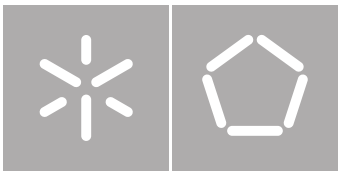
Universidade do Minho
Escola de Engenharia

João Pedro Jorge César

Fast Estimation of Network Size with Churn

Setembro de 2014



Universidade do Minho
Escola de Engenharia
Departamento de Informática

João Pedro Jorge César

Fast Estimation of Network Size with Churn

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de
Professor Carlos Baquero Moreno

Setembro de 2014

Agradecimentos

Agradeço ao Professor Carlos Baquero por me ter aceite ser meu orientador. Agradeço pela entrega e disponibilidade que sempre demonstrou, pelas soluções que sugeriu, pelas orientações que sempre me foi dado e que me possibilitaram concluir trabalho.

Não posso deixar de agradecer aos meus pais que sempre me apoiaram e me deram força para o trabalho.

Deixo também um agradecimento especial à Renata pela paciência que teve comigo e o apoio que deu.

Resumo

A agregação de dados é muito importante nos Sistemas Distribuídos de hoje em dia. Sistemas Distribuídos de larga escala são amplamente utilizados nos tempos de hoje, no entanto comportam a dificuldade de traçar um mapa global do sistema num determinado instante de forma a saber as propriedades do sistema. A agregação de dados é utilizada para estimar estas propriedades normalmente numa rede não estruturada. Existem várias abordagens para alcançar esta agregação de dados com diferentes características e aplicações. Neste documento será apresentada uma definição de agregação de dados e posteriormente o estado da arte da mesmo, explicando sucintamente várias técnicas usadas. Mais concretamente será introduzida a técnica de *Extrema Propagation*, que se considerou ser a mais interessante para o trabalho a realizar. Esta técnica não lida bem com CHURN. Com base nesta técnica tentou se arranjar uma solução que ultrapassasse essa limitação e assim resolver este problema. Foi então, desenvolvida uma técnica baseada na *Extrema Propagation* que espera uma rede estática para obter um bom desempenho, mas que agora se adaptasse melhor a constantes alterações na rede. Portanto foi desenvolvida uma variação desta técnica mas com funcionamento dinâmico. Após ter sido desenvolvida a nova vertente da técnica foram aplicadas métricas para medir o comportamento da mesma e analisados os resultados obtidos.

Key Words: Sistemas Distribuídos, Agregação de Dados; Extrema Propagation; estimação; CHURN.

Conteúdo

Lista de figuras

Capítulo 1

Introdução

O conceito de agregação de dados (AD) é essencial para os sistemas distribuídos (SD) de hoje em dia. Muita informação importante pode ser extraída através de AD. Por exemplo, pode ser utilizada para determinar o tamanho ou o diâmetro de uma rede, para saber a quantidade de informação armazenada em cada nó da rede, de forma a poder traçar uma visão global da rede e assim efetuar o *load balance* global.

Os dados obtidos podem então ser aplicados em diversos algoritmos relevantes na administração de SD, que é um dos objetivos da utilização de AD. Podem ser utilizados para efetuar o dimensionamento de *Hash Tables* distribuídas [16]. Dando outro exemplo, podem ser utilizadas para estabelecer o número de nós com os quais comunicar num algoritmo *gossip* [8].

Por vezes, não é necessário saber o exato valor da propriedade que se quer saber. Por isso, por uma questão de simplicidade e muitas vezes também para acelerar o processo, é utilizada uma estimação com um erro controlado.

Apesar da sua importância na gestão de um SD, alcançar AD não é trivial, uma vez que existem diversas soluções, o que leva a diferentes *trade-offs* entre eficácia, confiabilidade e complexidade, quer no número de mensagens enviadas, quer de tempo.

