuação, por mais pequena que seja, a representação binária nunca terá todos os bits a "1". A representação binária do parâmetro de confiança c pode ser obtida pela chamada da função $c2bit(c, b)$ apresentada no Alg.(2).

Algorithm 2: Representação Binária de c

```
1 Function c2bit (c,b)
2 |  return floor( c × ((1 ≪ b) − 1) ;        // floor(x) : ⌊x⌋
```

Exemplo: Se escolher uma precisão de $b = 5$ bits, então fica definido o erro absoluto $FBL_{ERR} = \pm 0,032 \approx \pm 3\%$.

A Figura 3 ilustra a representação correspondente às células [11-14] da Figura 2, onde é possível observar como o parâmetro confiança c é armazenado no filtro FBL, de modo que este seja formado por um vetor de bits, com uma dimensão controlada.

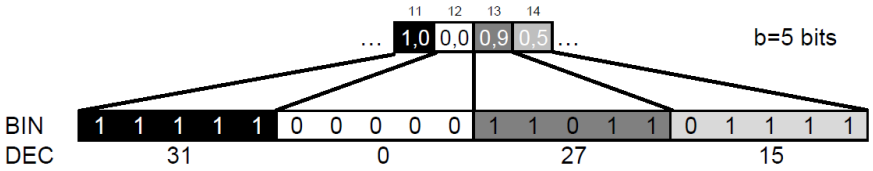


Figura 3: FBL - Representação Final em Memória

O FBL pode ser representado por um vetor de células, contendo em cada célula a representação binária do parâmetro de confiança c, como ilustrado na Figura 3.

Para interpretar a informação armazenada no FBL é necessário recorrer a operações inversas às anteriormente descritas. Por exemplo, para que um sensor possa realizar uma operação de atenuação, será necessário interrogar o FBL para obter a representação binária do valor c, posteriormente aplica-se uma operação inversa da quantização para obter a representação do valor estimado na forma de um número real, e por último aplicar a operação de atenuação.

4 Avaliação

Atendendo ao facto do FBL ser uma estrutura de dados probabilística, a interrogação ao FBL pode devolver uma informação distinta da inserida. Esta avaliação permite caracterizar como se comporta o FBL na sua função de **estimador** $\hat{c}$ correspondente ao mapeamento dos seus elementos noutra grandeza numérica. A avaliação mostra como se comporta o FBL em função da calibração dos principais parâmetros de configuração. No processo de avaliação são mantidas duas estruturas de dados para armazenar o mapeamento $(elem_n, c_n)$. Uma determinística que utiliza um vetor (*vetorElem[]*) que associa c a cada um dos $elem$ e outra implementada por um FBL que permite estimar o valor de c, ou seja $\hat{c}$. O comportamento médio do FBL é avaliado por um conjunto de testes correspondentes a $N = 100000$ iterações num ambiente de simulação em computador.

Os principais parâmetros do ambiente de simulação são apresentados na Tabela 1. Em cada iteração são gerados n pares $(elem_n, c_n)$, em que $elem_n$ é um identificador sequencial e c_n atribuído por uma distribuição pseudo-aleatória de valores uniformemente distribuídos no intervalo $[0, 1]$. Esta informação de mapeamento associativo é armazenada nas duas estruturas; i) determinística (*vetorElem[]*) e ii) FBL. A avaliação realizada tem em conta apenas os verdadeiros positivos, pois os elementos inseridos/consultados fazem parte do conjunto disponível no *vetorElem[]*.

Tabela 1: Parâmetros do Ambiente de Simulação

Atributo	**Valor/Expressão**
N (nº de iterações)	100000
dim(FBL)	4096 bits
m (células)	$\frac{dim(FBL)}{b}$
Erro de quantização (FBL_{ERR})	$\pm 1/(2^b - 1)$
4.2) Precisão (b)	1 bit $\leq b \leq$ 16 bits
4.3) Nº de funções de *hash* (k)	$2 \leq k \leq 16$

Nas subsecções 4.2 e 4.3 estão subjacentes as seguintes **métricas** para caracterizar os resultados obtidos com $N = 100000$ iterações:

- **Valor Médio** $\bar{\hat{c}}$ para o parâmetro confiança c, ver Eq.(5)
- **Desvio Padrão** σ para o parâmetro confiança c, ver Eq.(6)
- **Erro Quadrático Médio** do estimador $EQM(\hat{c})$, correspondente à média dos erros quadráticos, ver Eq.(7)
- **Taxa de ocupação** do filtro, correspondente à razão entre o número de bits a 1 e $(m \times b)$.

