Universidade do Minho
Escola de Engenharia
Departamento de Informática

Dissertação de Mestrado
Mestrado em Engenharia Informática

Estimativa de funções de probabilidade cumulativa em redes de larga escala

Miguel Ângelo Borges da Silva

Trabalho efectuado sob a orientação de
Doutor Carlos Miguel Baquero Moreno

Outubro 2011

Declaração

Nome: Miguel Ângelo Borges da Silva

Endereço Electrónico: mab@lsd.di.uminho.pt

Telefone: 967253177

Bilhete de Identidade: 11030785

Título da Tese: Estimativa de funções de probabilidade cumulativa em redes de larga escala

Orientador: Doutor Carlos Miguel Ferraz Baquero Moreno

Ano de conclusão: 2011

Designação do Mestrado: Mestrado em Engenharia Informática

Universidade do Minho, 31 de Outubro de 2011

Miguel Ângelo Borges da Silva

Se, quando leio, encontro dificuldades, não roo as unhas por isso: após duas ou três investidas, deixo-as onde estão. Se as ficasse a remoer, perder-me-ia nelas e perderia o meu tempo, pois tenho um espírito impulsivo. Aquilo que não vejo à primeira carga, ainda menos o vejo em me obstinando. Nada faço sem alegria, e a persistência, assim como a contenção excessiva, obnubila-me o juízo, contrista-o e afadiga-o, e a minha vista torna-se então confusa e turva. Tenho de, a intervalos, desconcentrá-lo e voltá-lo a concentrar, exactamente como, para ajuizar do brilho do escarlate, devemos passar os olhos pelo tecido, percorrendo-o a diversos golpes de vista, rápidos, repetidos e reiterados.

Michel de Montaigne (Dos Livros)

Agradecimentos

Agradeço ao Professor Carlos Baquero por ter aceite ser meu orientador. Foram fundamentais para o sucesso deste trabalho o seu entusiasmo e criatividade. Agradeço a paciência que teve comigo e as boas conversas que tivemos. Boas conversas tidas também com o Professor Paulo Sérgio Almeida, cujo entusiasmo contagiante não me deixou esmorecer.

Ao Paulo Jesus por todo o apoio, paciência e conhecimento transmitido – não esqueço as discussões iluminadoras que tivemos.

À Sofia, Fausto e Vicente: Obrigado pela paciência e amor.

Resumo

A capacidade de agregar dados é uma característica fundamental na concepção de sistemas de informação escaláveis, que permite a determinação de propriedades globais importantes de forma descentralizada, para a coordenação de aplicações distribuídas, ou para fins de monitorização.

Agregados simples como mínimos/ máximos, contagens, somas e médias foram já extensivamente estudados no passado. No entanto, este tipo de agregados pode não ser suficiente para caracterizar distribuições de dados enviesadas e na presença de valores atípicos (*outliers*), tornando-se então relevante a determinação de uma estimativa dos valores na rede (e.g. histograma, função de distribuição cumulativa), dado que métricas como médias ou desvio padrão escondem em muitos casos alterações na propriedade monitorizada que são relevantes para decisão de controlo.

São ainda relativamente escassos os trabalhos que se focam sobre a agregação de métricas mais expressivas. Uma proposta recente nesse domínio [SNSP10] refere atingir uma precisão nas estimativas superior à atingida em abordagens anteriores. Trata-se de um algoritmo para a determinação de funções cumulativas de distribuições.

Apesar do contributo, essa proposta mostra limitações na tolerância a faltas e no suporte à monitorização contínua de propriedades, dado que para acompanhar alterações dos valores amostrados, a estratégia usada exige que o protocolo seja reiniciado periodicamente. Para além disso, os pressupostos dessa abordagem não admitem a perda de mensagens nem a sua duplicação.

Asim, e tomando como ponto de partida o actual estado da arte, é apresentado nesta tese um algoritmo distribuído para a determinação de funções cumulativas de probabilidade em redes de larga escala. As suas principais vantagens são a imunidade à perda de mensagens, a velocidade de convergência e a precisão que se obtém na

aproximação à distribuição original. É simultaneamente adaptável a alterações no valor amostrado e resiliente a dinamismo no número de nodos na rede. Usa também um mecanismo de quiesciência dos nodos assim que a variação local da estimativa é inferior a um determinado limiar. Nessa circunstância, o nodo deixa de transmitir. Isto leva à diminuição do número de mensagens trocadas entre nodos.

As distribuições determinadas em todos os nodos permitem a tomada de decisões que tirem partido do facto de se estar a agregar uma função probabilística. Assim o nodo pode excluir *outliers* ou observar determinados quantis da propriedade. Para além disso, cada nodo da rede possui uma estimativa global sobre o estado geral da propriedade distribuída, o que lhe permite também a tomada de decisões com base em conhecimento local.

São apresentados nesta tese resultados de simulação que confirmam a validade da abordagem seguida. É também apresentada uma revisão da literatura relacionada cujo âmbito incluiu as técnicas mais representativas da agregação de dados para métricas escalares e as técnicas de agregação de dados para métricas complexas.

Abstract

The ability to aggregate data is a fundamental feature in the design of scalable information systems, which allows the estimation of relevant global properties in a decentralized way in order to coordinate distributed applications, or for monitoring purposes.

Simple aggregates such as minima/ maxima, counts, sums and averages have been thoroughly studied in the past. Nonetheless, this kind of aggregates may not be comprehensive enough to characterize biased data distributions and in presence of outliers, making the case for richer estimates of the values on the network (e.g. histograms, cumulative distributed functions), since scalar metrics like average or standard deviation hide in many cases changes in the property that are relevant to the control decision.

The amount of scientific work is relatively scarce in what concerns more expressive aggregation metrics. A recent proposal within this domain [SNSP10] claims to obtain estimates with a better precision than in previous approaches. It is an algorithm for the estimation of cumulative distribution functions.

Despite the contribution, the proposal mentioned above is not fault tolerant and is also not sensible to the continuous variation of the sampled properties, for it demands the protocol to be restarted frequently in order to achieve quasi-continuous monitoring. Besides, the approach does also not admit loss or duplication of messages.

Having this scenario as a starting point, this work presents a distributed algorithm for the estimation of cumulative distribution functions over large scale networks of which the main advantages are immunity to message loss, convergence speed and precision of the estimate. It can also cope with changes of the sampled property and is resilient to churn. It has also a quiescence mechanism that allows nodes

to minimize communication cost by not exchanging redundant messages, whenever local variations of the estimate fall below a specified threshold.

The estimated cumulative distribution function allows nodes to take advantage of having a broader view of the properties on the network: they may exclude outliers or monitor particular quantiles of a property. Also, each and every node of the network has a local vision of the global state of the property, thus allowing nodes to make decisions based on local knowledge.

This thesis presents simulation results that support and validate the proposed approach. It also presents a state of the art that includes both representative techniques for scalar aggregates and representative techniques for complex aggregates.

Conteúdo

Lista de Figuras

Lista de Algoritmos

Capítulo 1

Introdução

A capacidade de agregar dados é uma característica fundamental na concepção de sistemas de informação escaláveis, que permite a determinação de propriedades globais importantes de forma descentralizada, para a coordenação de aplicações distribuídas, ou para fins de monitorização.

Agregados simples como mínimos/ máximos, contagens, somas e médias foram já extensivamente estudados no passado. No entanto este tipo de agregados pode não ser suficiente para caracterizar distribuições de dados enviesadas e na presença de valores atípicos (*outliers*), sendo relevante a determinação de uma estimativa dos valores na rede (e.g. histograma, função de distribuição cumulativa).

Métricas como médias ou desvio padrão escondem em muitos casos alterações na propriedade monitorizada que são relevantes para decisão de controlo. Um exemplo genérico é o da monitorização de grandezas que sigam distribuições assimétricas (monitorização numa rede de detecção de fogo, carga numa rede de processadores): a distribuição estatística dos atributos envolvidos não é satisfatoriamente descrita com, por exemplo uma média, dado que facilmente valores anómalos a podem enviesar. Por exemplo, num cenário em que num campo rectangular haja um conjunto de sensores térmicos espalhados, a ocorrência de um foco de fogo num dos cantos do campo (afectando apenas a leitura de alguns poucos sensores) corre o risco de ser ignorada numa configuração em que apenas a média de valores dos sensores é medida. Em contrapartida, este fenómeno anómalo localizado pode ser detectado caso a agregação de dados na rede que cobre a área em questão seja mais expressiva. Neste caso, é fundamental caracterizar o atributo como uma distribuição estatística,

i.e., fornecendo um histograma que resume esta distribuição. Num outro exemplo, numa rede P2P poderá ser útil saber o espaço em disco de todos os nodos da rede. Uma métrica escalar como a média, diz pouco sobre a distribuição do espaço em disco. No entanto, a caracterização do espaço em disco como uma distribuição permite perceber quais os nodos com ou sem espaço disponível.

A agregação de dados em redes de larga escala, com foco na agregação de métricas escalares, encontra-se já extensivamente estudada na literatura, tanto para redes estruturadas (cuja topologia decorre de uma organização lógica que se sobrepõe à organização física dos nodos) como para redes não estruturadas (sem exigência de organização lógica rígida). No entanto são ainda relativamente escassos os trabalhos que se focam sobre a agregação de métricas mais expressivas. Uma proposta recente nesse domínio [SNSP10] refere atingir uma precisão nas estimativas melhor que em abordagens anteriores. Trata-se de um algoritmo para a determinação de funções cumulativas de distribuições, com um foco particular na estratégia de criação dos intervalos de amostragem para a determinação da função. Apesar do contributo, essa proposta mostra limitações na tolerância a faltas e no suporte à monitorização contínua de propriedades, dado que para acompanhar alterações dos valores amostrados, a estratégia usada exige que o protocolo seja reiniciado periodicamente. Para além disso, os pressupostos desta abordagem não admitem perda de mensagens e é também sensível à duplicação de mensagens.

Partindo deste ponto, apresentamos neste trabalho um algoritmo rápido e robusto para a determinação de distribuições estatísticas em redes de larga escala, em concreto a função distribuição cumulativa, com base num algoritmo de cálculo distribuído de médias (*Flow Update* [JBA09]). Esta abordagem prova-se tolerante a faltas e permite a monitorização contínua de atributos em ambientes dinâmicos (sujeitos a alterações topológicas com variação no número de nodos presentes na rede e alteração dos valores amostrados). O algoritmo tem por base pressupostos fracos em relação à rede subjacente, não exigindo a necessidade de ser reiniciado qualquer que seja a circunstância. Para além disso, é também apresentada uma técnica, que denominamos de quiesciência, cujo objectivo se centra na redução do custo de comunicação entre nodos.

1.1 Estrutura da tese e contribuições

Estrutura Esta tese está organizada da seguinte forma: o trabalho relacionado é exposto no segundo capítulo, apresentando algoritmos distribuídos que determinam a agregação de dados – desde a agregação de métricas não escalares até à agregação de distribuições estatísticas. No terceiro capítulo é mostrado o modelo usado na determinação de CDFs em redes distribuídas bem como as características de rede e estratégia de disseminação de mensagens. Apresenta-se também uma descrição intuitiva do algoritmo seguida da formalização detalhada do algoritmo. É também apresentado o mecanismo de quiesciência. No penúltimo capítulo são apresentadas as métricas usadas para aferir a qualidade do algoritmo, resultados de simulação e a sua discussão. O último capítulo traça algumas conclusões gerais sobre o trabalho e apresenta futuras direcções de investigação.

Contribuições Desta tese resultou a publicação de um artigo longo [BBJA11] em acta do Simpósio de Informática 2011, Coimbra. Foi também feita pelo autor uma comunicação oral no mesmo Simpósio em Setembro de 2011.

Capítulo 2

Trabalho relacionado

Neste capítulo expõem-se alguns dos trabalhos cujo conteúdo é relevante no contexto desta tese. O capítulo inicia-se com uma exposição sobre a noção de agregação em redes de larga escala e aborda conceitos e técnicas úteis à compreensão do tema. De seguida são mostrados alguns dos trabalhos científicos desenvolvidos em torno da agregação de dados em redes de larga escala com foco na agregação de métricas escalares tanto para redes estruturadas (cuja topologia decorre de uma organização lógica que se sobrepõem à organização física dos nodos) como para redes aleatórias (sem exigência de organização lógica rígida). O capítulo prossegue então com a apresentação de técnicas de agregação de dados aplicada às distribuições estatísticas. É feita de seguida uma exposição breve sobre outros algoritmos de agregação cujas características intrínsecas não permitem que sejam atribuídos às duas categorias anteriores. O capítulo termina com uma exposição breve sobre a construção de histogramas e funções de distribuição cumulativa.

2.1 Agregação de dados em redes de larga escala

A agregação de dados em redes de larga escala pressupõe que a função usada na agregação dos dados possa ser feita de forma distribuída pelos diferentes nodos que compõem a rede e que em simultâneo seja insensível à ordem com que os diferentes nodos a calculam. Para a obtenção destas características, é importante que o algoritmo garanta as propriedades de comutatividade e associatividade. É também característica desejável da função de agregação a insensibilidade à sua aplicação

repetida a um mesmo *input*. Estas três propriedades formalizam-se do seguinte modo:

Dada uma função de agregação $\mathcal{A}$ que mapeia os elementos de um multiconjunto M de um domínio E num domínio S, $\mathcal{A} : \mathcal{M}^{\mathcal{E}} \rightarrow S$

Propriedade associativa: $\mathcal{A}(a, b, c) \equiv \mathcal{A}(\mathcal{A}(a, b), c) \equiv \mathcal{A}(a, \mathcal{A}(b, c))$

Propriedade comutativa: $\mathcal{A}(a, b) \equiv \mathcal{A}(b, a)$

Idempotência: $\mathcal{A}(a, a, a) \equiv \mathcal{A}(a, a) \equiv \mathcal{A}(a)$

O domínio S pode ser também ele um multiconjunto. No entanto, dada a natureza de compressão da função de agregação, a cardinalidade deste deverá ser sempre inferior à cardinalidade do multiconjunto de entrada.

Qualquer função de agregação que satisfaça simultaneamente a propriedade associativa e a propriedade comutativa, pode ser eficazmente distribuída por diferentes nodos da rede, devido ao facto de cada operação de agregação poder ser computada em nodos distintos e com ordem de computação distinta.

O trabalho apresentado em [JBA10] sugere a classificação dos diferentes algoritmos de agregação de dados em rede, tomando por base critérios baseado na estruturação das redes e nas estratégias de roteamento de dados. As redes podem organizar-se de forma estruturada, tipicamente em *spanning tree*[1], *cluster* ou *multipath*. Esta forma de organização topológica pressupõem a criação de uma rede lógica entre os nodos que compõem a rede. A relação entre os nodos determina o tipo de relação hierárquica.

Estes esquemas topológicos pretendem evitar a existência de ciclos na rede que levem à possibilidade de haver mensagens de conteúdo idêntico entregues a um mesmo nodo. A sua maior fragilidade revela-se na dificuldade em lidar com dinamismo nos nodos: é frequente em cenários reais o abandono da rede pelos nodos, promovendo assim a existência de *churn* e possíveis particionamentos na rede. Esta dinâmica pode acontecer quer porque os nodos *falham*, ou se movem fisicamente ou por problemas temporários com as comunicações. Do mesmo modo, é comum

[1]Uma *spanning tree* de um grafo G é uma árvore composta por todos os vértices e um subconjunto de arestas de G, por forma a que não se formem quaisquer ciclos.

o crescimento da rede de forma dinâmica devido à adição de novos nodos ou em consequência de um particionamento temporário que entretanto deixou de existir.

Se uma falha nos nodos mais próximos das folhas de uma árvore podem ter consequências negligenciáveis, o mesmo não acontece com os nodos próximo do nodo raiz: uma falha num desses nodos comprometerá a precisão da métrica agregada. Para além disso, o volume total de dados que transita na rede será tanto maior quando mais próximo do nodo raiz. Dado que o consumo energético de um nodo estar fortemente correlacionado com o volume de mensagens trocadas, esta propriedade das topologias hierárquicas pode ter por consequência um consumo energético suficientemente elevado junto dos nodos próximos da raiz que rapidamente comprometa toda a estrutura e, em função disso, a operação de agregação.

No caso das redes geométricas aleatórias, a relação topológica entre nodos é determinada pela relação de vizinhança que decorre da área de cobertura radioeléctrica. Nestas redes, é a estratégia de roteamento de mensagens que normalmente define a tipologia da rede. O roteamento pode ser feito por *flooding*, *random walk* ou *gossip*. A não estruturação como o pressuposto base de uma rede de sensores evita a necessidade de criar uma topologia lógica e a dificuldade em mantê-la sob dinamismo dos nodos. No entanto, a função de agregação deverá ser robusta quanto à possibilidade de haver mensagens repetidas (dado ser provável a existência de ciclos), devendo por isso satisfazer a propriedade de idempotência.