Devido a este elevado número de formas de alcançar AD é necessário analisar cuidadosamente as várias soluções. Certas abordagens funcionam

perfeitamente quando aplicadas em ambientes específicos, com uma estrutura necessária para funcionar ou com um comportamento bem determinado da rede. No entanto basta que surja uma nova variável no sistema para que estas soluções deixem de ser viáveis. Portanto têm de ser estudadas as condições favoráveis à sua aplicação bem como os seus prós e os seus contras, compará-los e decidir em que situações a aplicação dessas técnicas será aconselhável, estudo esse que vai ser descrito na capítulo 2.

É necessário um estudo exaustivo para adaptar o problema a topologias de rede distintas, com as respetivas propriedades e comportamentos.

Apresenta-se de seguida uma visão global da AD e as contribuição das técnicas de agregação mais usadas, comparando-as entre elas. Depois de escolhida a técnica a utilizar, esta será explicada com maior nível de detalhe e posteriormente serão apresentadas algumas soluções que usem essa técnica.

1.1 Agregação de dados

De forma muito simplista, AD pode ser vista como o processo de resumir informação ou simplesmente obter uma amostra de tamanho limitado de uma enorme quantidade de informação, para conseguir determinar o comportamento ou certas propriedades do sistema.

Para este processo ser efetuado são necessárias as chamadas funções de agregação. Habitualmente são usadas funções simples, tais como, `MIN`, `MAX`, `SUM`, `COUNT` e `AVERAGE`. Por exemplo, a função `COUNT` pode ser usada para calcular o tamanho da rede assim como a função `AVERAGE` para determinar a média da carga de cada nó no sistema, para aplicações em *load balancing*.

1.2 Motivação e objetivos

Os cenários previamente apresentados têm sido a base de investigação para resolver alguns problemas e limitações de SD. Alguns estudos dizem que AD é um ponto de partida para ter escalabilidade em serviços de larga

escala [19], que corresponde à capacidade de um SD conseguir ter uma boa resposta mesmo quando é exposto a uma elevada taxa de utilização. Apesar de haver várias soluções para ambientes estáticos há muito poucas variantes que tenham resultados satisfatórios quando o ambiente se torna dinâmico e há existência de alterações regulares na rede, quer sejam estas de entrada ou saída de nós do sistema. O fenómeno de AD por si só já não é uma tarefa fácil, agora utilizá-lo para obter escalabilidade torna a situação ainda mais complicada. A grande motivação deste trabalho é portanto encontrar uma solução passível de ser uma mais valia a utilizar tendo em conta os pressupostos anteriormente apresentados, o elevado *churn* na rede e que possam ser utilizada para garantir escalabilidade.

Após a conclusão do trabalho realizado e de atingidas as metas propostas é esperado atingir resultados satisfatórios que possam ser uma mais valia e que possam servir de base para um trabalho futuro, tal como o trabalho realizado por muitos outros foi aproveitado para realizar o trabalho apresentado.

Capítulo 2

Estado da arte

Num recente *survey* [12], diversos algoritmos foram propostos e exaustivamente comparados, analisando o que cada um trazia de novo para solucionar o problema quando aplicado em determinadas circunstâncias.

Os algoritmos podem ser divididos em duas grandes categorias, comunicação e computação. O ponto de vista de comunicação abrange tudo o que é relativo a topologias de rede e protocolos de encaminhamento. Já a perspetiva de computação é relativa às próprias funções de agregação. Cada técnica de agregação conjuga tipicamente algoritmos destas duas classes.

2.1 Comunicação

Algoritmos de agregação do ponto de vista da comunicação podem ser classificados em três categorias: estruturados, não estruturados e híbridos.