Equações das métricas estatísticas aplicadas aos resultados das simulações e utilizadas na avaliação do FBL:

$$\bar{\hat{c}} = \frac{1}{N \cdot n} \sum_{i=1}^{N \cdot n} \hat{c} \tag{5}$$

$$\sigma = \sqrt{\frac{1}{N \cdot n} \sum_{i=1}^{N \cdot n} [\hat{c} - \bar{\hat{c}}]^2} \tag{6}$$

$$EQM(\hat{c}) = \frac{1}{N \cdot n} \sum_{i=1}^{N \cdot n} [\hat{c} - c]^2 \tag{7}$$

Para implementar o FBL é necessário escolher um valor para k, que deve ser ajustado em função do ambiente de simulação (neste caso parametrizado na Tabela 1). Para obter uma aproximação teórica para o valor de k, assumiu-se que o número de elementos inseridos no FBL não ultrapassa os 100 elementos ($n = 100$) e que uma precisão de $b = 4$ bits é suficiente para representar c, então de acordo com a Eq.(2) obtém-se $k = \frac{4096/4}{100} \ln 2; k \approx 7$.

4.1 Impacto da precisão na capacidade do FBL

Como se pretende manter a dimensão do FBL (dim(FBL)) num valor fixo, o número de células disponíveis no FBL vai depender do número de bits usados em cada célula (precisão), ou seja dim(FBL)=$m \times b$.

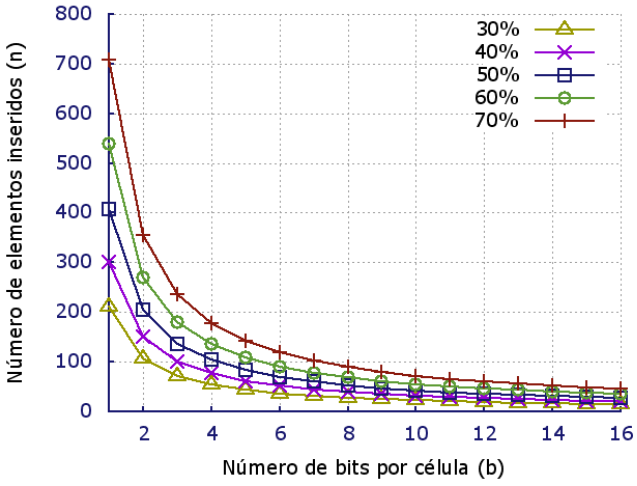


Figura 4: Impacto do nº de bits de precisão na capacidade do FBL

A Figura 4 compara diferentes cenários de carga considerando um FBL com $k = 7$ funções de *hash*, de acordo com a aproximação teórica apresentada acima. Neste conjunto de cenários a taxa de ocupação do FBL é mantida constante em valores pré-definidos entre 30% e 70%.

O processo utilizado para determinar o número de elementos a inserir para cada uma das taxas de ocupação em função dos número de bits de precisão, consistiu em inserir elementos ao FBL e parar quando a taxa de ocupação pretendida é atingida. Este processo de contagem foi repetido 1000 vezes, de modo a obter o inteiro mais próximo do valor médio dos elementos inseridos.

Na Figura 4 é possível observar a capacidade do FBL, correspondente à evolução do número de elementos inseridos no filtro em função do aumento de bits de precisão, garantindo uma taxa de ocupação constante.