No roteamento de mensagens baseado em *flooding*, cada nodo é responsável pela transmissão das mensagens recebidas para o conjunto dos nodos vizinhos. Desta forma, inevitavelmente e num tempo relativamente curto, todos os nodos que compõem a rede deverão receber a mensagem. O *flooding* apresenta como desvantagem o elevado volume de mensagens trocadas e a possibilidade de dar origem a mensagens repetidas caso haja ciclos na estrutura que constitui a rede.

A disseminação por *random walk* é uma forma de distribuição de mensagens estocástica que se processa da seguinte forma: assim que um nodo recebe uma mensagem, este encaminha-o para um outro nodo escolhido aleatoriamente de entre o conjunto de nodos vizinhos. Inevitavelmente, todos os nodos da rede deverão receber a mensagem.

O *gossip* é inspirado em modelos epidémicos e é também um processo estocástico: assim que um nodo receba uma mensagem, este retransmite-a a um subconjunto

de nodos escolhido aleatoriamente de entre o conjunto de nodos vizinhos. À luz desta estratégia de roteamento, o *flooding* pode ser visto como *gossip* com uma disseminação equivalente ao grau de conectividade do nodo, ou seja, igual ao número total de nodos vizinhos. Igualmente, o *random walk* pode ser visto como uma estratégia de *gossip* com um grau de disseminação unitário.

As estratégias epidémicas descritas acima não necessitam normalmente de mecanismos de recuperação e apresentam um custo de comunicação ligeiramente superior ao roteamento em topologias estruturadas. Demonstram uma elevada resiliência e podem ser escaladas para uma elevada quantidade de nodos sem problemas.

A taxonomia apresentada em [JBA10], propõem um nível de classificação onde congrega todos os algoritmos existentes na literatura relacionada. De forma a tornar mais clara a diferença entre algoritmos de agregação de métricas escalares (mínimos, máximos, médias, contagens, etc.) e de medidas de distribuições, será útil subdividir a categoria de algoritmos em algoritmos escalares e algoritmos não escalares.

De seguida, abordam-se alguns dos algoritmos de agregação escalares mais relevantes na literatura actual. Na categoria de "redes com topologias lógicas", tomamos como representativos os algoritmos apresentados na secção seguinte:

2.2 Algoritmos de agregação de métricas escalares

Nesta secção é feita uma breve descrição do estado da arte no que concerne aos algoritmos de agregação com foco na determinação de métricas pertencentes ao domínio dos números reais. São apresentados diferentes algoritmos de agregação agrupados em duas categorias relacionadas com a topologia da rede subjacente.

2.2.1 Redes com topologia lógica

Entende-se rede com topologia lógica como uma rede cuja relação entre nodos segue um padrão bem definido, tal como por exemplo uma árvore. Dentro desta categoria de redes são apresentados alguns algoritmos ilustrativos de agregação de métricas escalares.

FM Sketches O algoritmo apresentado em [CLKB04] usa a técnica de *counting sketches* publicada em [FM85]. Esta técnica probabilística visa estimar o número de itens distintos numa base de dados, estando restríta a um espaço de memória limitado. Trata-se de uma técnica insensível à existência de *sketches* duplicados. A precisão da contagem é tanto maior quanto menor for a sensibilidade à existência de duplicados.

De uma forma breve, um *FM sketch* guarda numa lista indexada de tamanho k os valores *zero* ou *um*, de acordo com uma função de *hash* binária aplicada ao elemento que se pretende contabilizar no sumário. Esta técnica exibe propriedades interessantes que são exploradas pelos autores para o desenho do processo de agregação: o *sketch* da união de dois multiconjuntos[2] é igual à operação de disjunção bit a bit dos seus *sketches* individuais. Para além disso o *sketch* de um multiconjunto é determinado apenas pelos itens distintos no multiconjunto - nem a duplicação nem a ordem dos seus elementos afecta o seu *sketch*. Os *FM sketches* são também dotados das seguintes propriedades: há um índice do *sketch* abaixo do qual os valores guardados são 1, com uma probabilidade delimitada e dependente do número de elementos distintos. Há um índice do *sketch* acima do qual os valores guardados são 0 com uma probabilidade também delimitada e dependente do número de elementos distintos. Estas propriedades são úteis na medida em sugerem que a cardinalidade do conjunto de elementos distintos sumarizada no *sketch* poderá ser aproximada pelo número de dígitos a 1 no prefixo do *sketch*.

A partir destas propriedades, os autores generalizam a técnica de contagem proporcionada pelos *sketches* à soma de valores. Para tal, consideram que os elementos do multiconjunto M são agora pares de elementos que contêm o valor a somar e a identidade do valor (c_i, k_i). A soma pode assim ser calculada pelo somatório dos valores para todos os pares valor/identidade distintos: $s \equiv \sum_{distinct((c_i,k_i)\in M)} c_i$

Os autores propõem a utilização desta técnica para a determinação de agregados de soma na rede, usando técnicas de roteamento multi-caminho garantindo em simultâneo um custo computacional e de comunicação baixo. O algoritmo funciona em duas fases: numa primeira fase, o nodo monitor envia por *flooding* uma mensagem de pedido de agregação. À medida que a mensagem se espalha na rede,

[2]Um multiconjunto $M = x_1, x_2, x_3, \ldots$ representa os valores que se pretende ver sumarizados na forma de um *sketch*.

é construída uma topologia hierárquica[3]. Numa segunda fase, os nodos criam os seus *sketches* locais e transmitem-nos para os nodos pai. Os nodos pai recebem os *sketches* dos seu filhos e criam um *sketch* parcial que é também enviado aos seus pais. Inevitavelmente, o nodo raiz recebe todos os *sketches* parciais que combina para obter o resultado final.

TAG A abordagem apresentada em [MFHH02] consiste na utilização de uma linguagem declarativa semelhante ao SQL para inquirição da rede de sensores. É criada uma rede em árvore a partir do nodo monitor. Numa primeira fase o nodo monitor distribui a "query" pelos restantes nodos da rede – esta operação fixa a estrutura topológica usada posteriormente na fase de recolecção. De seguida, os nodos agregam os dados no sentido do nodo raiz: os nodos folha transmitem um tuplo ao nodos pai. Cada nodo pai combina os tuplos recebidos e enviam o resultado ao seu nodo pai. Este processo é repetido até que o nodo monitor receba o agregado. Para além da operação de agregação, a estratégia descrita no *TAG* estipula a existência de um estado quiesciente durante o qual tanto o processador como o sistema rádio se encontram adormecidos. Os nodos apenas acordam em sincronia com um temporizador, ou na ocorrência de eventos de comunicação externos. O processamento e encaminhamento de mensagens são feitos no estado de "acordado", voltado o nodo de seguida ao estado quiesciente. Esta estratégia permite baixar o custo de processamento e de comunicação. Dado criar uma topologia em árvore, esta técnica é susceptível de pôr em causa a qualidade da estimativa na presença de falhas de nodos ou de mensagens.

Esta estratégia está preparada para a agregação das seguintes métricas: Mínimo, máximo, média e contagem de nodos. Decorrente do facto de no *TAG* se criar uma topologia em árvore, caso haja perda de mensagens ou falhas nos nodos a métrica fica irremediavelmente comprometida.

2.2.2 Redes geométricas aleatórias

Nas redes geométricas aleatórias (cuja topologia não obedece a um critério lógico de estrutura) as soluções propostas na literatura actual para a agregação de dados são mais vastas. Os algoritmos que operam sobre estas redes focam-se na estimativa de

[3]Em detalhe, um grafo dirigido acíclico ou na literatura, um DAG - *Directed Acyclic Graph*.

somas, médias, simples contagens na rede ou determinação de máximos ou mínimos.

De modo breve, pode explicar-se a estimativa de uma contagem numa rede geométrica aleatória da seguinte forma: todos os nodos da rede tomam como amostrado o valor 0 excepto um deles, que toma o valor 1. O nodo cujo valor é 1 envia para todos os nodos vizinhos (usando *flooding*) um valor igual à fracção do seu valor amostrado pelo número de vizinhos: dado um conjunto de vizinhos N, o valor enviado para cada um deles será $1/ \mid N \mid +1$, tomando para si como estimativa o valor $1/ \mid N \mid +1$. Iterativamente, todos os nodos que recebam uma estimativa procedem de modo idêntico, distribuindo uma fracção do valor recebido pelos seus nodos vizinhos. Inevitavelmente, e dada uma rede de tamanho D, todos os nodos convergirão para o valor $1/D$. Deste modo, cada nodo pode estimar o tamanho da rede como sendo o recíproco do valor para o qual convergiram $((1/D)^{-1})$.

A estimativa de médias e somas pode ser feita de modo idêntico. O conceito subjacente a todas estas formas de estimativa em modo distribuído é mencionado na literatura como "distribuição de massa" [KDG03].

De entre os algoritmos distribuídos para a estimativa de métricas escalares em redes geométricas aleatórias, destacam-se os seguintes:

Push-sum O algoritmo de *Push-sum* é apresentado em [KDG03]. Trata-se de uma algoritmo distribuído para a estimativa da média e soma dos valores amostrados nos nodos. Pode também ser usado para a contagem do número de nodos que constituem a rede. O roteamento de mensagens entre nodo é feito por *gossip* e processa-se da seguinte forma:

1. Num dado instante t, cada nodo i da rede mantém um par de valores soma/peso $(s_{t,i}, w_{t,i})$. O nodo é inicializado com a sua soma igual ao valor amostrado x_i e com peso 1 – $(s_{0,i} = x_i, w_{0,i} = 1)$

2. Em rondas subsequentes, o nodo envia para si próprio e para um nodo aleatório dos conjunto dos nodos vizinhos o tuplo $(s_{t,i}/2, w_{t,i}/2)$.

3. Os tuplos dos nodos vizinhos j recebidos num nodo i determinam a nova estimativa local: $(s_{t,i}, w_{t,i}) = (\sum_{\forall j} s_{t-1,j}, \sum_{\forall j} w_{t-1,j})$.

Em cada instante t, a estimativa da média num dado nodo i é dada pelo quociente entre a sua estimativa de soma e a sua estimativa de peso actuais. A precisão da

estimativa é tanto maior quantas mais rondas decorrerem.

Com o mesmo algoritmo, é possível estimar a soma dos valores nos nodos, bastando para isso o peso $w_{t0,i}$ ser fixado em 1 apenas num dos nodos da rede, mantendo os restantes a 0. Da mesma forma, para efectuar a contagem de nodos, basta fixar o valor amostrado $s_{t,i}$ em 1 para todos os nodos e peso 1 em apenas um dos nodos da rede. Os autores introduzem o conceito de *conservação de massa* e postulam que a média de todos os $s_{t,i}$ é correcta e a média de todos os $w_{t,i}$ em qualquer instante t é constante e igual ao número de nodos da rede. Na eventualidade de um nodo perder uma mensagem ou de desaparecer da rede, este postulado é violado. Os autores admitem a existência de um mecanismo de detecção de falhas que actua da seguinte forma: caso um nodo detecte que a sua mensagem não foi entregue, este entrega-a a si próprio.

Push-pull O protocolo *Push-pull* [Jel04] permite, tal como o *Push-sum*, que todos os nodos que compõem a rede obtenham uma estimativa local da propriedade global. Os autores propõem um mecanismo de agregação denominado de *anti-entropia* que permite reduzir a variância da média global da rede de uma forma iterativa. O algoritmo de agregação funciona da seguinte forma:

1. Um dado nodo i aguarda um instante pré-definido.
2. Num dado instante t, o nodo escolhe aleatoriamente um vizinho j do conjunto de nodos vizinhos N e envia a sua estimativa $x_{i,t}$ para o nodo j.
   (a) O nodo vizinho j recebe a estimativa de i , $x_{i,t}$ e envia a sua estimativa $x_{j,t}$ para i.
   (b) O nodo vizinho j calcula a sua nova estimativa $x_{j,t+1}$ em função da sua estimativa actual e da estimativa recebida.
3. Entretanto, o nodo i terá recebido do vizinho escolhido a estimativa $x_{j,t}$.
4. Calcula a sua nova estimativa $x_{i,t+1}$ em função da sua estimativa actual $x_{i,t}$ e da estimativa recebida $x_{j,t}$, e.g., a estimativa da média resulta de $\frac{1}{2}(x_{i,t} + x_{j,t})$.

Os autores assumem a existência de nodos com identificadores únicos, numa rede totalmente ligada, e assumem também um mecanismo global de sincronização.

DRG O algoritmo *DRG (Distributed Random Grouping)* [CPX06] usa criação probabilística de grupos de nodos numa rede de sensores para fazer convergir cada um deles para a estimativa da propriedade. Em cada ronda de agregação, o algoritmo processa-se do seguinte modo:

1. Um nodo da rede, em estado *adormecido*, decide de forma independente e com uma probabilidade p tornar-se líder de um grupo. Entra então em modo *líder* e transmite uma mensagem de agregação de grupo com identidade i, ficando a aguardar a resposta dos nodos vizinhos.

2. Um nodo vizinho j em modo *adormecido* que receba a mensagem de pedido de agregação, responde ao nodo líder i com a sua estimativa actual v_j e com a sua identidade j. Então, o nodo j entra em modo *membro de grupo* e aguarda instruções do líder de grupo.

3. O líder de grupo recolhe então todas as informações recebidas na sequência do seu pedido de informação de grupo, e calcula o valor médio dos valores amostrados recebidos. De seguida, o líder envia para os nodos do grupo o valor médio calculado e retorna ao estado *adormecido*.

4. Os nodos do grupo, no estado de *membro de grupo*, recebem a média calculada pelo líder, actualizam a sua estimativa com esse valor e retornam ao estado *adormecido*.

A estimativa é calculada ao longo de várias iterações do algoritmo, durante as quais vários novos grupos se vão formando aleatoriamente. O sucesso desta estratégia está directamente relacionado com a escolha da probabilidade de formação de grupos, para garantir que haja ao longo do tempo sobreposição entre membros de grupos, para que a estimativa consiga ser difundida por toda a rede.

Flow Update A técnica de *flow update* [PJA10] exige poucos pressupostos da rede subjacente. É tolerante a faltas e exibe boa capacidade de lidar com dinamismo na rede. Ao invés da abordagem usada em [KDG03], em que a eventual perda de massa pressupõem divergência na estimativa, no *flow update* a massa é conservada, mesmo na circunstância em que se perdem mensagens, devido ao facto de todos os nodos preservarem o seu valor amostrado. Esta técnica baseia-se num conceito simples da

Algoritmo 1: Algoritmo de *Flow Update*.

inputs:
  v_i, valor amostrado
  $\mathcal{D}_i$, conjunto de vizinhos, dado por um detector de falhas
variáveis de estado:
  fluxos: inicialmente, $F_i = \{\}$
função geradora de mensagens:
  $\text{msg}_i(F_i, j) = (i, f, \text{est}(v_i, F_i))$;
  with $f = \begin{cases} F_i(j) & \text{if } (j, \_) \in F_i \\ 0 & \text{caso contrário} \end{cases}$
função de transição de estado:
  $\text{trans}_i(F_i, M_i) = F_i'$
  com
  $F = \{j \mapsto -f \mid j \in \mathcal{D}_i \wedge (j, f, \_) \in M_i\} \cup$
    $\{j \mapsto f \mid j \in \mathcal{D}_i \wedge (j, \_, \_) \notin M_i \wedge (j, f) \in F_i\}$
  $E = \{i \mapsto \text{est}(v_i, F)\} \cup$
    $\{j \mapsto e \mid j \in \mathcal{D}_i \wedge (j, \_, e) \in M_i\} \cup$
    $\{j \mapsto \text{est}(v_i, F_i) \mid j \in \mathcal{D}_i \wedge (j, \_, \_) \notin M_i\}$
  $a = (\sum\{e \mid (\_, e) \in E\})/\,|E|$
  $F_i' = \{j \mapsto f + a - E(j) \mid (j, f) \in F\}$
função estimadora:
  $\text{est}(v, F) = v - \sum\{f \mid (\_, f) \in F\}$

teoria de grafos: o fluxo. O fluxo dirigido ao longo de um vértice entre um nodo i e um nodo j, na forma de um número real, é simétrico do fluxo entre o mesmo nodo j e o nodo i. Combinando o conceito de preservação de valor inicial com a transmissão de informação sobre fluxos, o *flow update* envia valores de fluxo entre nodos adjacentes cujo significado intuitivo é o de sinalizar ao nodo destinatário a "quantidade" com que este deve afectar a sua estimativa para que ambas se aproximem. Ou, dito de outra forma, o fluxo serve para diminuir a variância entre estimativas. O algoritmo é apresentado em detalhe em 1.

Este algoritmo assume a existência de um detector de faltas que indica a um dado nodo o conjunto de nodos vizinhos nesse estado. O seu modelo de execução baseia-se no modelo pubicado em [Lyn96]: estado inicial arbitrário e canais de comunicação

vazios. Depois em rondas sucessivas, gera as mensagens a enviar aos nodos vizinhos aplicando a função de geração de mensagens ao seu estado actual. De seguida computa o seu novo estado com base no estado actual e nas mensagens de entrada.