2.1.1 Algoritmos Estruturados

Estes algoritmos são dependentes de uma determinada topologia de rede e do protocolo de encaminhamento. Caso não estejam disponíveis as condições necessárias para o correto funcionamento do algoritmo, o processo tem de ser adaptado por forma a realizar uma computação extra que permita criar as condições ideais para o algoritmo correr. É fácil perceber que este tipo de

algoritmos é diretamente afetado pelos problemas inerentes à topologia de rede e ao protocolo de comunicação. Estão sempre limitados à capacidade da rede lidar com *churn* ou falhas, uma vez que basta a saída de um nó da rede para se perder um subconjunto da estrutura, alterando assim o resultado final. Devido aos problemas identificados pode se dizer que não funcionam corretamente em ambientes dinâmicos, já que, devido às constantes alterações da rede o processo de *setup* do algoritmo pode ser recorrente, trazendo custos adicionais à *performance* do mesmo e podendo obter resultados alterados devido a perdas de informação. São apenas aconselhados em ambientes livres de faltas e *churn*. Por outro lado, são rápidos e exigem poucas mensagens trocadas.

2.1.1.1 Abordagens hierárquicas

Este tipo de algoritmos pressupõe a existência de uma estrutura hierárquica, caso esta não esteja criada é necessário um *setup* inicial de forma a criar essa estrutura, o que cria desde já computação extra.

Topologia em árvore A estrutura criada é tipicamente uma árvore, como acontece na técnica **TAG** [15], embora haja recurso a outro tipo de estruturas mas para já vai se considerar como estrutura base uma árvore. Posta a configuração inicial da rede é efetuado o pedido de agregação que é iniciado pelo nó no nível superior da rede, a raiz. O pedido é propagado de nó em nó até atingir as folhas. Quando está concluída esta fase dá-se início à fase de resposta em que os nós encaminham os dados a serem agregados desde as folhas até à raiz, que junta toda a informação e computa o resultado final.

O facto de só o nó superior ter o resultado computado é uma desvantagem em relação a outros algoritmos em que todos os nós computam o resultado. A grande vantagem desta abordagem é a simplicidade do processo que garante um resultado exato computado de forma eficiente, tanto no número de mensagens trocadas como no número de computações efetuadas.

Outro dos problemas deste tipo de abordagem é que não tolera bem as

falhas. Por vezes, basta que a falha de um único nó possa dividir a estrutura, perdendo-se assim grande parte da informação relevante ao algoritmo, adulterando assim os resultados. Para suprimir este problema em ambientes dinâmicos são necessárias computações extra tal como na configuração inicial do algoritmo, o que em caso de elevado *churn*, levaria um *trade-off* elevado e que faz com que esta prática seja desaconselhada num ambiente dinâmico.

Topologia em anel Os algoritmos que usem esta topologia para fazer agregação de dados são pouco utilizados, uma vez que, comparada com a topologia em árvore apresenta ainda piores resultados quando ocorre a falha de um nó, já que a perda de nós pode mais provavelmente vir a separar a rede. Por outro lado perde a grande vantagem da topologia em árvore que é a eficiência. Demora muito mais propagar a informação pela rede porque há menos caminhos/ligações por onde propagar a informação. Isto normalmente resulta no aumento do diâmetro, que também é a causa de haver menos tolerância a faltas.

No entanto para contornar as limitações apresentadas, a técnica descrita por Horowitz e Malkhi [9], cria uma anel virtual na rede para estimar o tamanho da rede (*count*) tendo em conta apenas a entrada e saída de nós, ou seja, é uma técnica muito eficiente. Cada nó possui um estimador, para determinar o contador, e está ligado a um nó sucessor. Quando ocorre a entrada de um nó, a este é-lhe atribuído um nó sucessor e o respetivo contador. Ambos incrementam o contador em uma unidade. No caso de uma saída o acontece o inverso, o nó sucessor decrementa o contador. Este mecanismo leva as estimações dispersas na rede, já que só os pontos de contacto, e os nós a entrar ou sair é que sofrem alterações. Esta dispersão resulta numa variância de resultados estimados na rede entre $n/2$ e n^2, o que a define como uma técnica pouco precisa. Tem a vantagem de ter um custo de comunicação muito baixo.