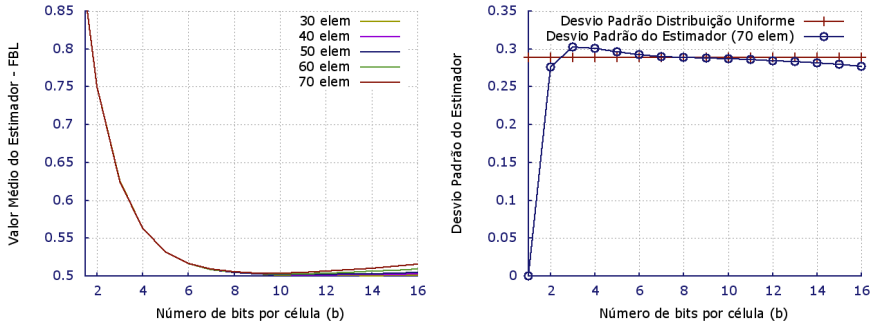
Como previsível, o número de elementos armazenado no FBL aumenta à medida que aumenta a percentagem de ocupação. Contudo, o diferencial entre taxas de ocupação é atenuado à medida que os elementos aumentam a sua precisão. Para armazenar cerca de 100 elementos no FBL e manter uma taxa de ocupação de 50% (Figura 4), o número máximo de bits para a quantização não deve ultrapassar os 4 bits por célula. O número de bits por célula tem um grande impacto na capacidade de armazenamento de dados num FBL de dimensão fixa, em especial na fase inicial do aumento da precisão. Analisando o gráfico apresentado na Figura 4 é possível identificar a ordem de grandeza da capacidade de armazenamento do FBL, que será usada durante as restantes subsecções da avaliação, conjuntamente com as parametrizações apresentadas na Tabela 1.

4.2 Impacto da calibração da precisão (b) no FBL

Para os testes de avaliação considerou-se constante o parâmetro $k = 7$ (ver Eq.(2)), variando o número de elementos inseridos (n) em função da precisão das células (b).

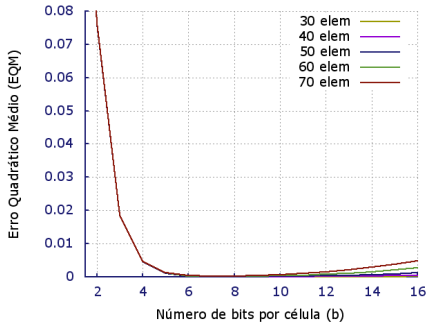
Para avaliar o comportamento do filtro em função da precisão b, são considerados cinco cenários correspondentes à inserção de 30, 40, 50, 60 e 70 elementos, comparando casos com taxas de ocupação reduzidas até zonas próximas da saturação. A Figura 5 apresenta quatro gráficos para caracterizar estatisticamente o estimador $\hat{c}$ proporcionado pelo FBL. Sabendo que c assume valores de uma distribuição uniforme no intervalo $[0, 1]$, o valor médio esperado é 0.5. Este valor foi comprovado durante a avaliação, pois o valor médio calculado com a informação armazenada estrutura determinística foi de $\overline{c} = 0.5005$. Para minimizar a perturbação causada por algum evento/característica que possa surgir durante a comparação da qualidade do estimador, foi escolhido o valor médio como indicador estatístico do estimador, ou seja $\overline{\hat{c}}$.

Na Figura 5a é possível observar como o $\overline{\hat{c}}$ se afasta desse valor ideal (linha base 0.5) para valores demasiado baixos de b, provocando erros significativos na quantização e induzindo o FBL a uma sobreestimação exacerbada. Para caracterizar a dispersão de valores face à média determinou-se o respetivo desvio padrão usando a Eq.(6), que para facilitar a leitura da Figura 5b apenas se apresenta