O conteúdo das mensagens transmitidas ao conjunto de nodos vizinhos é constituído por um identificador do nodo origem, o fluxo calculado para esse nodo (porquanto pertença ao conjunto de vizinhos) e uma estimativa (função do seu valor amostrado e do seu fluxo), linhas 7–8 do Algoritmo 1.

O estado de transição de um nodo resulta de uma operação sobre os conjunto dos fluxos actuais do nodo e sobre o conjunto das mensagens por si recebidas dos nodos vizinhos, resultando num novo estado F_i' (linhas 10–18). A função de transição de estado usa duas variáveis auxiliares: F, o conjunto que mantém o valor simétrico dos fluxos recebidos em mensagens dos nodos vizinhos ou o seu valor actual caso nenhuma mensagem tenha sido recebida; e o conjunto de estimativas E que mapeia estimativas para nodos vizinhos.

Do conjunto E de estimativas, o nodo i guarda uma estimativa com base no seu valor amostrado e conjunto de fluxos F recém calculado; guarda também para os nodos vizinhos dos quais tenha recebido mensagens, a estimativa que lhes enviou; e, caso não tenha recebido mensagem de um nodo vizinho, retém uma estimativa com base nos fluxos recebidos por esse vizinho na ronda anterior.

A variável a mantém o valor médio do conjunto das estimativas E (linha 17). A função de estimativa (linha 20) resulta da afectação do valor amostrado pelo somatório do conjunto dos fluxos F. O mapa de fluxos F_i' (linha 18) é calculado afectando as estimativas E com os fluxos F para que estas se aproximem da média a.

O funcionamento do algoritmo descrito acima pode ilustrar-se com uma rede simples de dois nodos com valores iniciais $v_i = 1, v_j = 2$ (ver também para maior clareza, a Figura 2.1). Assim, temos:

1. Inicialmente, ambos os nodos dispõem dos seus valores iniciais; A estimativa num nodo corresponde à soma do valor amostrado com os valores actuais de fluxos. Dado não ter ocorrido ainda troca de mensagens, a estimativa é de valor igual ao valor inicial e fluxos são nulos.

2. Na primeira ronda r_1, trocam as suas estimativas e fluxos – o nodo i recebe o

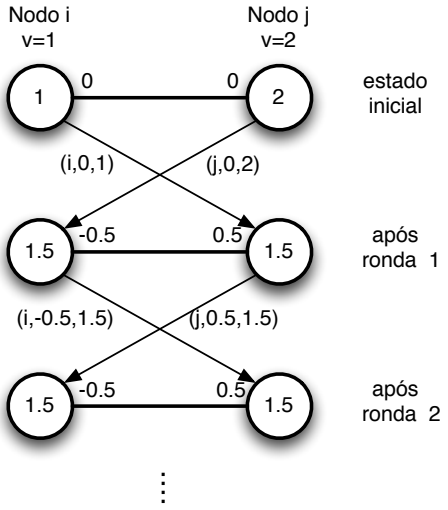

Figura 2.1: Algoritmo de *flow update* aplicado a dois nodos.

valor 2 e o nodo j recebe o valor 1. Os fluxos trocados são iguais a 0.

- O nodo i calcula a sua nova estimativa com base nos valores recebido de j. Como o fluxo de j é nulo, este não contribui para a nova estimativa de i: $e_i = \frac{1}{2}(v_i + e_j) = 1.5$. Com base neste valor, i calcula também o fluxo a devolver a j na próxima ronda: $f_{i\to j} = e_i - e_j = -0.5$, que intuitivamente se interpreta como: de acordo com a estimativa actual de i, o nodo j deve afectar a sua estimativa em -0.5 para que ambas se aproximem.
- Do mesmo modo, o nodo j calcula a sua nova estimativa: $e_j = \frac{1}{2}(v_j + e_i) = 1.5$. O Fluxo a devolver a i na ronda seguinte: $f_{j\to i} = e_j - e_i = 0.5$.

3. Na ronda r_2 os nodos trocam as suas novas estimativas e fluxos. O nodo i recebe $(e_j = 1.5, f_{j\to i} = 0.5)$ e o nodo j recebe $(e_i = 1.5, f_{i\to j} = -0.5)$.

   - O nodo i calcula a sua nova estimativa, função do seu valor inicial, da estimativa recebida de j e do fluxo também recebido de j. Como o fluxo agora é não nulo, aplica-se a propriedade de simetria: o fluxo recebido de j é visto por i como o seu simétrico $f_{j\to i} = -f_{j\to i}$. Assim, a nova estimativa de i é: $e_i = \frac{1}{2}(v_i - f_{j\to i}) + e_j = 1.5$. O nodo i calcula então o novo fluxo a entregar na ronda seguinte a j, agora afectando a estimativa calculada com o simétrico do fluxo recebido de j: $f_{i\to j} = -f_{j\to i} + (e_i - e_j) = -0.5$.

- O nodo j procede de modo idêntico. Calcula a sua nova estimativa, considerando o fluxo simétrico recebido de i: $e_j = \frac{1}{2}(v_j - f_{i\to j}) + e_i = 1.5$ e o fluxo a devolver a i na ronda seguinte: $f_{j\to i} = -f_{i\to j} + (e_j - e_i) = 0.5$.

4. A partir desta ronda tanto os fluxos como as estimativas se encontram em equilíbrio e com valor médio exactamente igual ao valor médio dos valores presentes na rede.

Intuitivamente, pode verificar-se o comportamento deste algoritmo em caso de falha de um dos nodos, pressupondo a existência de um detector de faltas: O nodo que falha deixaria de enviar mensagens pelo que na ronda seguinte a esse evento, o nodo sobrevivente convergiria para o seu valor inicial, uma vez que a estimativa nesse caso seria função de quociente unitário, dado haver apenas uma estimativa em E e $j \notin n_i$.

Extrema propagation Neste trabalho [BAM09], os autores apresentam um algoritmo probabilístico para o cálculo de somas e estatísticas ordinais em redes de larga escala com base em operações idempotentes (insensíveis à aplicação sobre mensagens duplicadas) e logo, apropriado para redes não estruturadas. De forma intuitiva o algoritmo funciona da seguinte forma: Admitindo que todos os nodos que compõem a rede geram um número aleatório de acordo com uma distribuição estatística parametrizada (normal, exponencial ou outra) comum a todos, é expectável que o mínimo global desses valores siga uma distribuição cujos parâmetros dependem do número de nodos na rede. Ora, como a operação de agregação de mínimos (e também de máximos) é idempotente, esta pode ser aplicada trivialmente em redes não estruturadas usando *gossip*. Com um pouco mais de detalhe o algoritmo funciona da seguinte forma:

1. Cada nodo gera um vector de números aleatórios de acordo com uma distribuição estatística $rDist$.

2. Sempre que um nodo recebe o vector de um dos seus nodos vizinhos, este estima o seu novo vector como sendo o mínimo dos dois vectores (componentes emparelhadas). Envia de seguida o novo vector para o conjunto dos seus nodos vizinhos.

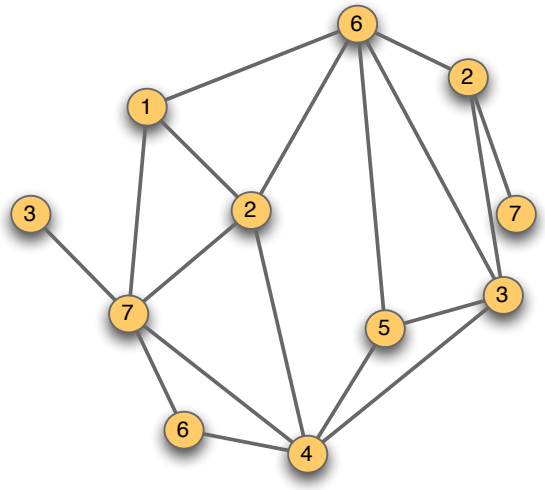

Figura 2.2: Representação de um conjunto de nodos com os respectivos valores amostrados $\langle 1, 6, 3, 5, 3, 4, 7, 2, 6, 7, 2 \rangle$.

3. A cada ronda, qualquer dos nodo pode estimar o tamanho da rede usando para isso um estimador de máxima verosimilhança a partir dos valores presentes no vector.

Os autores mostram que a função estimadora de contagem de nodos não enviesada $\hat{N}$, para o cenário em que os valores dos vectores nos nodos são gerados a partir de uma distribuição exponencial com $\lambda = 1$ e para um vector x de números aleatórios de tamanho K é $\hat{N} = \frac{K-1}{\sum_{i=1}^{K} x[i]}$. Não foi preocupação central neste trabalho a precisão da medida mas antes a rapidez com que se chega a uma estimativa do tamanho da rede (devido ao facto de se usar uma operação idempotente). A precisão do estimador é função do tamanho K do vector.

2.3 Algoritmos de agregação de distribuições estatísticas

Neste tipo de algoritmos pretende-se estimar a distribuição estatística dos valores presentes no conjunto dos nodos que constituem a rede. Ilustrando, dada uma rede cujos nodos apresentam valores amostrados $V_i = \langle 1, 6, 3, 5, 3, 4, 7, 2, 6, 7, 2 \rangle$ (ver ilustração na Figura 2.2), uma possível representação discreta da distribuição dos valores na rede poderia ser feita na forma de um histograma (ver Figura 2.3) com quatro

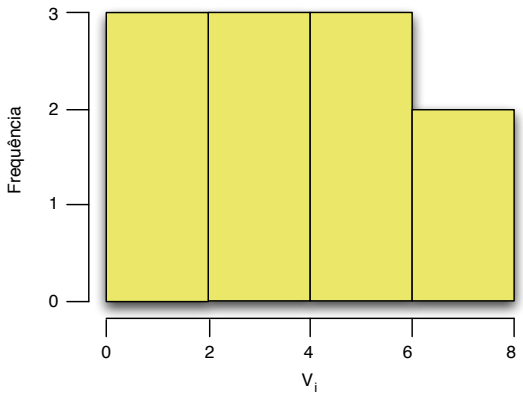

Figura 2.3: Histograma com quatro classes para $\langle 1, 6, 3, 5, 3, 4, 7, 2, 6, 7, 2 \rangle$.

classes (intervalos). O mesmo conjunto de valores discretos pode ser representado na forma de um histograma cumulativo (ver Figura 2.4).

Apresentam-se em seguida algoritmos representativos da agregação de distribuições de dados em redes de larga escala.

Equi-depth A estratégia de agregação apresentada em [HR08] pretende que numa rede não estruturada, usando roteamento *gossip*, todos os nodos da rede determinem uma distribuição estatística de todos os valores presentes na rede. Os autores admitem a existência de mensagens duplicadas e, ao invés de tentar mitigar a sua presença, assumem que o seu efeito é negligenciável em face do objectivo: dotar os nodos de uma visão geral da distribuição dos valores na rede, sem que esta seja com precisão. O protocolo proposto processa-se em fases e estas consistem em várias rondas. No fim de cada fase, os nodos da rede (com um identificador único na rede) deverão obter uma aproximação dos valores presentes nos nodos no início da fase. Em fases subsequentes, o valor amostrado pelos nodos pode variar. Em detalhe, numa dada fase, o algoritmo processa-se do seguinte modo:

1. Todas as rondas têm uma duração fixa δ.

2. Os nodos têm um vector de k elementos, inicializado com os seu valor observado v_i no início de uma fase.

3. Em cada ronda, um nodo escolhe aleatoriamente um vizinho j do qual solicita o seu vector. O nodo passa agora a ter um vector de tamanho $2k$ que é depois comprimido numa operação denominada de *Swap* por forma a ficar

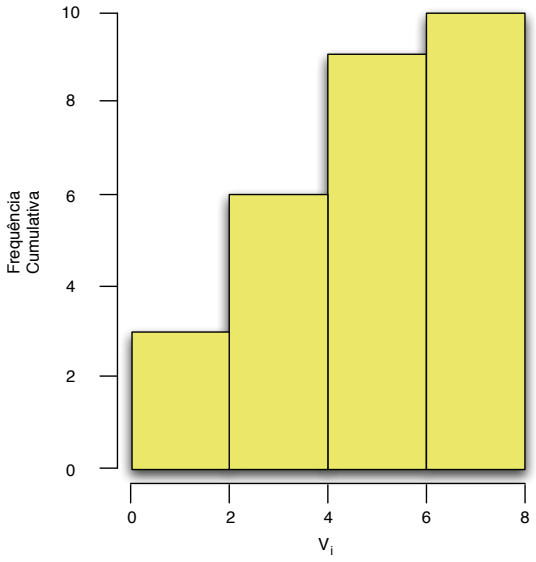

Figura 2.4: Histograma cumulativo com quatro classes representativo da lista de valores $\langle 1, 6, 3, 5, 3, 4, 7, 2, 6, 7, 2 \rangle$.

com tamanho k. Na forma mais elementar, a operação de *Swap* consiste em descartar aleatoriamente k elementos do vector.

Para evitar a perda de informação e reduzir o consumo de espaço, os autores propõem três técnicas de construção de sinopses, cujo efeito nos dados é mais "benigno" que o simples descartar aleatório:

Concise counting Com esta estratégia de construção de sinopse, o vector estimativa guarda um tuplo $(valor, contador)$. Assim, sempre que são adicionados novos tuplos ao vector, este é ordenado pelo *valor* e os tuplos cujo valor é mais próximo são fundidos num só tuplo. A fusão consiste na escolha aleatória de entre os valores dos tuplos candidatos à fusão e a soma dos seus respectivos contadores.

Equi-width histograms Para a construção de histogramas equi-espaçados (*equi-width*) pretende-se formar intervalos de tamanho idêntico entre valores. A dificuldade está em que os nodos não têm a noção dos limites mínimo e máximo dos valores presentes na rede. No trabalho em questão, os autores admitem que o vector do histograma toma os valores entre zero e o seu valor amostrado v_i na ronda inicial. Em rondas subsequentes, à medida que novos valores são conhecidos pelo nodo, os limites mínimo e máximo do histograma são

actualizados. Simultaneamente, todos os valores intermédios do vector são escalados de acordo com os novos limites. Os contadores associados a cada valor são redistribuídos pelos novos valores.

Equi-depth histograms Na construção de histogramas equi-frequentes (*equi-depth*), cada tuplo do vector deverá conter o elemento contador aproximadamente idêntico. O vector de histograma contém inicialmente valores entre zero e v_i. Esta estratégia usa operações de partição (*split*) e fusão (*merge*) de intervalos à medida que novos valores são conhecidos. Quando um novo vector é dado a conhecer ao nodo, este ordena os tuplos $(valor, contador)$ e determina quais os valores consecutivos que, quando combinados, resultam no intervalo mais pequeno. Esta operação de fusão resulta na soma dos contadores dos dois intervalos e o novo valor resulta da média aritmética dos intervalos. Este processo repete-se até restarem k tuplos no vector.

H-GAP; Jurca, Dan and Stadler, Rolf [JS10] Trata-se de um protocolo para a determinação de histogramas de propriedades distribuídas numa rede de nodos, descentralizado e assíncrono, baseado num *overlay* em *spanning tree*. O protocolo pressupõe a existência de um nodo de controlo para o qual fluem as agregações. De forma a evitar o "estrangulamento" dos nodos próximos da raiz da árvore, o algoritmo estipula que em cada nodo existem filtros cuja função é conjugar em tempo real a precisão global da estimativa com o *overhead* do protocolo. Os dados são agregados incrementalmente no sentido das "folhas" da árvore para o nodo monitor. A comunicação entre nodos é feita em modo assíncrono, usando uma abordagem "push based". Este protocolo tem por objectivo a monitorização de grandezas em rede. A dinâmica dos nodos da *spanning tree* é gerida pelo protocolo GAP [DS05] (Generic Aggregation Protocol), tratando-se de um protocolo de agregação assíncrono e distribuído que gere a entrada e saída de nodos de uma *spanning tree* BFS[4].

O protocolo reduz o *overhead* da agregação, expresso em número de mensagens processadas na rede por unidade de tempo, admitindo um limiar de variação dos valores iniciais e dos histogramas agregados nos nodos da rede. Caso entre duas agregações consecutivas a sua diferença seja inferior ao limiar estabelecido, o nodo

[4]BFS (Breadth First Search): trata-se de um algoritmo de pesquisa em árvores, que se inicia no nodo raiz e explora todos os nodos descendentes mais próximos. Termina assim que encontrar a solução pretendida ou toda a árvore seja percorrida.

não envia a agregação para os nodos pai.

Parameter-Based Data Aggregation; Hongbo Jian et. al. Em [JJW10], são apresentadas técnicas de agregação de dados *in-network*, escaláveis baseadas num *modelo de mistura*[5] para a determinação da distribuição dos dados. A proposta de agregação apresentada nesse trabalho assenta na transmissão entre nodos de apenas os parâmetros que definem as distribuições estatísticas do modelo de mistura. Para controlar o custo de comunicação, o algoritmo proposto admite um aumento do erro da estimativa.