2.1.2 Algoritmos não Estruturados

Já a classe de algoritmos não estruturados é independente quer de topologia de rede, quer de protocolos de comunicação. Geralmente estes algoritmos utilizam protocolos de disseminação para trocar mensagens por toda a rede. Esta abordagem tem como vantagem a simplicidade do processo, ao contrário da maneira estruturada é facilmente aplicável em ambientes dinâmicos, garantindo maior escalabilidade e robustez. Por outro lado, por vezes o elevado número de mensagens trocadas pode se tornar num problema, caso as mensagens congestionem a rede. Por isso é desejável que seja limitado, de forma razoável, o número de mensagens que podem ser trocadas.

2.1.2.1 Abordagens do tipo *flooding/Broadcast*

As abordagens do tipo *flooding/Broadcast* utilizam todos os nós para o processo de disseminação da informação pela rede, normalmente informação que tem como iniciante o nó inicial e como destinatários todos os nós da rede (*broadcast*). Este fenómeno provoca uma elevada taxa de utilização da rede por causa do elevado número de mensagens trocadas e por vezes implica algum tipo de centralização para mascarar o problema introduzido, por exemplo direcionar as mensagens para um número mais reduzido de destinatários. Por sua vez, o problema da centralização pode originar um *single point of failure* na rede, um ponto único de falha que pode estragar a computação final, o que não é o pretendido. Para evitar este problema, as abordagens deste tipo apresentam soluções para lidar com este problema.

Para melhor perceber o problema apresentado pode se ver com mais detalhe alguns algoritmos em específico e analisar os seus problemas e vantagens em concreto.

Randomized Reports Este algoritmo, de uma forma muito simples, consiste no envio do pedido à restante rede por parte do nó inicial, utilizando por exemplo a primitiva de *broadcast* e pela recolha das respostas recebidas. Utilizando tal estratégia rapidamente se pode observar alguma congestão na

rede porque independentemente da topologia da rede e da forma de comunicação haverá no mínimo n^2 mensagens trocadas, tendo conta que há partida este valor será sempre maior e tomando um n consideravelmente grande o problema será grave. Este problema já foi identificado na secção 2.1.2.1. Para evitar esse efeito de congestão na rede foi sugerida uma alteração neste algoritmo [3]. Pode ser introduzida na equação uma probabilidade p associada a cada nó que consiste na probabilidade de cada nó responder. Essa probabilidade pode ser ajustada de forma a melhor utilizar os mecanismos de transmissão da rede sem causar transtorno, aumentando ou diminuindo o p em função do tamanho estimado da rede. Depois de enviados os pedidos sabe-se que vai haver um número r de respostas em função do total de nós n com a probabilidade p. Logo para saber a estimação do número de nós, tem-se:

$$\widehat{n} = r/p \tag{2.1}$$

Com a introdução de p há um ganho no descongestionamento da rede mas perde-se a exatidão do valor calculado, pois passa-se de um valor exato para um valor estimado com um erro de precisão associado.

2.1.2.2 *Random-Walk*

Um *random-walk* constitui uma alternativa às abordagens tipo *flooding/ Broadcast*. A técnica **Random Tour** [17] nó inicial escolhe aleatoriamente um dos seus vizinhos e envia-lhe uma mensagem, o recetor dessa mensagem por sua vez escolhe também aleatoriamente um dos seus vizinhos que não tenha recebido a mensagem e assim por adiante, até que um determinado número de mensagens seja enviado ou seja atingida uma condição para que a mensagem retorne finalmente ao nó inicial. Para marcar os recetores da mensagem basta que cada recetor da mensagem, antes de enviar etiquete a sua identificação à mensagem. Como só uma pequena parte da rede é utilizada para percorrer o caminho até ao nó inicial, está implícito que o valor calculado não é exato mas sim uma aproximação. Este mecanismo está normalmente associado a mecanismos de *sampling* que serão estudados na

secção 2.2.5

2.1.2.3 Gossip

No conceito de comunicação por *gossip*, o nó iniciante escolhe aleatoriamente um subconjunto dos seus vizinhos e endereça-lhes uma mensagem, vizinhos esses que repetem o processo para um subconjunto dos seus respetivos vizinhos. Esta solução confina-se simples, robusta, eficiente, escalável e de forma completamente descentralizada. Em comparação com *flooding*(2.1.2.1) é tão eficiente como esta mas congestiona menos a rede.