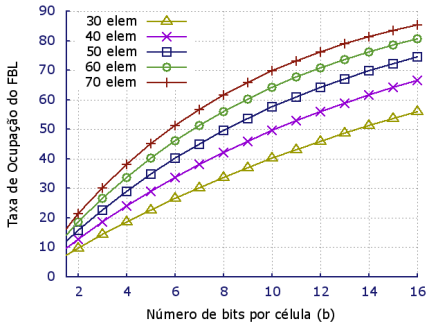


(a) Valor médio do estimador

(b) Desvio padrão do estimador



(c) Erro Quadrático Médio do estimador



(d) Taxa de ocupação do FBL

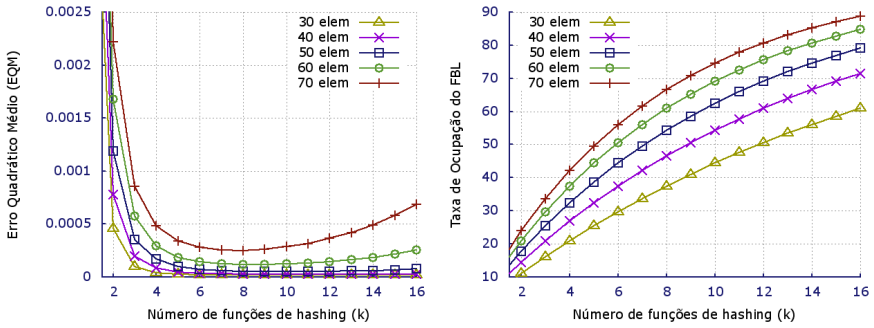
Figura 5: FBL - Influência da precisão (b) na calibração do estimador

o caso correspondente à inserção de 70 elementos. A comparação apresentada na Figura 5b permite confirmar que a dispersão do estimador é semelhante à dispersão característica de uma distribuição uniforme, exceto no caso em que $b = 1$ (que por usar apenas um único bit tem dispersão nula).

Para realizar a representação binária de c (no caso estudado) seriam suficientes 5 bits por célula. A Figura 5d permite observar a taxa de ocupação do filtro em função do número de elementos inseridos e do número de bits por célula. Na Figura 5c é possível observar que o erro do estimador se aproxima de zero quando $5 \leq b \leq 12$, ou seja existe um limite inferior mínimo e um limite máximo aconselhável para o número de bits por célula. A análise conjunta das Figuras 5d e 5c, permite concluir nos casos em estudo que: i) o erro com b inferior a 5 é originado por erros de quantização já que a taxa de ocupação do filtro é inferior a 50%; ii) quando b é superior a 12 a origem do erro resulta de uma utilização demasiado densa do filtro, pois a taxa de ocupação do filtro é superior a 50%, aproximando-se claramente da zona de saturação.

4.3 Impacto do número das funções de *hash* (k) no FBL

Para completar a avaliação do FBL foram considerados cinco cenários com uma variação dos elementos inseridos n semelhante à avaliação anterior, mas fixando uma precisão de $b = 8$ bits por célula (limitando o erro absoluto de quantização a $\pm 0,004$) e variando $2 \leq k \leq 16$. Estes cenários permitem avaliar o impacto que o número de funções de *hash* exercem sobre o valor médio do estimador, obtido ao interrogar o FBL.



(a) Erro Quadrático Médio do estimador (b) Taxa de ocupação do FBL

Figura 6: FBL - Influência do número de funções de *hash* (k) na calibração do estimador

Para o cenário com $n = 70$ elementos inseridos no filtro (com 100000 iterações) obteve-se $\overline{\hat{c}} = 0.5074 \pm 0.0051$ e $\sigma = 0.2882 \pm 0.0007$, ou seja o FBL não se afasta dos parâmetros típicos de uma distribuição uniforme no intervalo $x \in [0, 1]$, em que $\overline{x} = 0.5$ e $\sigma_x \simeq 0.2887$. Ao variar k o valor médio do estimador e o respetivo desvio padrão permaneceram praticamente constantes, ou seja a resposta do filtro não apresenta oscilações significativas para a média e desvio padrão.

A imagem da Figura 6a mostra claramente que o intervalo $4 \leq k \leq 12$ corresponde à zona de menor impacto no estimador, pois os valores do erro $EQM(\hat{c})$ são inferiores a 0.0005. O limite inferior assegura que em casos de sobreposição das funções de indexação, o erro do estimador não seja predominantemente influenciado por um k pequeno. Em contrapartida, não se pode aumentar demasiado o número de funções de *hash*, porque isso faz aumentar o espaço ocupado por cada elemento e atendendo que o FBL tem uma dimensão fixa pode provocar a saturação do FBL. Como se pode observar na Figura 6b para os casos em estudo, quando $k > 12$ o filtro aproxima-se do modo de saturação e o erro $EQM(\hat{c})$ começa a aumentar (comprar com Figura 6a).