O processo de agregação toma a direcção do nodo receptor e usa uma estratégia de "multi-path routing", o que permite a aumentar a tolerância a faltas ainda que aumentando o custo de comunicação. A agregação nos nodos intermédios adiciona as amostras locais à agregação recebida e encaminha-a na direcção do nodo receptor. O processo é repetido iterativamente até que o nodo receptor receba e agregue os resultados finais. O grafo de encaminhamento de mensagens é construído por *flooding*[6].

Um nodo transmite um tuplo contendo um peso w e o conjunto de parâmetros que caracteriam o modelo de mistura Θ. O novo peso do tuplo é directamente proporcional à soma dos pesos recebidos e inversamente proporcional ao número de sucessores imediatos afectado de um rácio de perdas. Logo, este factor reflecte o processo de particionamento, uma vez que no nodo seguinte a operação de agregação será aplicada de novo. O rácio de perdas de mensagens é introduzido para modelar perdas de mensagens na rede. Este parâmetro é configurado *à priori* e tem por objectivo estimar o potencial de perdas na rede. De notar que caso este parâmetro seja mal estimado, a agregação determinada deverá ser errada.

O desafio chave neste algoritmo de agregação é a estimação dos parâmetros da função de distribuição. Os autores resolvem-no usando um algoritmo de *Expectation Maximization*[7] para aproximar em cada nodo o valor amostrado a um modelo de

[5] ou *Mixture model*: é um modelo probabilístico usado para caracterizar estatisticamente sub-populações dentro de uma população combinando distribuições estatísticas da mesma família paramétrica caracterizadas por parâmetros distintos.

[6] O nodo raiz envia uma mensagem aos seus vizinhos com um pedido de agregação e estes fazem o mesmo, e assim sucessivamente até todos os nodos da rede receberem a mensagem com o pedido.

[7] Trata-se de um algoritmo iterativo para estimar por máxima verossimilhança os parâmetros de um modelo estatístico.

mistura com distribuições Gaussianas.

Dado tratar-se de uma estratégia que modela os dados amostrados na rede como uma combinação de funções probabilísticas gaussianas, não é possível recolher mínimos e máximos da rede. Do mesmo modo não é também possível a detecção de *outliers*. Para ultrapassar esta questão, os autores propõem a possibilidade de múltiplas rondas de agregação em que, após a primeira ronda, o nodo receptor envia para todos os nodos os parâmetros da agregação e daí em diante, cada nodo determina qual a probabilidade de o seu valor estar representado na distribuição. Caso essa probabilidade seja baixa, a menos de um limiar, então o nodo envia esse valor para o receptor. Assim, o nodo receptor terá recebido uma colecção de valores extremos numa ronda subsequente.

Os autores relatam que para valores baixos de componentes do modelo, a aproximação da estimativa não é precisa e concluem que será necessária uma fase de "treino" da rede para se ter uma noção geral da distribuição dos valores na rede.

O trabalho compara-se com o algoritmo q-digest [SBA04], ultrapassando-o em desempenho, no custo de comunicação, precisão, e desempenho em presença de falhas nas ligações entre nodos.

Distributed Data Classification in Sensor Networks; Ittay Ayal et. al. Neste trabalho [EKR10], os autores propõem um algoritmo de agregação em redes de larga escala cuja função é a classificação de dados em clusters multi-dimensionais. O contributo mais significativo deste trabalho prende-se com a abordagem abstracta à classificação de dados em redes distribuídas. Os dados distribuídos são modelados como colecções de sumários e conjuntos de classificações. A determinação da classificação é feita com recurso a operações de sumarização dos dados, particionamento e fusão. O algoritmo funciona do seguinte modo:

Sumarização No instante inicial, cada nodo cria uma classificação com uma colecção correspondente ao sumário do seu valor amostrado. A essa colecção associa o peso 1.

Particionamento No evento de envio, a classificação é particionada em duas novas: ambas com o mesmo sumário e com peso de aproximadamente metade (mas cuja soma é 1, garantindo a persistência de massa em toda a rede). O nodo retém uma das classificações e envia a remanescente (com peso complementar)

para o nodo escolhido. A transferência pode dar-se por *push*, *pull* ou *push-pull*.

Fusão e Classificação Assim que uma colecção é entregue a um nodo, este funde-a com a sua classificação actual. De seguida, cria K partições desse resultado tomando-as como a sua classificação, cujo peso total é a soma dos pesos das colecções fundidas.

Os autores provam que numa rede, com qualquer estratégia de classificação, comunicação assíncrona, qualquer topologia e cujas operações respeitem os invariantes estipuladas no artigo, todos os nodos convergirão para uma estimativa de classificação igual. De notar que a rede é modelada como sendo fiável: todas as mensagens chegam inevitavelmente ao destino, não há mensagens duplicadas e nenhuma mensagem espúria é criada.

No mesmo trabalho os autores apresentam uma experiência com um modelo de mistura gaussiano, usando como estratégia de particionamento o algoritmo de *Expectation Maximization* (estratégia também usada em [JJW10]). Mostram resultados de agregação para uma rede totalmente ligada de 1000 sensores, com valores amostrados em $\mathbb{R}^2$ (localização em x e valor lido pelo sensor). Não comparam com outros trabalhos da área e mostra como é sensível à eliminação de *outliers*, a capacidade de agregação multi-dimensional do algoritmo e a sua velocidade de convergência. De notar que esta última está intimamente relacionada com a topologia totalmente ligada da rede usada na experiência.

Medians and beyond: New aggregation techniques for sensor networks (Shrivatsava et. al.) Os autores deste artigo [SBA04] apresentam o algoritmo *Quantile Digest* ou *q-digest*: uma técnica de agregação de dados distribuídos, caracterizando-os como uma distribuição estatística. Os dados que compõem os agregados são comprimidos por forma a manter os custos de comunicação baixos. É sensível à perdas de mensagens e não é robusto, dado criar uma rede *overlay* em *spanning tree*, cuja fragilidade decorre do efeito que uma falha em apenas um nodo produz na agregação. Não admite também a duplicação de mensagens nem a existência de ciclos na rede (daí a imposição do *overlay* em árvore).

A agregação é feita a partir de um nodo monitor (raiz da árvore da rede), que envia para os restantes nodos uma mensagem de "início de agregação" e esta processa-se

no sentido das folhas da árvore para a raiz. O algoritmo apresentado oferece garantias probabilísticas quanto ao erro da estimativa e custo de mensagens máximos.

A compressão dos dados pressupõe o conhecimento à priori da gama σ de valores da propriedade a agregar (limitada a valores naturais superiores a 1). Pressupõe também o conhecimento do número de valores amostrados na rede n, i.e., o número de nodos da rede. O agregado é construído com uma *spanning tree* de profundidade fixa $\log \sigma$, cujas folhas guardam contadores para os valores amostrados. Os nodos da árvore podem ser vistos como intervalos do histograma. Os níveis superiores desta árvore resultam no somatório dos contadores dos seus nodos filho.

A árvore que representa os valores agregados (quer resulte da adição de um novo valor simples ou da fusão com outra árvore de agregação) passa por uma operação de compressão sempre que as regras seguintes não sejam satisfeitas:

- A contagem num dado nodo v da árvore do agregado não deverá ser superior a uma fracção do número de amostras n na rede, $count(v) \leq \lceil n/k \rceil$, com k um factor de compressão.
- O somatório das contagens de um nodo com as contagens dos seus predecessores e dos seus sucessores deverá ser superior à fracção do número de amostras na rede por um factor de compressão, $count(v) + count(v_{parents}) + count(v_{siblings}) > \lceil n/k \rceil$.

São excepção a estas regra os nodos folha e a raiz da árvore do agregado.

A primeira regra dita que nenhum nodo deverá ter uma contagem elevada. Esta propriedade "força" a distribuição uniforme das contagens pelos nodos da árvore. A segunda regra dita que não deverá haver nodos e respectivos filhos com contagens demasiado baixas. Daqui decorre que no caso de haver dois nodos filhos com contadores de valor baixo, o seu valor é somado e passa a estar representado no nodo pai.

Quando dois nodos da rede trocam de árvores de agregação, estas são combinadas numa operação de união de ambas as árvores. São de seguida aplicadas as regras de compressão da árvore de agregação.

Sobre os agregados é possível aplicar operações de consulta de *ranking*, de valores por intervalo e de valores de consenso. Para todas estas consultas, os autores provam o seu erro máximo.

Neste artigo, os autores provam que o tamanho máximo da árvore de agregação é inferior a $3k$, com k o factor de compressão. Provam também que o erro máximo de contagem num dado nodo da árvore de agregação é de $\frac{\log \sigma}{k} n$. Provam ainda que a união de duas árvores de agregação não altera o erro máximo de contagem na árvore resultante da operação.

Os autores apresentam o desempenho deste esquema a partir de simulações usando dados aleatórios e dados correlacionados espacialmente (no caso, dados de altimetria), com redes de diferentes tamanhos. Nada é dito sobre o processo de criação da rede em árvore. São apresentados dados sobre a precisão da agregação tomando por referência o histograma que resultaria da não compressão dos dados, i. e. a aproximação *naïve* em que o nodo raiz receberia todos os dados de todos os restantes nodos da árvore. Apresentam também resultados sobre o custo de transmissão de mensagens - que para um erro admissível de até 2%, baliza o tamanho das mensagens em 400 bytes. De uma maneira geral, para dados correlacionados espacialmente, o número de mensagens trocadas é inferior ao cenário com dados aleatórios.

Adam2 Em [SNSP10] é usada uma técnica de determinação de funções distribuição cumulativa (CDF) fazendo o particionamento da gama de valores da propriedade a monitorizar em pontos de amostragem. Os nodos contribuem para cada um desse pontos, caso o seu valor amostrado seja igual ou inferior ao ponto de amostragem. Esse algoritmo usa *Push-Pull gossiping* [KDG03] para estimar a fracção de nodos associada a cada ponto de amostragem. Neste algoritmo, a rede arranca com um conjunto de instâncias com pontos de amostragem equi-espaçados e com mínimo e máximo da propriedade pré-definidos. Para acompanhar alterações dos valores amostrados, a estratégia usada exige que o protocolo seja reiniciado periodicamente. Para além disso, os pressupostos desta abordagem não admitem perda de mensagens e é também sensível à duplicação de mensagens.

Probabilisticamente, novas instâncias são criadas em determinadas rondas. O conjunto de pontos de amostragem dessas novas instâncias de agregação é determinado com base na instância anterior, de forma a melhorar a precisão da estimativa. Esta partição refinada dá origem a uma distribuição cumulativa cujos pontos de amostragem tendem a ser equi-frequentes – o conjunto dos pontos de interpolação é distribuído tal que a frequência para todos eles é equi-espaçada. Esta abordagem

permite caracterizar a distribuição com mais precisão nos intervalos onde a variação de frequência é maior. O algoritmo usa três estratégias de cálculo de pontos de interpolação equi-frequentes: *Hcut*, *MinMax* e *Lcut*.

Hcut A primeira estratégia serve-se dos pontos de interpolação de uma instância anterior e calcula os novos pontos de interpolação como sendo aqueles cuja projecção no eixo do atributo correspondem a intervalos equi-espaçados no eixo das frequências.

MinMax A estratégia de *MinMax* procura numa instância anterior os dois pontos de amostragem cuja diferença de frequências seja maior para todo o conjunto de pontos, e determina um novo ponto de amostragem para a nova instância cujo valor é a interpolação do par de pontos que satisfaz a condição de maior diferença. Então, do conjunto de pontos de amostragem inicial é removido o ponto cuja diferença de frequências com um ponto adjacente seja a menor para o conjunto de pontos de amostragem.

Lcut Esta estratégia toma a estimativa de uma dada instância e, dos segmentos de recta que constituem a função cumulativa discreta, calcula os novos pontos de interpolação dividindo a poli-linha resultante em segmentos de tamanho igual.

2.4 Outros algoritmos de agregação

Uma medida com interesse numa rede distribuída de larga escala, cujo significado qualitativo está algures entre uma métrica escalar e a caracterização da distribuição estatística subjacente, são as estatísticas ordinais (*rankings*).

FILA [WJXW07] Esta estratégia usa o que os autores denominam por *filtros aritméticos* para excluir a progressão na árvore de agregação de valores irrelevantes para a lista ordenada. O algoritmo monitoriza dinamicamente alterações nos valores lidos pelos nodos, actualizando a lista ordenada de acordo com os filtros disponibilizados aos nodos da rede pelo nodo monitor. Este protocolo é inspirado no *TAG* (ver Secção 2.2.1): após a construção de uma topologia em árvore, os nodos agregam os dados no sentido do nodo monitor; O nodo monitor ordena então os valores e envia para os nodos da árvore um tuplo (inf, sup) – o filtro com o limite inferior e

superior. A partir desse momento, um nodo apenas reporta uma alteração do seu valor amostrado caso este esteja fora do limite imposto pelo seu filtro. Os autores propõem diferentes estratégias de filtragem.

A abordagem de filtragem mais simples, denominada *filtro uniforme*, determina que assim que o nodo raiz está na posse da lista ordenada, calcula para cada um dos nodos que contribuíram para a lista o filtro cujos limites são definidos pela Equação 2.1. Dados os valores ordinais $\langle v_1, v_2, \ldots, v_k \rangle$, com k o número de itens da lista de valores, o filtro para os nodos cujo valor é máximo, é definido como $v_1 - 1, +\infty$. Os nodos cujo valor amostrado não está incluído na lista, dado possuírem valores inferiores, deverão receber um filtro $(-\infty, v_k - 1)$. Os limites inf e sup a distribuir pelos restantes nodos n que contribuíram para a lista, é dado por:

$$\forall_n, sup_{n+1} = inf_n = \frac{v_n + v_{n+1}}{2} \tag{2.1}$$

Se um nodo amostrar um valor que esteja fora do intervalo definido para si, este envia-o para o nodo pai. O nodo pai propaga este valor caso a condição se mantenha para o seu filtro. Na eventualidade de o valor chegar ao nodo monitor, este recalcula a ordem dos valores e estima novos filtros a distribuir pelos nodos da árvore.

EXTOK No trabalho descrito em [MNN11], os autores propõem um algoritmo de agregação de listas ordenadas de valores em redes de sensores sem fios, derivado do trabalho publicado em [WJXW07]. O algoritmo proposto tem por condição uma topologia de rede particular – em árvore DST (*Dominating Set Tree*). Os autores destacam a importância da utilização de tal topologia e provam a correcção do algoritmo proposto. Uma DST constrói-se conectando os nodos dominantes[8] de um grafo, resultando na minimização do número de mensagens transmitidas. Neste trabalho é também demonstrado que, ao contrário do que sucede no algoritmo FILA [WJXW07], o EXTOK permite processar as *queries* de forma mais eficiente e em simultâneo agregar os dados com mais precisão.

O algoritmo funciona da seguinte forma:

- Numa primeira ronda todos os nodos reportam ao nodo raiz/ monitor os seus

[8] O conjunto dos vértices dominantes de um grafo $G = (V, E)$ é um subconjunto D dos vértices V tal que os vértices que não pertencem a D se ligam por pelo menos uma aresta a um elemento de D.

valores amostrados e este determina as estatísticas ordinais (*top-k values* ou *ranking*). beginitemize

- O nodo monitor calcula um limiar τ, que é o mínimo dos valores recebidos pelos nodos.
- Após esta operação, os nodos entram em um de dois estados: "modo temporal" ou "modo filtragem". Um nodos entra em "modo temporal" caso tenha dado origem a algum dos valores da estatística ordinal. Os nodos neste estado reportam ao nodo monitor em todas as rondas subsequentes. Caso um nodo não tenha contribuído com o seu valor amostrado para a estatística ordinal, este entra em "modo filtragem". Neste estado, os nodos reportam ao nodo monitor apenas quando observam alterações no seu valor amostrado que sejam iguais ou superiores a τ.
- Todas as rondas subsequentes são compostas por três ciclos:
  1. Num primeiro ciclo, os nodos determinam as alterações a reportar de acordo com o seu estado. Depois de receber os dados dos nodos, o nodo monitor valida os valores actuais da estatística e se necessário invoca um procedimento de validação.
  2. No segundo ciclo do algoritmo, alguns nodos em "modo filtro" poderão responder ao procedimento de validação invocado pelo nodo monitor. No final deste ciclo, o nodo monitor determina a resposta correcta à *query*.
  3. No ciclo final, o nodo monitor ajusta o valor de τ com base na nova estatística ordinal e caso este tenha mudado, reporta o novo valor de τ aos restantes nodos da rede.

2.5 Histogramas e a sua construção

A caracterização de distribuições estatísticas usando histogramas oferece vantagens na análise dos valores que uma propriedade pode tomar. Trata-se de uma forma compacta de representar a frequência desses valores em intervalos pré-definidos. As questões que rodeiam a construção de histogramas centram-se na determinação das fronteiras dos seus intervalos e no número de intervalos que o compõem. Há literatura cientifica [PHIS96] que sumariza as questões relacionadas com a determinação

das fronteiras dos intervalos no domínio da agregação de dados em sistemas de bases de dados não distribuídas.