Protocolo *Push Sum* O protocolo *Push Sum* [13] é utilizada para calcular funções de agregação como `SUM` ou `AVERAGE` que consiste na distribuição da pares de valores pela rede. Cada par de valor é representado por (s_{ti}, w_{ti}), com s_{ti} a representar a soma dos valores trocado e com w_{ti} o peso associado a essa soma, no nó i no instante t. Tendo como exemplo a função `AVERAGE`, com v_i o valor inicial no nó, tem-se $s_{0i} = v_i$ e w_{0i} para todos os nós. A cada iteração, é escolhido um vizinho aleatório e é adicionado metade do valor do par para nó escolhido e para o próprio nó. Todos os nós podem efetuar a estimação em qualquer momento, para isso basta calcular (s_{ti}/w_{ti}). A precisão do estimador vai aumentando ao longo do tempo, à medida em que vão sendo trocados os pares, convergindo para o valor correto.

2.1.3 Algoritmos híbridos

Os algoritmos híbridos [5] tem como intuito oferecer as melhores funcionalidades dos algoritmos estruturados e dos não estruturados, aliando a rapidez do processo à escalabilidade e à robustez, oferecendo uma solução melhor do que quando são utilizados isoladamente.

Habitualmente são combinadas a topologia hierárquica com um protocolo de *gossip*. Como já foi abordado na secção 2.1.1.1 este tipo de soluções hierárquicas apesar de muito eficiente e preciso, não é tolerante a faltas, enquanto que pelo contrário, um protocolo de *gossip* é naturalmente tolerante

e faltas mas é muito menos eficiente em termos de número de mensagens trocadas. Combinando estas duas soluções é possível mitigar as desvantagens de ambas e tornar a solução híbrida mais robusta.

Chitnis et al Este problema foi estudado em Chitnis em [5]. Foi observado que a agregação baseada em árvore é muito eficiente quando quase não há falhas mas essa eficiência cai abruptamente na ocorrência recorrente de faltas. Já o protocolo de *gossip* é apenas ligeiramente abrandado quando sujeito a faltas. Foi portanto sugerido um protocolo híbrido que combinasse efetivamente os dois protocolos de forma a obter uma solução tolerante a faltas num sistema de larga escala (large scale sensor networks). Numa primeira fase do algoritmo são definidos grupos para dividir a rede sendo uma agregação de dados *gossip* a todos os nós de cada grupo. Em cada grupo é eleito um líder e é feito uma AD baseada em árvore começando precisamente nos líderes de cada grupo.

Foi paralelamente realizado um estudo para obter o tamanho ótimo dos grupos em relação ao tamanho da rede e da probabilidade da ocorrência de faltas. No entanto é necessário a computação do tamanho do grupo de *gossip* antes de começar o algoritmo nas condições ótimas.

2.2 Computação

As cinco maiores categorias deste tipo de algoritmos são: Hierárquicas [15], Averaging [13], Sketches [6], Digest[20] e Sample[6].

Os artigos acima citados ajudam a perceber com mais detalhe este assunto, no entanto neste documento vão se explicar as vantagens e desvantagens de cada técnica e compará-las.