A Figura 6a permite comprovar que o valor teórico obtido pela na Eq.(2) corresponde a uma zona onde o Erro Quadrático Médio é mínimo.

5 Conclusões

Este trabalho apresenta uma nova variação de um filtro de Bloom, designado por Filtro de Bloom Linear (FBL) que permite armazenar junto de cada elemento uma grandeza no intervalo [0,1], mantendo uma dimensão constante do filtro.

A motivação para o desenvolvimento do FBL surgiu da necessidade de difundir informação em redes WSN, mas esta nova estrutura de dados probabilística tem um carácter mais geral, podendo ser aplicável noutras áreas distintas.

A avaliação faz uma análise de sensibilidade dos parâmetros de configuração do FBL, assegurando que o erro do estimador esteja perfeitamente delimitado por comparação com a resposta de uma estrutura determinística. O estudo da calibração do b e do k permitiu encontrar fronteiras para estes parâmetros e caracterizar a técnica do FBL.

Referências

1. P. S. Almeida, C. Baquero, N. Preguica, and D. Hutchison. Scalable bloom filters. *Information Processing Letters*, 101(6):255 – 261, 2007.
2. B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.
3. F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese. An improved construction for counting bloom filters. In *Proceedings of the 14th Conference on Annual European Symposium - Volume 14*, ESA'06, pages 684–695, London, UK, UK, 2006. Springer-Verlag.
4. B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal. The bloomier filter: An efficient data structure for static support lookup tables. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 30–39, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.
5. B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher. Cuckoo filter: Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, pages 75–88, New York, NY, USA, 2014. ACM.
6. L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Trans. Netw.*, 8(3):281–293, June 2000.
7. D. Guo, Y. He, and Y. Liu. On the feasibility of gradient-based data-centric routing using bloom filters. *Parallel and Distributed Systems, IEEE Transactions on*, 25(1):180–190, Jan 2014.
8. D. Guo, J. Wu, H. Chen, Y. Yuan, and X. Luo. The dynamic bloom filters. *Knowledge and Data Engineering, IEEE Transactions on*, 22(1):120–133, Jan 2010.
9. K. Huang, J. Zhang, D. Zhang, G. Xie, K. Salamatian, A. Liu, and W. Li. A multi-partitioning approach to building fast and accurate counting bloom filters. In *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 1159–1170, May 2013.
10. A. Kalmar, R. Vida, and M. Maliosz. Context-aware addressing in the internet of things using bloom filters. In *Cognitive Infocommunications (CogInfoCom), 2013 IEEE 4th International Conference on*, pages 487–492, Dec 2013.

11. R. P. Laufer, P. B. Velloso, and O. C. M. B. Duarte. A generalized bloom filter to secure distributed network applications. *Comput. Netw.*, 55(8):1804–1819, June 2011.
12. X. Li, J. Wu, and J. Xu. Hint-based routing in wsns using scope decay bloom filters. In *Networking, Architecture, and Storages, 2006. IWNAS '06. International Workshop on*, pages 8 pp.–, 2006.
13. R. Lima, C. Baquero, and H. Miranda. Adaptive broadcast cancellation query mechanism for unstructured networks. In *9th International Conference on Next Generation Mobile Applications, Services and Technologies 2015 (NGMAST'15)*, Cambridge, United Kingdom, Sept. 2015.
14. Y. Qiao, T. Li, and S. Chen. Fast bloom filters and their generalization. *Parallel and Distributed Systems, IEEE Transactions on*, 25(1):93–103, Jan 2014.
15. S. Rhea and J. Kubiatowicz. Probabilistic location and routing. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1248–1257 vol.3, 2002.
16. S. Tarkoma, C. Rothenberg, and E. Lagerspetz. Theory and practice of bloom filters for distributed systems. *Communications Surveys Tutorials, IEEE*, 14(1):131–155, First 2012.
17. T. Yang, G. Xie, R. Duan, X. Sun, and K. Salamatian. Towards practical use of bloom filter based ip lookup in operational network. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–4, May 2014.