O número de intervalos de um histograma deve ser determinado tendo em conta o propósito final do histograma. Assim, por exemplo para a finalidade de visualização, existem heurísticas como a regra de *Sturges* que estima o número de intervalos k de um conjunto de amostras N como $k = \lceil 1 + \log_2 \mid N \mid \rceil$, a regra de *Scott* que estima $k = \lceil (2 \mid N \mid^{1/3}) \rceil$ ou ainda uma aproximação ao número de intervalos $k = \lceil \sqrt{\mid N \mid} \rceil$.

A partição dos intervalos pode classificar-se em equi-espaçada e equi-frequente.

Equi-espaçados No primeiro caso, para um conjunto de intervalos disjuntos, o seu tamanho (largura) h é igual para todos os seus elementos e pode ser calculado a partir das heurísticas que determinam o número de intervalos, considerando o valor mínimo e máximo do conjunto das amostras: para a regra de *Sturges* $h = \frac{\max_x - \min_x}{\lceil 1 + \log_2 \mid N \mid \rceil}$; para a regra de *Scott* $h = \frac{\max_x - \min_x}{2} \lceil \mid N \mid^{-1/3} \rceil$, ou ainda e simplesmente para um número arbitrário de k intervalos, $h = \frac{\max_x - \min_x}{k}$.

Equi-frequentes Os intervalos equi-frequentes são calculados por forma a que a frequência dos valores amostrados seja distribuída uniformemente entre eles. Desta forma resultam intervalos que poderão ter tamanhos diferentes mas cuja frequência (altura) é idêntica, como o caso da criação de intervalos *v-optimal*.

Num histograma *v-optimal* pretende-se a criação de intervalos cujas fronteiras são calculadas por forma a minimizar a variância cumulativa pesada dos intervalos. Posto de outra forma, a quantidade

$$\sum_{\forall j} n_j V_j \tag{2.2}$$

deve ser minimizada, com n_j o número de amostras do intervalo j e V_j a variância entre os itens que fazem parte desse mesmo intervalo.[9]

A exigência de minimização da Expressão 2.2 resulta no teste exaustivo de todas as combinações possíveis de amostras e partição de intervalos. Esta abordagem, apesar de óptima, é claramente impraticável mesmo com um número baixo de in-

[9] O estimador variância não enviesado, para um intervalo j com n amostras: $V_j = \frac{1}{n-1} \sum_{\forall x} (x - \overline{X}_j)^2$.

tervalos. Para além disso, exige que o conjunto de valores com o qual se pretende criar o histograma seja conhecido na totalidade à partida. Estas questões tornam a construção de histogramas equi-frequentes de forma distribuída um desafio.

Capítulo 3

Estimativa de Função de Distribuição Cumulativa

Neste capítulo é apresentado de forma detalhada o algoritmo criado para a determinação da estimativa da função distribuição cumulativa numa rede de larga escala. Como ponto de partida, faz-se uma exposição intuitiva do algoritmo ilustrando com diagramas e com uma explicação sucinta o seu funcionamento. A sua formalização é apresentada *à posteriori*.

3.1 Apresentação breve

Para tornar claro o funcionamento do algoritmo, abordamos em primeiro lugar a forma como num dado nodo a determinação de intervalos equi-espaçados é feita. Assumindo que esse nodo, após algumas trocas de mensagens com nodos vizinhos, conhece um mínimo min e um máximo max amostrados no conjunto dos nodos, o cálculo do espaçamento processa-se da seguinte forma:

- Dado que o número de intervalos k é fixo para todos os nodos da rede, o tamanho t de cada intervalo l (adiante denominado por *label*) é dado pela expressão $t = \frac{max-min}{k}$
- Os k intervalos tomam o valor que resulta da adição do espaçamento calculado com a ordem do intervalo, tendo como início o valor do mínimo observado: $l = min + k * t$

Este processo é ilustrado na Figura 3.1. Em seguida o nodo procede ao cálculo do posicionamento relativo do seu valor amostrado com base nos *labels* recém calculados. Este passo carece de uma explicação intuitiva. Cada nodo contribuirá com um valor unitário para os intervalos aos quais o seu valor amostrado pertença. Desta forma, e com base no cálculo de uma média aplicada intervalo a intervalo, será possível estimar o número de valores amostrados que ocorrem nesse intervalo para o universo da rede e, dessa forma, estimar o número de ocorrências acumuladas.

Para calcular a posição relativa do seu valor amostrado, o nodo compara-o com cada um dos intervalos calculados. Se o seu valor amostrado for igual ou superior ao do intervalo, o nodo coloca na posição respectiva o valor *um*. Caso o seu valor amostra seja inferior ao do intervalo, o nodo coloca nessa posição o valor *zero*. Desta forma, o vector de posições relativas serve como um contador de valores amostrados. Assim, e por meio de um método de cálculo de médias, um nodo pode estimar a proporção de nodos que contribuem com valores amostrados para aquele intervalo. Todo este procedimento está ilustrado na Figura 3.2.

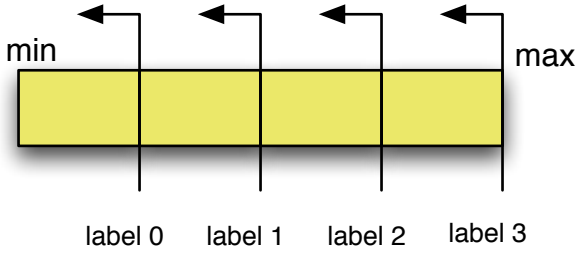

Figura 3.1: Exemplo da formação de um intervalo num nodo.

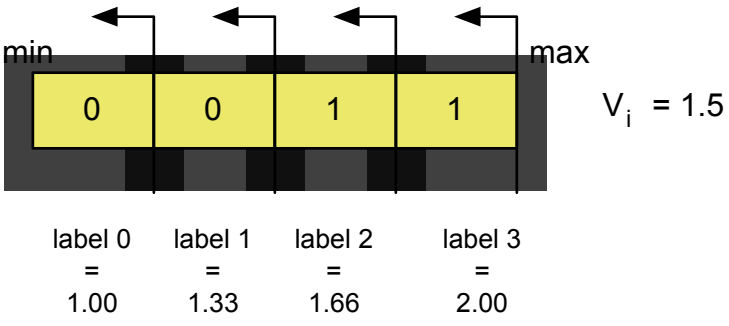

Figura 3.2: Exemplo da formação de um vector com posicionamento relativo para um valor amostrado exemplificativo de 1.5.

Agora, vejamos ainda de forma intuitiva o comportamento do algoritmo em rede, tomando por base uma uma rede constituída por três nodos A, B e C. Esta rede apresenta-se com uma topologia em anel e com valores amostrados $1, 2, 3$, tal como é apresentado na Figura 3.3, e representados por baixo da identificação de cada nodo. Admitamos que cada nodo irá construir uma função cumulativa com quatro pontos de amostragem (ilustrado na forma de 4 rectângulos adjacentes).

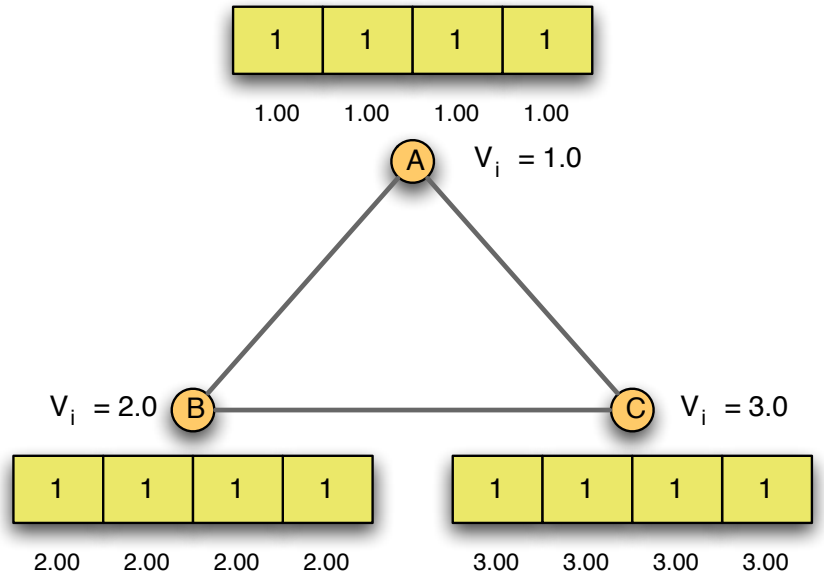

Figura 3.3: Nodos com o valor amostrado, vector da CDF com os respectivos valores iniciais e representação da topologia.

Na ronda inicial, os nodos apenas têm conhecimento do seu valor inicial. Com base nesse conhecimento, o nodo estima que a função cumulativa é constituída apenas pelo seu valor amostrado (todos os pontos de amostragem têm valor idêntico ao valor amostrado, tal como está ilustrado pelos valores imediatamente abaixo dos quatro rectângulos adjacentes). Simultaneamente, cada nodo calcula se o seu valor amostrado é igual ou inferior a cada ponto de amostragem: Como todos os pontos de amostragem são de valor igual ao valor amostrado, todos os nodos preenchem os quatro rectângulos adjacentes com o valor *um*, cujo significado é "para este ponto de amostragem o meu valor amostrado é igual ou inferior".

Admitindo que aleatoriamente um dos nodos troca com outro nodo a sua estimativa – sem perda de generalidade, esta troca está ilustrada como acontecendo em simultâneo entre os nodos A e B na Figura 3.4. No entanto o comportamento do

algoritmo é assimétrico.

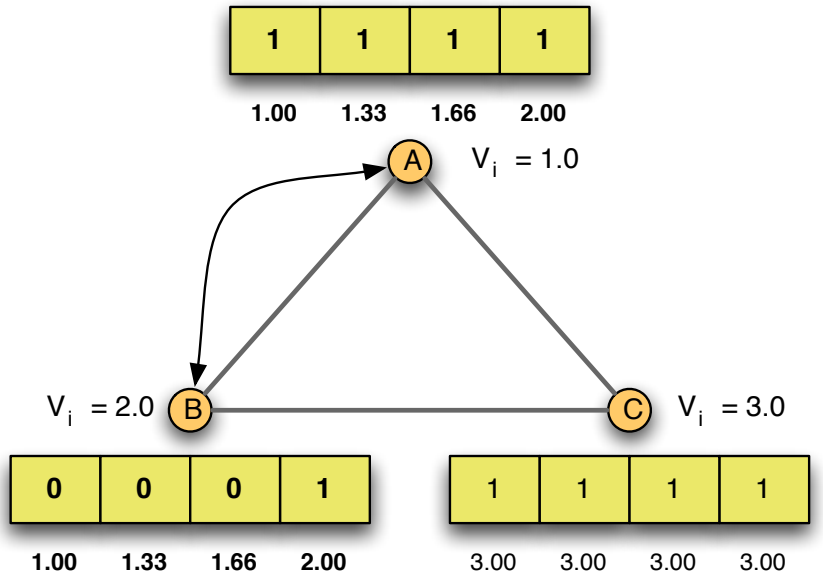

Figura 3.4: Mensagem trocada entre os nodos A e B e respectivas novas estimativas de CDF.

O nodo A envia para o nodo B a sua estimativa. Com base na informação recebida, o nodo B determina que o valor observado por A corresponde a um novo mínimo tendo em conta os seus valores actuais, e recalcula os seus pontos de amostragem com base neste novo conhecimento: quatro pontos cujo mínimo é 1 e o máximo é 2 e cujos valores intermédios são 1.33 e 1.66. De seguida, o nodo B determina qual a posição do seu valor amostrado com base neste novo conjunto de pontos de amostragem: *dois* é menor ou igual 1? Não. Menor ou igual que 1.33? Não. Menor ou igual que 1.66? Não. Menor ou igual que 2? Sim. Daqui resulta a sua nova estimativa de CDF, com os novos valores de pontos de amostragem e novos valores de posicionamento relativo do seu valor amostrado.

O nodo A procede de forma idêntica.

Num outro momento subsequente, os nodos A e C procedem também a uma troca de estimativas (ver Figura 3.5). O nodo C recalcula os seus pontos de amostragem com base no conhecimento recebido de A. Logo, os seus pontos de amostragem dividir-se-ão entre 1 e 3, com valores intermédios de 1.66 e 2.33. De seguida, o nodo C determina qual a posição do seu valor amostrado com base neste novo conjunto de pontos de amostragem. Daqui resulta a sua nova estimativa de CDF, com os

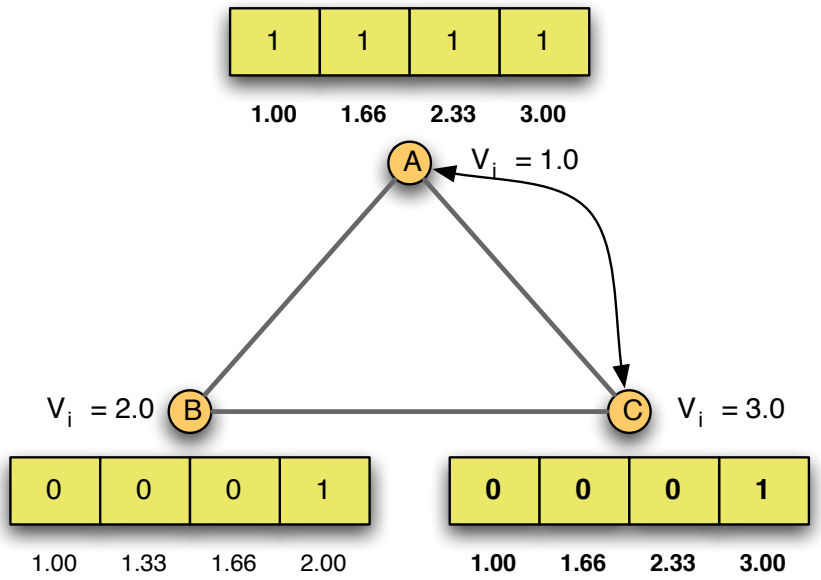

Figura 3.5: Mensagem trocada entre os nodos A e C (após a operação da Figura 3.4) e respectivas novas estimativas de CDF.

novos valores de pontos de amostragem e novos valores de posicionamento relativo do seu valor amostrado.

O nodo A procede de forma idêntica: os seus pontos de amostragem serão idênticos aos de C mas os seus valores de posicionamento relativo serão todos iguais a 1 dado que o seu valor inicial é sempre inferior ou igual aos pontos de amostragem calculados.

Num outro momento, os nodos B e C procedem a uma troca de mensagens (tal como é ilustrado na Figura 3.6). O nodo B recalcula os seus pontos de amostragem de acordo com a informação recebida de C, passando a ter intervalos idênticos a C. A posição relativa do seu valor amostrado (2) é então calculada com base nos valores dos pontos de amostragem e resultam no vector 0011. A partir desse ponto, qualquer troca de mensagens subsequente entre os nodos da rede não alterará os pontos de amostragem nem os valores relativos de cada nodo.

Nestes três passos apresentou-se de forma intuitiva o funcionamento de parte do algoritmo de determinação de uma CDF. No entanto falta um passo fundamental para a obtenção de uma CDF *de facto*, e que é dado por um algoritmo subjacente: o algoritmo de *averaging*. Por meio de uma técnica de *averaging* é possível determinar qual a fracção de valores dos nodos se encontra em cada um dos pontos de

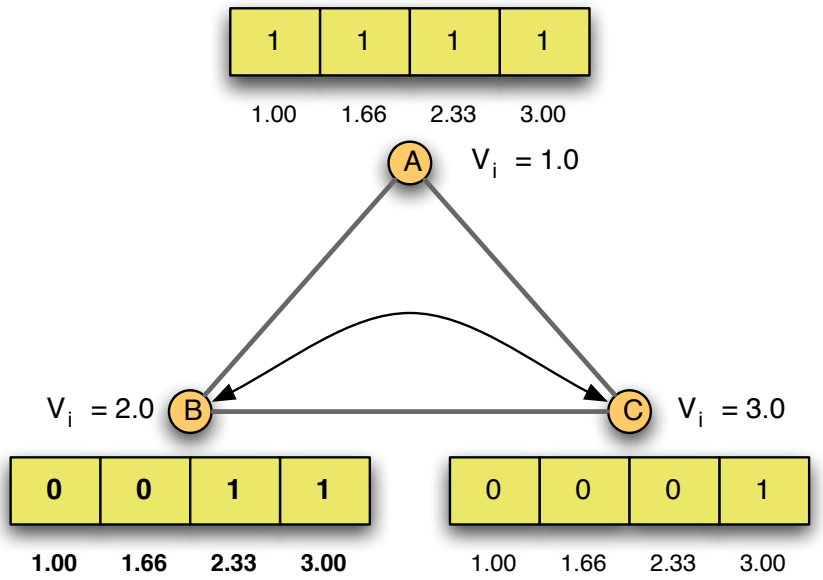

Figura 3.6: Mensagem trocada entre os nodos B e C (após a operação da Figura 3.5) e respectivas novas estimativas de CDF.

amostragem. Estes algoritmos estimam de forma distribuída a média de um valor amostrado numa rede de nodos. No caso particular, é aplicada a cada um dos valores associados aos pontos de amostragem. Pelas suas características intrínsecas, tal como estão descritas na Secção 2.2.2, utilizou-se neste trabalho o algoritmo de *Flow Update*.