2.2.1 Hierárquicas

As abordagens *Hierárquicas* dependem da criação prévia de estrutura de comunicação hierárquica como já foi referido na secção 2.1.1 e por isso são

altamente dependentes do comportamento dessa estrutura. Na ocorrência de faltas toda a operação pode estar comprometida. Resumindo, esta técnica é eficiente e eficaz mas não é tolerante a faltas e só o nó no nível mais alto da estrutura tem acesso ao correto valor computado. É um processo simples com cálculo exato dos dados agregados (em caso de não haver ocorrência de faltas). Neste trabalho é suposto lidar com o problema de *churn* (constante entrada e saída de nós da rede), e por esse motivo este método não é aplicável ao problema. Pode se concluir então que este método em geral, quando usado isoladamente, só é aconselhável quando o sistema não está sujeito a faltas.

2.2.2 *Averaging*

A técnica de *Averaging* consiste em algoritmos em que para cada iteração, cada nó calcula a média do seu valor com o valor dos nós vizinhos. Tipicamente este valor corresponde à estimativa que cada nó tem nesse preciso momento da característica a calcular pelo algoritmo, à medida que vão sendo trocadas mensagens entre vizinhos, cada nó vai atualizando o seu valor em consequência da receção dos valores dos vizinhos e guarda o valor médio. Este processo é muito preciso já que os valores acabam por convergir ao longo do tempo e tendem para o valor exato [13], que é a grande vantagem deste tipo de abordagem.

Quando o algoritmo estabiliza e é atingida essa convergência de valores em todos os nós, que sabem o valor computado, ao contrário dos algoritmos hierárquicos em que apenas o iniciante tem acesso ao valor computado. Um bom exemplo deste tipo de agregação é protocolo de *gossip*, como é o caso do protocolo **Push-Sum** [13], em os nós vão partilhando o seu valor médio calculado com as respostas dos restantes nós com vizinhos escolhidos aleatoriamente, contribuindo para que os valores convirjam ao longo do tempo para o valor correto.

Para que tal aconteça, para que os valores convirjam para o resultado correto, é necessário garantir o princípio da conservação da massa que diz que a soma dos valores agregados em todos os nós da rede deve permanecer

constante enquanto é executado o algoritmo [13]. Também contrariamente aos algoritmos hierárquicos, este tipo de algoritmos para além de produzir resultados em todos os nós é tolerante a faltas. Inclusivamente é considerado como a técnica que apresenta o melhor comportamento na presença de faltas.

Apesar de no plano da complexidade computacional esta técnica apresentar um bom comportamento, já que apenas envolve a computação de operações simples utilizando poucos recursos computacionais, resultando num processo simples. Esta técnica exige um número mais elevado de mensagens trocadas em comparação com a abordagem hierárquica e por conseguinte, resulta num processo menos eficiente, nesta medida.

Como é uma técnica que se adequa bem a ambientes dinâmicos, tolera bem falhas e é precisa, acaba por ser aplicável à resolução problema proposto.

O melhor exemplo em termos de técnicas e *averaging* é à técnica de *Flow Updating* [11] que é a técnica deste tipo recomendada para a maioria dos casos.

Flow Updating A maior parte das técnicas de *averaging* utiliza o valor presente em cada nó que é trocado com os restantes obtendo um valor médio em cada nó ao longo do tempo. Esta técnica [11] utiliza outra estratégia, utiliza o conceito de fluxo (*flow*) que está associado a uma aresta, ligação entre dois nós. Estes valores de fluxo que são calculados obtêm, inicialmente, a média entre o valor inicial v_i e v_j respetivo a cada nó. Os fluxos f_{ij} e f_{ji} (associados a v_i e v_j respetivamente) são relativos à aresta entre os nós e são valores simétricos ($f_{ij} = -f_{ji}$) que subtraídos aos dois nós produzem a mesma estimativa nos dois nós. A estimativa $e_i^{'}$ é calculada periodicamente em cada nó utilizando a estimativa local e_i, subtraindo os *flows* dos vizinhos ao seu valor inicial v_j, e é feita a média com os e_j calculados e enviados pelos respetivos vizinhos. Os valores f_{ij} locais são atualizados de forma a refletir a diferença entre a nova estimação $e_i^{'}$ e o valor estimado pelo respetivo nó e_j, esta diferença é adicionada ao fluxo. É por fim enviado o simétrico do fluxo calculado ao seu nó e o valor da nova estimação para garantir o

princípio de conservação de massa. Outro requisito para que tal aconteça é que o somatório dos fluxos de todos os nós tem de manter constante ao longo do tempo. O valor inicial de cada nó ao contrário dos outros algoritmos é inalterado ao longo da execução do algoritmo. A realização deste processo continuamente leva à convergência dos valores estimados pelos nós.