Veremos agora o que resulta da aplicação de um algoritmo de *averaging* na estimativa da distribuição do valores cumulativos presentes na rede:

- Tomando a primeira posição dos quatro rectângulos adjacentes em todos os nodos temos (relativo ao ponto de amostragem 1): $A:1, B:0, C:0$. O nodo pode então determinar que há $\frac{1+0+0}{3} = 1/3 = 0.33$ dos nodos com valor igual ou inferior a 1.

- Tomando a segunda posição, relativa ao ponto de amostragem 1.66: $A:1, B:0, C:0$. O nodo determina que há 1/3 dos nodos com valor igual ou inferior a 1.66 o que significa, dado tratar-se de uma função cumulativa, que não há valores na rede para o intervalo aberto entre 1 e 1.66.

- No terceiro ponto de amostragem (2.66) temos os valores $A:1, B:1, C:0$. O valor cumulativo para este ponto é então de 2/3 Significando que 2/3 dos

nodos possuem valor igual ou inferior a 2.66.

- Tratando-se de uma função cumulativa, verifica-se para o último ponto de amostragem (3) que a fracção de nodos com valor igual ou inferior a 3 é de 100%: $A:1, B:1, C:1$.

Após convergência das $k = 4$ instâncias de *averaging*, a representação gráfica da CDF em cada nodo teria um aspecto idêntico ao mostrado na Figura 3.7.

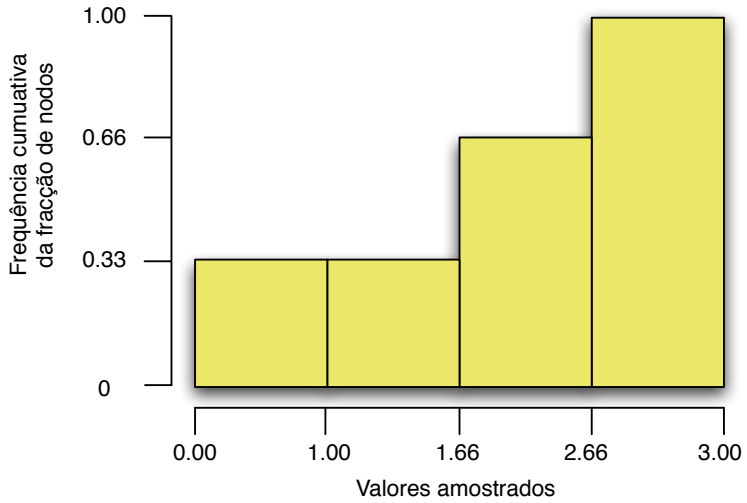

Figura 3.7: Resultado da CDF após convergência da estimativa.

Sumarizando, o algoritmo para a determinação da função distribuição cumulativa tira partido de um algoritmo distribuído de cálculo de média para aferir a fracção de nodos cujo valor está no limiar ou abaixo de um conjunto de pontos de amostragem, pontos estes calculados a partir da colecção do máximo e mínimo presente na rede. Note-se que não houve necessidade de contar o número de nodos na rede.

3.2 Formalização

Uma Função de Distribuição Cumulativa (ou CDF - Cumulative Distribution Function) define-se como a probabilidade de uma variável aleatória X com uma dada função de distribuição, ocorrer num valor igual ou inferior a x, $F(x) = P(X \leq x)$.

Dado que se pretende a estimativa de uma CDF discreta, será contada a proporção de valores da propriedade em k intervalos. Assim, a CDF é discretizada

num conjunto de k intervalos com fronteiras, ou pontos de amostragem l, tais que: $CDF = \{P(x \leq l_0), P(x \leq l_1), \ldots, P(x \leq l_k)\}$, com l_0 o valor mínimo observado e l_k o valor máximo observado. Denominamos os l_k pontos de amostragem como *labels* e estes identificam o intervalo. Em cada intervalo irá ser estimada a proporção do atributo que verifica a condição de o seu valor amostrado V_i ser igual ou inferior aos *labels*. O tamanho de todos os intervalos da CDF é obtido por $(max - min)/k$.

Considera-se que cada nodo da rede pretende estimar uma CDF de um dado atributo cujos valores pertencem a $\mathbb{R}$, sem qualquer conhecimento *à priori* sobre a gama do atributo a ser medido.

Para uma dado intervalo l da CDF, a sua estimativa de acumulação e é resultado da média dos nodos N cujo valor amostrado V_i satisfaz a condição de ser igual ou inferior ao *label* l:

$$\forall i \in N, \forall l \in Labels : e = \frac{\sum_i \langle 1 \mid V_i \leq l \rangle}{\mid N \mid} \tag{3.1}$$

O cálculo da média para cada um dos intervalos é feito de forma dinâmica com *Flow Update* [JBA09]. Trata-se de um protocolo distribuído de *averaging* baseado no princípio de distribuição de massa usado noutros algoritmos [CPX06] [KDG03].

Com o *Flow Update*, a estimativa para um dado intervalo da CDF resulta da diferença entre o valor do posicionamento relativo do valor amostrado no nodo i para esse intervalo e o somatório de fluxos f dos nodos vizinhos j.

O modelo de rede base é o de um grafo D conectado e não orientado.

3.3 Algoritmo para a determinação de CDF

O algoritmo distribuído para a determinação da CDF tem por variáveis de entrada o valor amostrado V_i, o número de *labels* k e o conjunto de vizinhos do nodo i, $\mathcal{D}_i$, manipulado por um detector de faltas (linhas 2–4 do Algoritmo 2).

Como variáveis de estado, o algoritmo possui o vector de posicionamento relativo $\vec{v_i}$, cujo valor inicial é 1 dado o nodo apenas conhecer o seu valor amostrado. Possui também um vector de *labels* $\vec{l_i}$ com valor V_i. O estado inicial do vector de posicionamento relativo verifica a condição $P(V_i \leq l_0)$.

Algoritmo 2: Algoritmo para a estimativa de uma CDF com base em *Flow Update*.

1 **inputs:**
2 V_i, valor amostrado
3 $\mathcal{D}_i$, conjunto de vizinhos, dado por um detector de falhas
4 k, número máximo de *labels*
5 **variáveis de estado:**
6 fluxos: inicialmente, $F_i = \{\}$
7 vector de posicionamento relativo: inicialmente, $\vec{v_i} = [1]$
8 vector de *labels*: inicialmente, $\vec{l_i} = [V_i]$
9 valor máximo: inicialmente, $max_i = V_i$
10 valor mínimo: inicialmente, $min_i = V_i$
11 **função geradora de mensagens:**
12 $\text{msg}_i(F_i, \vec{v_i}, \vec{l_i}, max_i, min_i, j) = (i, \vec{l_i}, max_i, min_i, \vec{f}, \text{est}(\vec{v_i}, F_i))$;
13 **with** $\vec{f} = \begin{cases} \overrightarrow{F_i(j)} & \text{if } (j, \_) \in F_i \\ \vec{0} & \text{caso contrário} \end{cases}$
14 **função de transição de estado:**
15 $\text{trans}_i(F_i, \vec{v_i}, \vec{l_i}, max_i, min_i, M_i) = (F_i', \vec{v_i}', \vec{l_i}', max_i', min_i')$
16 **with**
17 $max_i' = \max(\{max \mid (\_, \_, max, \_, \_, \_) \in M_i\} \cup max_i)$
18 $min_i' = \min(\{min \mid (\_, \_, \_, min, \_, \_) \in M_i\} \cup min_i)$
19 $\vec{l_i}' = labelsEquiEspacados(max_i', min_i')$
20 $\vec{v_i}' = vectorPosicionamento(\vec{l_i}')$
21 $\vec{f}' = \{j \mapsto adjust(\vec{f}, \vec{l}, \vec{l_i}') \mid j \in \mathcal{D}_i \wedge (j, \vec{l}, \_, \_, \vec{f}, \_) \in M_i)\}$
22 $\vec{e}' = \{j \mapsto adjust(\vec{e}, \vec{l}, \vec{l_i}') \mid j \in \mathcal{D}_i \wedge (j, \vec{l}, \_, \_, \_, \vec{e}) \in M_i)\}$
23 **for** $n = 1$ **to** k **do**
`/* invoca a transição de estado do algoritmo FU para todos os elementos */`
24 $F_i'[n] = flowUpdate(\vec{v_i}'[n])$

Tanto o mínimo min_i como o máximo max_i locais possuem o mesmo valor de V_i no estado inicial (linhas 6–10 do Algoritmo 2).

As mensagens transmitidas entre nodos são compostas por um tuplo (identificador do nodo origem, vector de *labels* $\vec{l}$, máximo local, mínimo local, vector de

Algoritmo 3: Criação de *labels* equi-espaçados.

```
1 labelsEquiEspacados(max, min):
2     if max = min then
3         return [max]
4     else
5         Δ ← (max − min)/k
6         l⃗[k] ← max
7         for n = k − 1 to 1 do
8             l⃗[n] ← l⃗[n + 1] − Δ
9         return l⃗
```

fluxos $\vec{f}$, vector de estimativas) (linha 12). De início, dado ainda não ter havido troca de mensagens, os fluxos enviados são nulos (linha 13).

O algoritmo processa as mensagens que lhe chegam fazendo alterar as suas variáveis de estado em função do seu conteúdo e em função do seu estado actual. Em consequência, cria novas mensagens a emitir e que resultam desse novo estado.

É determinado um novo máximo e mínimo locais, cujo valor é obtido do conjunto de todos os máximos e mínimos recebidos pelos nodos vizinhos (linhas 17–18 do Algoritmo 2).

De seguida o nodo determina os novos *labels* de forma equi-espaçada com auxílio da função *labelseEquiEspacados*, descrita no Algoritmo 3.

Com o novo conjunto de *labels*, o algoritmo determina qual o seu posicionamento relativo tendo por base o seu valor amostrado V_i. Esta operação é feita invocando a função *vectorPosicionamento*, de acordo com o Algoritmo 4.

Antes de proceder ao cálculo, para cada elemento dos vectores, das estimativas e fluxos, estes precisam de ser ajustados ao novo referencial de *labels* $\vec{l_i}'$. Esta operação é necessária dado que tanto os fluxos como as estimativas recebidas dos nodos vizinhos estão referenciadas ao conjunto de *labels* locais a esses nodos. Ora, uma vez que o nodo cria um conjunto de novos *labels*, será necessário transformar os vectores recebidos para o referencial local. Este procedimento está descrito no Algoritmo 5.

Este algoritmo tem como parâmetros um vector de *labels* $\vec{b}$ associado a um vector

Algoritmo 4: Algoritmo para o cálculo das posições no vector de posicionamento.

```
1 vectorPosicionamento(l⃗):
2     for n = 1 to k do
3         if V_i ≤ l⃗[n] then
4             v⃗[n] ← 1
5         else
6             v⃗[n] ← 0
7     return v⃗
```

Algoritmo 5: Ajuste dos vectores de *labels* $\vec{a}$ e valores associados $\vec{b}$ a um vector de *labels* $\vec{l}$.

```
1 adjust(a⃗, b⃗, l⃗):
2     foreach l ∈ l⃗ do
3         c = max({b ∈ b⃗ | b ≤ l})
4         r⃗[l] = a⃗[c]
5     return r⃗
```

de valores $\vec{a}$ e que se pretende transformar para uma base de referência de *labels* $\vec{l}$. Do procedimento resulta um vector de valores $\vec{r}$ cujos elementos provêm de $\vec{a}$ e cujo posicionamento é determinado pela proximidade entre os elementos dos *labels* de $\vec{b}$ com os *labels* de $\vec{l}$. A posição dos elementos em $\vec{r}$ resulta do elemento mais próximo em $\vec{b}$ cujo valor é igual ou inferior a l. (linhas 3–4 do Algoritmo 5).

Após estas operações de ajuste de referência, os valores de fluxo e demais vectores serão usados para o cálculo da nova estimativa e restantes variáveis de estado por invocação do *Flow Update* (linhas 23–24 do Algoritmo 2).

3.4 Algoritmo para controlo de quiesciência

A motivação para a implementação de um mecanismo de quiesciência, para que cada nodo pare de transmitir activamente numa dada ronda, prende-se com a necessidade de reduzir o volume de informação passada entre nodos da rede, uma vez atingido um determinado grau de precisão na estimativa da CDF.

Em determinadas condições, cada nodo pode decidir localmente não enviar informação sobre a sua estimativa actual e respectivos fluxos para os nodos vizinhos, permitindo assim baixar o custo de comunicação. Esta decisão implica apenas a não transmissão de mensagens para os nodos vizinhos - o nodo continua a computar alterações ao seu estado actual com base em mensagens recebidas e no seu estado intrínseco.

A decisão local é tomada aferindo a taxa de variação entre estimativas consecutivas, dado não se ter conhecimento sobre a distribuição real. O algoritmo de quiesciência tem como parâmetros dois valores que definem os níveis de histerese (o limite de variação mínimo ε_{low} a partir da qual o nodo entra no estado quiesciente $\chi = asleep$ e o limite máximo ε_{high} a partir do qual o nodo abandona o estado quiesciente, $\chi = awake$).

A utilização de um mecanismo de histerese evita oscilações no estado de quiesciência assim que ε_{low} é ultrapassado. Para que o nodo retorne ao estado de *awake*, é necessário que a variação local ultrapasse o limiar superior ε_{high}. Qualquer dos limiares é testado apenas no caso em que entre duas estimativas consecutivas, ambos os vectores de *labels* tenham os mesmos elementos.

A taxa de variação entre estimativas para um dado *label* é dada pela Equação 3.2

$$\Delta_t(l) = \frac{P(X \leq l)_t - P(X \leq l)_{t-1}}{P(X \leq l)_t} \tag{3.2}$$

De uma CDF, numa ronda r, é considerado o intervalo cujo Δ_t é máximo, tal como expresso na Equação 3.3.

$$\Delta_{tmax} = \max_{l \in Labels} (\Delta_t(l)) \tag{3.3}$$

Caso Δ_{tmax} seja inferior ao limiar de erro mínimo ε_{low}, o nodo entra em quiesciência ($\chi = asleep$).

Neste estado, o nodo continua a monitorizar variações do seu valor amostrado e a computar novas estimativas de acordo com essa variação e de acordo com mensagens de nodos vizinhos que possa receber. Caso o resultado de um par de estimativas consecutivas calculada em quiesciência seja superior ao limiar de erro máximo estipulado ε_{high}, o nodo passa então a enviar de novo mensagens para a vizinhança.

Um nodo arranca o protocolo com estado inicial acordado, $\chi = awake$. Este comportamento é descrito pela Equação 3.4.

$$\chi \leftarrow f(\Delta_t, l, \chi) = \begin{cases} asleep & \textbf{if} \max_l(\Delta_t(l)) \leq \varepsilon_{low} \wedge \chi = awake, \\ awake & \textbf{if} \max_l(\Delta_t(l)) \geq \varepsilon_{high} \wedge \chi = asleep, \\ \chi & otherwise. \end{cases} \tag{3.4}$$

A quiesciência da rede é assim determinada por variações locais. Tal como está, o mecanismo não resolve a possibilidade de sucessivos incrementos na diferença com valor inferior a ε_{high} quando está quiesciente: uma variação lenta faria com que o nodo não acordasse.

Para resolver esta questão o algoritmo é reforçado com um mecanismo que assegura que a partir do momento em que o nodo está em estado quiesciente ($\chi = asleep$) este passa a *awake* na condição descrita na Equação 3.4, **ou** caso o valor acumulado de variações positivas desde que este está em estado adormecido, for superior a ε_{high}, tal como está especificado nas linhas 7 e 12 do Algoritmo 6, que sumariza todo o procedimento desta secção.

Algoritmo 6: Algoritmo de quiesciência.

inputs:
- ε_{low}, limiar inferior para adormecer o nodo
- ε_{high}, limiar superior para acordar o nodo
- CDF_{t-1}, CDF_t, estimativas consecutivas de valores cumulativos

variáveis de estado:
- estado de quiesciência: inicialmente, $\chi = awake$
- acumulado de pequenas variações: inicialmente, $a = 0$

função de transição de estado:
- trans$(CDF_t, CDF_{t-1}, \varepsilon_{low}, \varepsilon_{high}, a) = \chi'$
- **with**
- $\Delta_{tmax} = calculaDeltaMax(CDF_t, CDF_{t-1})$
- $a = accum(\Delta_{tmax})$
- $\chi' = (asleep \mid \Delta_{tmax} \leq \varepsilon_{low}) \vee (awake \mid (\Delta_{tmax} \vee a) \geq \varepsilon_{high})$

calculaDeltaMax(CDF_{t-1}, CDF_t)**:**
- `/* ver no corpo do texto */`

accum(Δ_{tmax})**:**
- **if** $\chi = asleep$ **then**
  - $a \leftarrow a + \Delta_{tmax}$
- **else**
  - $a \leftarrow 0$
- **return** a

Capítulo 4

Avaliação

Após apresentar a formalização da abordagem para a determinação distribuída da CDF, é mostrada neste capítulo uma síntese dos resultados de simulação obtidos a partir da implementação dos algoritmos e uma breve discussão sobre esses mesmos resultados. Começa-se por apresentar as métricas usadas neste estudo. Estas servem para aferir quantitativamente o ajuste da agregação à real distribuição dos dados.