A grande vantagem desta técnica é o facto de ser tolerante à perda de mensagens, caso este fenómeno aconteça não leva a resultados incorretos apenas retarda a convergência. Como os fluxos são apenas referentes a dois nós, caso haja perda de nós é fácil de lidar com a situação descartando o *flow* do nó removido e a convergência é posteriormente alcançada depois de removido este fluxo, que deixa de entrar nas contas da estimativa local. É portanto resistente a perda de mensagens e de nós adaptando-se a mudanças na rede e lida bem com *churn* [10].

2.2.3 *Sketches*

Tal como o método de *averaging*, o método de *sketches* [6] funciona corretamente independente de topologias de rede ou protocolos de comunicação e é também tolerante a faltas. O que difere estes dois tipos de abordagem é o objetivo de cada uma. Se o objetivo do método de *averaging* é a precisão, neste caso o objetivo é a rapidez do processo, uma estimação rápida.

Se para o método de *averaging* são utilizados valores que representam uma estimativa da característica da rede a calcular, como por exemplo o tamanho da rede, que vão sendo trocados e atualizados ao longo das iterações, no caso dos algoritmos de *sketches* são utilizadas estruturas auxiliares, normalmente de tamanho fixo, com o intuito de representarem um esboço da informação da rede para serem posteriormente utilizadas para extrair informações sobre a mesma. Estas estruturas são criadas no *setup* do algoritmo, normalmente com o recurso a um gerador de números aleatórios, ou através de um função probabilística que tem como input alguma característica do nó. Ao longo do algoritmo estas estruturas vão sendo atualizadas ou fundidas pelas estruturas dos nós vizinhos. É independente da topologia da rede e permite vários

caminhos para a propagação da informação.

Existem dois grandes grupos de técnicas de *sketches*, as de *hash* e as de mínimos (*k-min*).

As *hash sketches* têm como objetivo a contagem, na estrutura, de elementos distintos. A estrutura representa um mapa de bits que inicialmente está repleto de zeros. É calculado o *hash* do valor de cada nó e mapeado na estrutura na posição calculada com valor 1. Depois de os valores serem trocados entre os vários nós é contado o número de zeros para obter indiretamente a contagem de elementos distintos.

RIA-LC/DC Com esta técnica [7] numa primeira fase é feito o pedido pela raiz, pedido este que é propagado por toda a rede e cada nó cria o seu *sketch*. Partindo de um nível inferior as folhas vão enviando os seus *sketches* que vão ser combinados com os nós superiores, através de disjunções (operações OR), à medida que vão subindo até ao topo da hierarquia até chegar ao topo que assim calcula o valor final. Esta técnica não produz um valor em cada nó mas existem outras técnicas baseadas em sketches que o fazem. Apesar da estrutura hierárquica de recolha, a operação OR é tolerante a *multipath*.

As técnicas de *k-min sketches* consistem em definir vetores de k elementos, gerados de forma aleatória, trocados pelos vários nós, até se obter o vetor k com os valores mínimos na rede, para cada índice de k. São então utilizados os k valores mínimos para extrair o tamanho da rede.