Para a simulação foi desenvolvido um módulo de simulação em *Java* que corre sobre o motor de simulação desenvolvido no âmbito do trabalho descrito em [Jes07]. Este simulador está habilitado para a simulação de funções de agregação em redes de topologia variável e com modelos de comunicação síncrono ou assíncrono e com a possibilidade de usar diferentes estratégias de *gossiping* (escolha aleatória de um ou vários nodos vizinhos, *flooding*). Para mais informação sobre o simulador, poder-se-á consultar o trabalho descrito em [Jes07].

4.1 Simulação

São usados como critério de avaliação qualitativa do protocolo, dois erros de ajuste entre a estimativa dos nodos e a distribuição estatística dos valores iniciais atribuídos aos nodos. Os erros de ajuste são calculados em todas as r rondas. Em cada ronda, o simulador observa o valor amostrado de cada nodo e constrói, de acordo com os parâmetros de simulação, a função de distribuição cumulativa real. Desta forma, nos testes de simulação em que o valor amostrado varia (tal como na experiência apresentada no Gráfico 4.5), o erro de ajuste da CDF estimada é calculado

permanentemente em função da CDF real.

Todos os nodos da rede enviam mensagens de modo síncrono por rondas, seguindo o modelo descrito em [Lyn96]. O envio de mensagens é feito por *broadcast* para o conjunto de nodos vizinhos.

4.1.1 Medição do erro

Um dos erros de ajuste é teste estatístico Kolmogorov-Smirnov que corresponde ao intervalo (*label*) que possui a maior diferença absoluta entre as frequências da função distribuição cumulativa estatística de referência (real), com base numa visão global sobre o valor dos nodos, e a função cumulativa determinada num nodo n na ronda r (estimada) para os mesmos *labels*, tal como descrito na Equação 4.1.

$$KS_r^n = \max_{l \in Labels} \mid P(X \leq l)_r - P(\widehat{X \leq l})_r^n \mid \tag{4.1}$$

O segundo erro usado na análise é a diferença de áreas, que corresponde ao somatório da diferença entre as frequências da função cumulativa dos valores da distribuição estatística de referência (real) nos nodos e a função cumulativa determinada por *averaging* num nodo na ronda r (estimada) em que os intervalos sejam idênticos. A diferença é calculada para todos os l *labels*, com max e min os valores mínimo e máximo do conjunto dos valores da distribuição estatística real, de acordo com a Equação 4.2.

$$\Sigma\Delta_r^n = \frac{max - min}{\mid labels \mid} \sum_{l \in Labels} \mid P(X \leq k)_r - P(\widehat{X \leq k})_r^n \mid \tag{4.2}$$

As métricas de erro apresentadas incidem sobre um dado nodo e permitem aferir o ajuste da estimativa nesse nodo numa ronda em particular. Com vista a medir o desempenho da rede como um todo, foram definidos erros máximos e médios para ambas as medidas KS e $\Sigma\Delta$. O erro máximo de uma medida reporta-se ao nodo com o maior erro no conjunto de todos os nodos N da rede. O erro médio de uma medida refere-se à média dos erros em todos os nodos da rede. Assim, temos que o erro KS máximo, i.e., o nodo n com o maior desvio num dos seus intervalos distribuição estatística inicial, é dado pela Equação 4.3.

$$KS_{maxr} = \max_{n \in N}(KS_r^n) \tag{4.3}$$

O erro KS médio em todos os nodos n da rede é dado de acordo com a Equação 4.4.

$$KS_{\mu r} = \frac{1}{\mid N \mid} \sum_{n \in N} KS_r^n \tag{4.4}$$

O erro de diferença de áreas máximo, i.e., o nodo n cuja diferença de áreas entre a função cumulativa estimada e a função cumulativa efectiva é maior para todos os nodos N da rede, é dado pela Equação 4.5.

$$\Sigma\Delta_{maxr} = \max_{n \in N}(\Sigma\Delta_r^n) \tag{4.5}$$

E a medida de média de diferença de áreas para toda a rede de nodos n é dada pela Equação 4.6.

$$\Sigma\Delta_{\mu r} = \frac{1}{\mid N \mid} \sum_{n \in N} \Sigma\Delta_r^n \tag{4.6}$$

4.2 Configuração experimental e resultados

Tanto quanto nos dado conhecer, este trabalho é pioneiro na agregação de distribuições tolerante a faltas em sistemas distribuídos. O algoritmo directamente concorrente (*Adam2*) [SNSP10], é baseado no algoritmo *push-pull* levando, em caso de perda de mensagens, à convergência para uma estimativa errada. Para além disso, no artigo citado acima, os autores usam o simulador PeerSim [Jel], cuja operação de *push-pull* é simulada de forma atómica, uma assunção muito optimista tendo em consideração sistemas reais.

Para avaliar a solução proposta, foram efectuadas uma série de simulações com diferentes configurações. Os resultados de simulação correspondem à média de 30 simulações efectuadas para cada uma das condições de teste.

As condições base de simulação usam uma rede de 1000 nodos gerados a partir do modelo Erdõs-Rényi [ER60] com conectividade média 3 - cada nodo da rede possui em média três vértices de ligação de dados com nodos vizinhos. Não se requer que

os nodos da rede possuam identificadores globais.

São mostrados no Gráfico 4.1, para uma rede aleatória de 1000 nodos, os diferentes erros medidos na estimativa da CDF. Na apresentação dos restantes resultados de simulação é usada maioritariamente a métrica de erro médio de diferenças máximas KS_{μ}, pois permite apresentar uma mais clara noção da diferença máxima absoluta entre a distribuição real e a distribuição estimada.

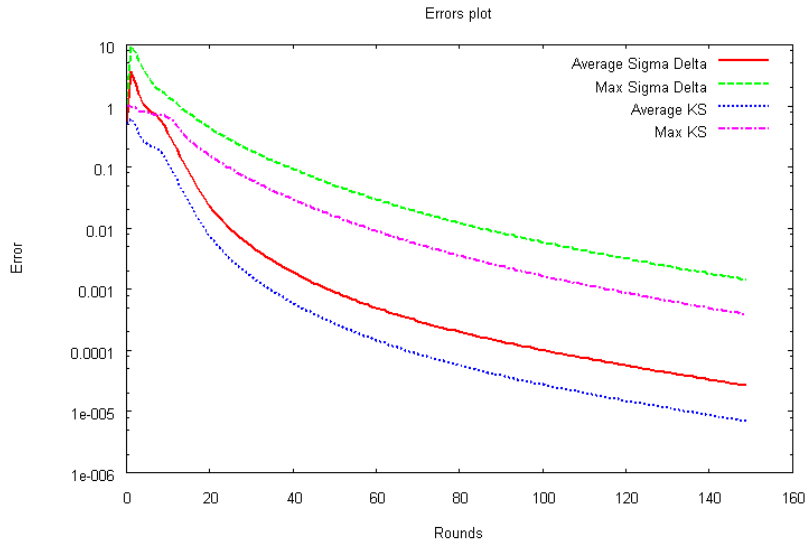

Figura 4.1: Representação dos tipos de métrica de erro usados nas simulações.

Apresentam-se de seguida os diferentes resultados de simulação com uma discussão breve sobre o seu significado.

Variação na topologia Esta simulação compara a velocidade de convergência entre uma rede geométrica aleatória e uma rede 2D. Ambas as redes possuem 1000 nodos. A CDF estimada possui 20 *labels*. A distribuição real segue os parâmetros $rNorm(10, 2)$.

A simulação de variação da topologia da rede (apresentada na Figura 4.2) mostra que a estimativa da CDF converge mais rapidamente em redes geométricas aleatórias do que em redes 2D. O resultado foi obtido para uma conectividade média de 3 em ambas as redes. Este resultado compara-se com uma rede de configuração idêntica mas usando topologia 2D (também referida por *mesh* na literatura). O maior tempo decorrido (medido em rondas) para a convergência em redes com topologia 2D com

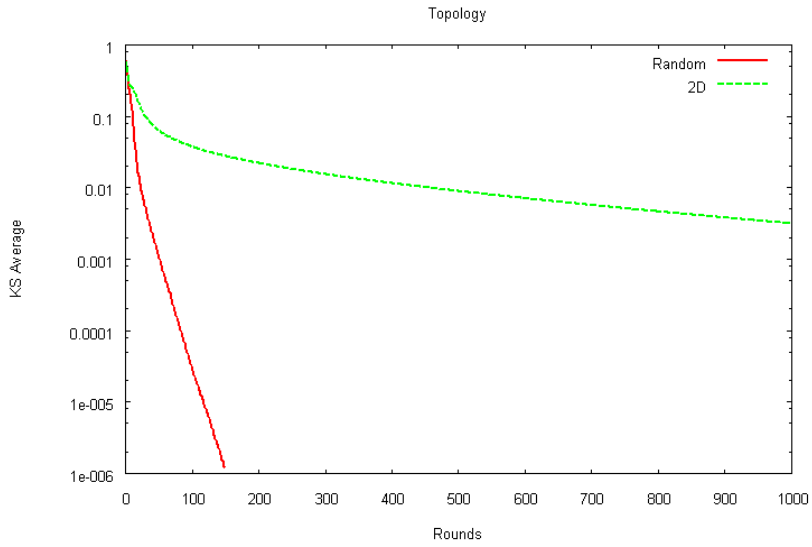

Figura 4.2: Topologia 2D vs. aleatória para uma rede de 1000 nodos.

um grau de conectividade idêntico, mostra uma relação inversa entre a densidade da rede e a velocidade de convergência. Embora não aprofundado neste estudo, estes resultados sugerem que ao invés do que intuitivamente se poderia concluir, uma maior densidade não implica necessariamente uma velocidade de convergência superior na determinação da estimativa.

Tamanho da rede Pretende-se nesta simulação avaliar o impacto do número de nós numa rede na determinação da CDF. Trata-se de uma rede geométrica aleatória com conectividade média 3. O teste foi efectuado com uma rede de tamanho 100 e outra de tamanho 1000. A distribuição real segue os parâmetros $rNorm(10, 2)$.

O erro de ajuste médio KS é tanto maior quanto maior for o tamanho da rede (apresentado em 4.3). Logo, a convergência da estimativa é mais lenta para redes com um maior número de nodos.

Este resultado vai de encontro ao que intuitivamente se esperaria: quanto maior o número de nodos, mais lenta será a convergência da estimativa. É expectável que a velocidade de convergência, possa estar também relacionada com o diâmetro[1] da rede. Dada a conectividade média das duas redes ser igual, o tamanho da rede aparenta ser o único parâmetro a influenciar a taxa de convergência diferente.

[1] O diâmetro de um grafo é o maior dos caminhos mais curtos entre os nodos do grafo.

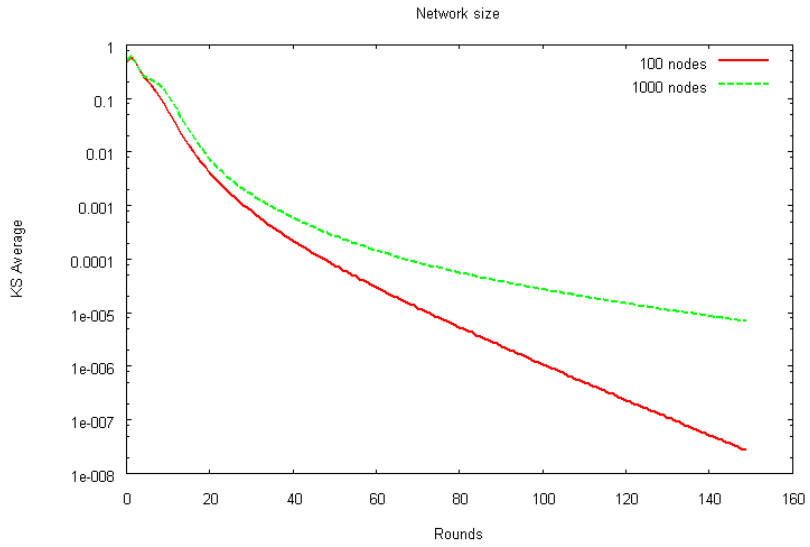

Figura 4.3: Tamanho da rede (100 e 1000 nodos).

Perda de mensagens Esta simulação foi efectuada sobre uma rede geométrica aleatória de conectividade média 3 e com 1000 nodos. A estimativa foi calculada para uma CDF de 1000 nodos. A distribuição real segue os parâmetros $rNorm(10, 2)$. Foram simuladas perdas de 5%, 10% e 20% das mensagens em cada ronda.

Na simulação de estimativa de CDF com perda de mensagens (no Gráfico 4.4) nota-se uma convergência mais rápida na ausência de perdas até cerca da ronda 50. Desse ponto em diante, o facto de se haver perda de mensagens faz com que a convergência acelere. Conclui-se então que o comportamento do algoritmo não só é resiliente à perda de mensagens como também a presença de uma taxa de 10% e pelo menos até 20% de perdas melhora a velocidade de convergência da estimativa. Este resultado é coerente com o apresentado em [JBA09]. Este comportamento emergente contraria de certa forma a intuição de que o desempenho na convergência devia degradar-se com a introdução de perda de mensagens. Os resultados de simulação sugerem em certa medida o contrário. Este fenómeno poderá entender-se se se olhar para a perda de mensagens como uma alteração momentânea na topologia da rede (a perda de uma mensagem equivalerá à extinção momentânea de um vértice da rede).

A distribuição estatística desta alteração (no caso desta simulação, a perda de mensagens afecta uniformemente todos os nodos com uma perda média de 10%),

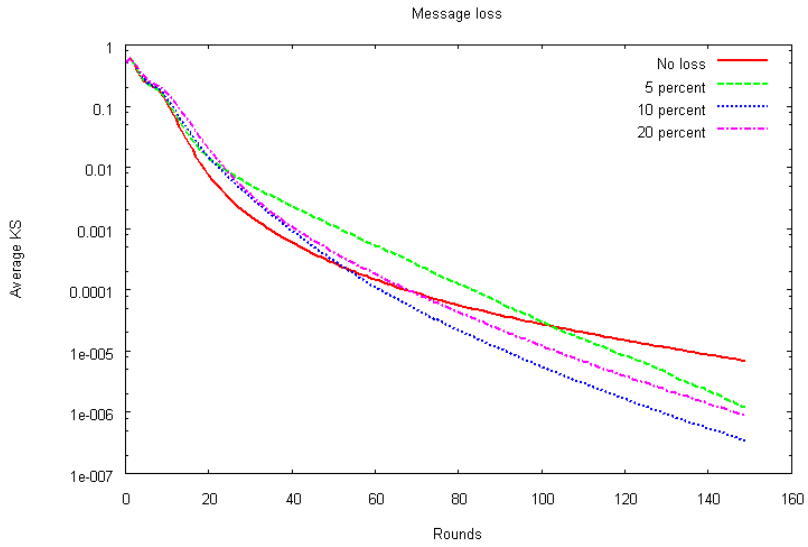

Figura 4.4: Efeito de perda de mensagens na estimativa para uma rede *random*.

combinada com a distribuição estatística dos valores amostrados, pode equivaler-se a uma topologia de grau de conectividade mais baixo e, tal como os resultados da experiências descritas no Gráficos 4.3 e 4.2, contribuir para a aceleração da convergência da estimativa. O efeito justificar-se-á ainda devido ao facto de o número de ciclos numa rede tende a deteriorar o desempenho da convergência no Flow Update [JBA09].

Perturbação e recuperação Para testar o comportamento do algoritmo sob dinamismo dos valores amostrados foi efectuada uma experiência com 1000 nodos em rede geométrica aleatória e de distribuição inicial dos valores amostrados $rNorm(10, 2)$. Na ronda 75 foi aplicada uma perturbação que fez com que 20% dos nodos escolhidos aleatoriamente seguindo uma distribuição uniforme, vissem o seu valor amostrado crescer 10% do valor amostrado pré-perturbação.

Apresenta-se no Gráfico 4.5 a resiliência da rede quando introduzidas perturbações nos valores da distribuição inicial (os valores amostrados). No caso, um aumento no valor amostrado V_i de 10% em 20% dos nodos da rede na ronda 75. No momento da perturbação o erro médio total sobe, para rapidamente convergir para um ajuste com um erro próximo de 10^{-5} na ronda 150. É mostrado em simultâneo o comportamento da mesma rede sem perturbação.

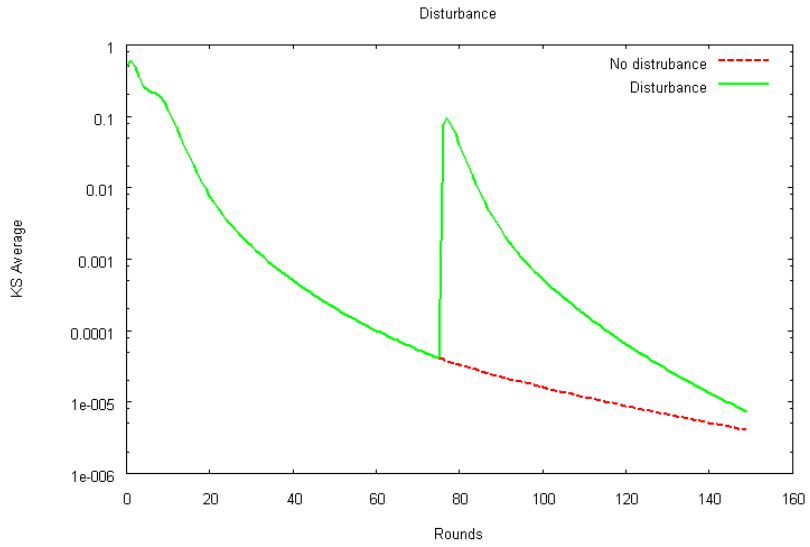

Figura 4.5: Perturbação e recuperação numa rede *random* com 1000 nodos.

Este resultado é ilustrativo da natureza adaptativa do algoritmo proposto. Sem necessidade de reiniciar o protocolo, o erro na determinação da CDF aumenta no momento da perturbação dos valores amostrados, convergindo rapidamente para valores de erro semelhantes aos que existiam pré-perturbação. Este comportamento decorre do facto de se estar a usar como protocolo de *averaging* o *flow update*. Com o valor amostrado alterado, um dado nodo recalcula o seu novo vector inicial tal como é referido na Secção 3.3. Em rondas subsequentes, propaga a nova estimativa e novos fluxos aos nodos vizinhos. Iterativamente, no decorrer das rondas todos os nodos ajustam as suas estimativas tendo em conta as perturbações introduzidas e as consequentes alterações em fluxos.

Normal vs. exponencial Por forma a verificar a adequação do algoritmo a diferentes distribuições estatísticas, procedeu-se a uma experiência com duas distribuições sintéticas dos valores amostrados: uma que segue a distribuição $rNorm(10, 2)$ e outra que segue a distribuição $rExp(1.5)$. A parametrização da distribuição exponencial foi escolhida por forma a reflectir uma concentração acentuada de valores nos primeiros percentís.

No que diz respeito ao comportamento da estimativa de CDF em presença de distribuições sintéticas diferentes(ver Gráfico 4.6), nota-se uma convergência mais rápida na distribuição exponencial $\lambda = 1.5$. Este comportamento deve ser depen-

dente do número de intervalos considerados (no caso $k = 20$) pelo que um número de intervalos baixo e com uma distribuição exponencial com λ maior deverá apresentar uma convergência mais lenta. Esta possibilidade resulta do facto de as distribuições exponenciais "concentrarem" as frequências nos intervalos inferiores da propriedade. Uma distribuição desta natureza implica que apenas alguns (poucos) nodos da rede

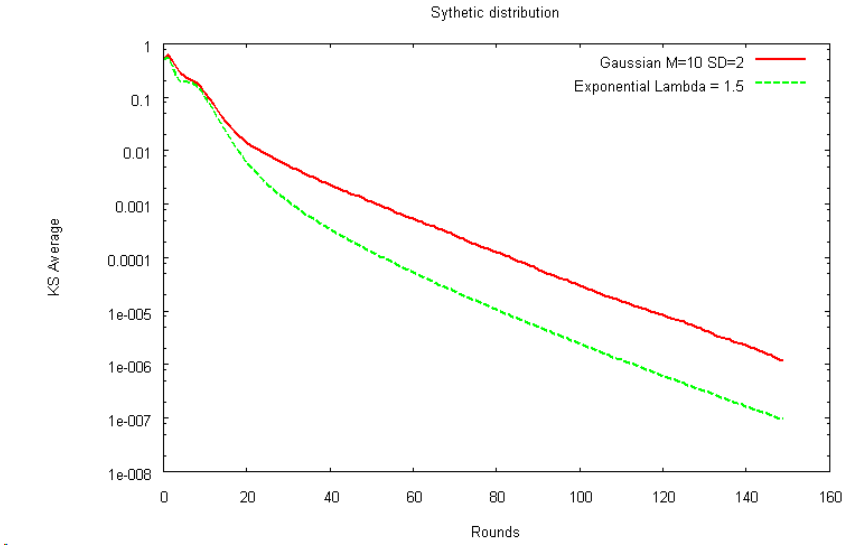

Figura 4.6: Normal *vs.* Exponencial.

concentrem em si valores extremos da distribuição, com elevada variância. A variância dos valores amostrados pelos restantes nodos da rede é baixa. Neste cenário, a contribuição dos nodos com elevada variância é rapidamente "absorvida" pela maioria dos nodos com baixa variância.

Entrada/ saída de nodos Nesta experiência foi usada uma rede de 1000 nodos com topologia geométrica aleatória de conectividade média 3.

Um outro dado referente à resiliência do algoritmo é mostrado no Gráfico 4.7. Este apresenta a evolução do erro da estimativa para a introdução de 25% de nodos novos entre a roda 50 e a ronda 75. São depois removidos 25% de nodos na ronda 125. Neste gráfico (4.7) é mostrado também um perfil dinâmico com o número de nodo que constituem a rede. Estes resultados mostram a adaptabilidade do algoritmo a elevadas taxas de entrada/ saída de nodos e o quão rapidamente o algoritmo converge para valores de erro relativo médio baixos. O cálculo da estimativa acompanha a

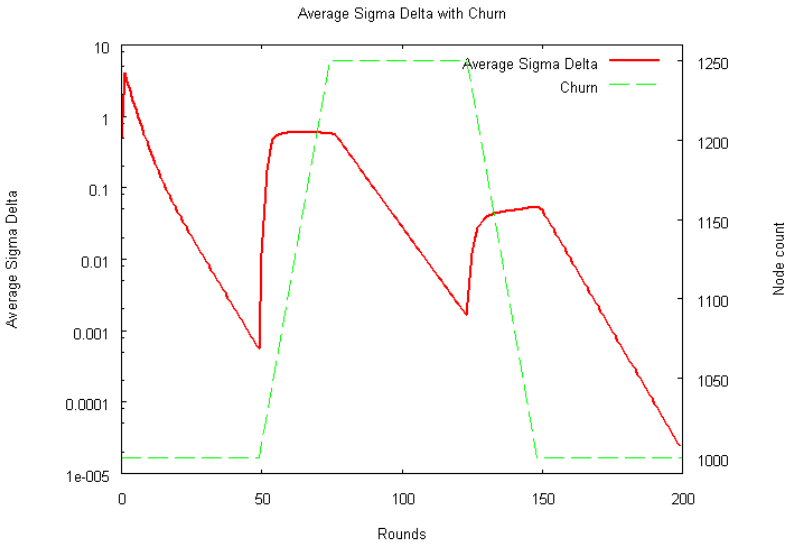

Figura 4.7: Entrada/saída de nodos.

dinâmica da rede sem a necessidade de reiniciar o algoritmo - esta propriedade emergente torna o algoritmo altamente adaptável e tolerante a faltas.

Quiesciência No Gráfico 4.8 é apresentada uma simulação com o mecanismo de quiesciência activado. Este foi parametrizado para um $\varepsilon_{low} = 0.01$. Com esta configuração é possível demonstrar a capacidade de adormecimento de toda a rede, dado o limiar superior nunca ser atingido. No mesmo Gráfico é apresentado o número cumulativo de mensagens trocadas entre os nodos. Pode observar-se que todos os nodos entram em estado quiesciente cerca da ronda 45 e que até então o número de mensagens trocadas vai diminuindo, acompanhando o adormecimento dos nodos. A troca de mensagens cessa assim que todos os nodos adormecem, como seria expectável.

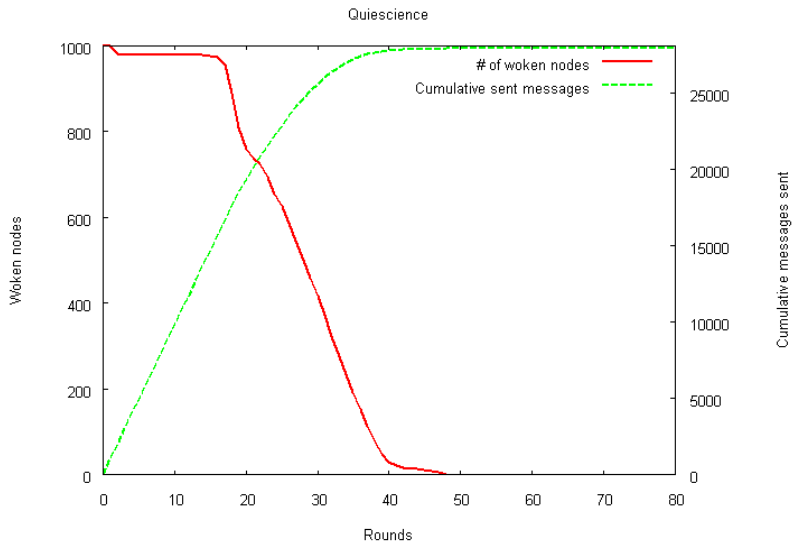

Figura 4.8: Quiesciência *vs.* número de mensagens trocadas.

Capítulo 5

Conclusões

É apresentado nesta tese um algoritmo distribuído para a determinação de funções cumulativas de probabilidade em redes de larga escala. Este algoritmo tem como principais vantagens a imunidade à perda de mensagens e a relação logaritmica entre o tempo durante o qual o algoritmo corre e a precisão que se obtém na aproximação à distribuição original. É simultaneamente adaptável a alterações no valor amostrado e resiliente a dinamismo no número de nodos na rede. Usa também um mecanismo de quiesciência dos nodos assim que a variação local da estimativa é inferior a um determinado limiar. Isto leva à diminuição do número de mensagens trocadas entre nodos. As distribuições determinadas por todos os nodos permitem a tomada de decisões que tirem partido do facto de se estar a agregar uma função de distribuição. Assim o nodo pode excluir *outliers* ou observar determinados quantis da propriedade. Por outro lado, cada nodo da rede possui uma estimativa global sobre o estado geral da propriedade distribuída, o que permite também a tomada de decisões com base em conhecimento local. Apresentamos também resultados de simulação que corroboram estas conclusões.

Foi em simultâneo apresentada uma revisão da literatura relacionada, cujo âmbito incluiu as técnicas mais representativas da agregação de dados para métricas escalares e as técnicas de agregação de dados para métricas complexas. Desta apresentação conclui-se que a agregação de dados em redes de larga escala, com foco na agregação de métricas escalares, encontra-se já extensivamente estudada na literatura. No entanto são ainda relativamente escassos os trabalhos que se focam sobre a agregação de métricas mais expressivas. De uma forma geral, as propostas

mostram limitações na tolerância a faltas e no suporte à monitorização contínua de propriedades ou na imposição de topologias lógicas que pela sua natureza tornam os algoritmos complexos e não compatíveis com dinamismo na rede.

5.1 Trabalho Futuro

Como trabalho futuro, pretende-se explorar a criação dinâmica de intervalos equi-frequentes por forma a aproximar os pontos de amostragem dos pontos onde a variação da função cumulativa é maior. Irá ser também avaliado o custo de comunicação e processamento em comparação com sistemas semelhantes. Pretende-se ainda estudar o comportamento do algoritmo para determinar funções cumulativas de distribuições multivalor - em que cada nodo contribui com mais de um valor. Pretende-se ainda o tratamento das funções cumulativas com funções contínuas que permitirão um maior nível de compressão da CDF e testar um novo algoritmo baseado na manipulação de funções simbólicas. Pretende-se também analisar de que forma distribuições probabilísticas correlacionadas com a distribuição espacial dos nodos e a sua variação no tempo, influenciam a qualidade da determinação da função cumulativa. Poder-se-á ainda avaliar o modelo proposto num sistema real de monitorização.

Bibliografia

[BAM09] C. Baquero, P.S. Almeida, and R. Menezes. Fast estimation of aggregates in unstructured networks. In *Autonomic and Autonomous Systems, 2009. ICAS '09. Fifth International Conference on*, pages 88 –93, April 2009.

[BBJA11] Miguel Borges, Carlos Baquero, Paulo Jesus, and Paulo Almeida. Estimativa contínua e tolerante a faltas de funções distribuição cumulativa em redes de larga escala. A publicar nas actas do inforum 2011, September 2011.

[CLKB04] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. *Proceedings. 20th International Conference on Data Engineering*, pages 449–460, 2004.

[CPX06] J.-Y. Chen, G. Pandurangan, and D. Xu. Robust computation of aggregates in wireless sensor networks: Distributed randomized algorithms and analysis. *Parallel and Distributed Systems, IEEE Transactions on*, 17(9):987 –1000, sept. 2006.

[DS05] Mads Dam and Rolf Stadler. A generic protocol for network state aggregation. In *In Proc. Radiovetenskap och Kommunikation (RVK*, pages 14–16, 2005.

[EKR10] Ittay Eyal, Idit Keidar, and Raphael Rom. Distributed data classification in sensor networks. In *PODC'10*, pages 151–160, 2010.

[ER60] P. Erdős and A Rényi. On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[FM85] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182 – 209, 1985.

[HR08] Maya Haridasan and Robbert Van Renesse. Gossip-based distribution estimation in peer-to-peer networks. *International Workshop on Peer-to-Peer*, 2008.

[JBA09] P. Jesus, C. Baquero, and P. Almeida. Fault-tolerant aggregation by flow updating. In Twittie Senivongse and Rui Oliveira, editors, *Distributed Applications and Interoperable Systems*, volume 5523 of *Lecture Notes in Computer Science*, pages 73–86. Springer Berlin / Heidelberg, 2009.

[JBA10] Paulo Jesus, Carlos Baquero, and Paulo Almeida. State of the art on distributed data aggregation algorithms. Technical report, Universidade do Minho, Braga, Portugal, 2010.

[Jel] Márk Jelasity. Peersim: A peer-to-peer simulator. `http://peersim.sourceforge.net`.

[Jel04] Márk Jelasity. Epidemic-style proactive aggregation in large overlay networks, 2004.

[Jes07] Paulo Jesus. Agregação e contagem em redes p2p. Master's thesis, Universidade do Minho, Braga, Portugal, 2007.

[JJW10] Hongbo Jiang, Shudong Jin, and Chonggang Wang. Parameter-based data aggregation for statistical information extraction in wireless sensor networks. *Vehicular Technology, IEEE Transactions on*, 59(8):3992 – 4001, oct. 2010.

[JS10] Dan Jurca and Rolf Stadler. H-GAP: estimating histograms of local variables with accuracy objectives for distributed real-time monitoring. *IEEE Transactions on Network and Service Management*, 7(2):83–95, June 2010.

[KDG03] D. Kempe, a. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 482–491, 2003.

[Lyn96] Nancy A. Lynch. Distributed algorithms. pages 17–23. Morgan Kaufmann Publishers Inc., 1996.

[MFHH02] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36:131–146, December 2002.

[MNN11] Baljeet Malhotra, Mario A. Nascimento, and Ioanis Nikolaidis. Exact top-k queries in wireless sensor networks. *Knowledge and Data Engineering, IEEE Transactions on*, 23(10):1513 –1525, oct. 2011.

[PHIS96] V. Poosala, P.J. Haas, Y.E. Ioannidis, and E.J. Shekita. Improved histograms for selectivity estimation of range predicates. *ACM SIGMOD Record*, 25(2):294–305, 1996.

[PJA10] C. Baquero P. Jesus and P. Sergio Almeida. Fault-tolerant aggregation for dynamic networks. *Reliable Distributed Systems, IEEE Symposium on*, 0:37–43, 2010.

[SBA04] N Shrivastava, Chiranjeeb Buragohain, and D Agrawal. Medians and beyond: new aggregation techniques for sensor networks. *networked sensor*, pages 239–249, 2004.

[SNSP10] Jan Sacha, Jeff Napper, Corina Stratan, and Guillaume Pierre. Adam2: Reliable distribution estimation in decentralised environments. *Distributed Computing Systems, International Conference on*, 0:697–707, 2010.

[WJXW07] M. Wu, J. Jianliang Xu, X. Xueyan Tang, and W.-C. Wang-Chien Lee. Top-k Monitoring in Wireless Sensor Networks. *IEEE Transactions on Knowledge and Data Engineering*, 19(7):962–976, July 2007.