Extrema Propagation A técnica de *Extrema Propagation* [1] utiliza a vertente de *k-min* para calcular o tamanho de uma rede. Cada nó inicializa o seu vetor x gerando k valores através de um gerador de números aleatórios. Estes valores são trocados com os nós vizinhos e são atualizados, quando recebem um valor dos vizinhos que é menor que o atual para a mesma posição do vetor. A eficiência desta técnica depende do diâmetro d da rede. Após iniciado o algoritmo passadas d rondas, os valores de x em todos os nós convergem (para o mesmo valor dos x_i em cada nó e o processo de troca de

informação acaba por estagnar, o que leva a que não haja um aumento da precisão da estimação, que é um dos problemas desta técnica que se propõe resolver neste trabalho). O vetor x é por fim utilizado como *input* a um estimador que extrai a informação do vetor e produz a estimativa para o tamanho da rede. O objetivo desta técnica não é um valor preciso mas sim uma estimativa rápida.

A complexidade computacional deste tipo de algoritmos pode resumir-se à complexidade das operações de criação e alteração dos *sketches*.

Esta técnica é normalmente rápida mas depende sempre do protocolo utilizado para propagar a informação, chegando perto do mínimo teórico, o diâmetro da rede. É confiável mas introduz um erro de estimativa. É possível aumentar a confiabilidade da estimativa aumentando o tamanho dos *sketches* mas existe naturalmente um *trade-off* entre o aumento do tamanho do vetor k e o aumento dos recursos computacionais e de transmissão necessários para computar a alteração do tamanho. Para além de ser uma técnica rápida é dependente do parâmetro k. O processo é portanto dependente do tamanho da mensagem em vez do tamanho da rede e por essa razão define-se um processo escalável.

2.2.4 *Digest*

Esta classe de algoritmos comprime informação através de função probabilísticas, com o problema de perder informação no processo. São utilizadas para obter um resumo da informação disponível na rede que é posteriormente utilizado para obter as funções de agregação. Tal como *sketches*, não produzem um valor exato mas neste caso é produzida uma aproximação, muitas vezes grosseira. Um *digest* pode ser visto como uma distribuição de dados estatísticos dos valores na rede. É feito este resumo da informação por questões de eficiência e de escalabilidade. Esta aproximação é pior em relação a uma estimação, em termos de precisão, e requer mais recursos computacionais para se tentar aproximar da qualidade de estimação de outras técnicas, mas

tem a particularidade de operar com funções de agregação mais complexas, o que é a sua grande contribuição para este tipo de problemas. Para além de poder utilizar as funções de agregação convencionais como `min`,`max`,`count` ou `min` tem a vantagem de se poder utilizar funções como `median` ou `frequency` como acontece na técnica **Q-Digest** [20].

2.2.5 *Sample*

As técnicas de *sampling* consistem num processo aleatório de recolha de informação de forma a providenciar uma aproximação probabilística do valor desejado, normalmente um `count`.

Estes algoritmos são muito afetados pelas funções de probabilidade que utilizam no processo de aproximação e lidam com as suas limitações. Se a função de probabilidade não for suficientemente boa, ou seja, se não produzir amostras suficientemente aleatórias, os resultados obtidos não são úteis. O grande problema deste tipo de técnicas assenta na dificuldade de encontrar um boa função probabilística, uma função suficientemente aleatória, que é uma das grandes questões afetas à segurança dos sistemas de informação. Resumindo este tipo de técnicas está sempre limitado pelo fator de erro presente na amostra gerada.

De uma forma geral, a aproximação apesar de ser eficiente é feita num único nó, é lenta e errónea. Portanto, não é considerada muito interessante já que não traz grandes vantagens em relações às outras classes de algoritmos a não ser alguma eficiência no número de mensagens trocadas.

Capítulo 3

Planeamento de Investigação

3.1 Métodos de Pesquisa

Para chegar a uma solução para o problema em mãos foi necessário:

1. Estudar o problema: Para compreender a fundo um problema tão complexo foi necessário um estudo exaustivo para analisar o que já se encontrava e feito e conhecer as várias soluções aplicáveis ao problema com as suas condições de aplicação e os correspondentes prós e contras.

2. Escolher a solução mais apropriada: Depois de se ter feito o ponto da situação, foi necessário escolher de entre as várias soluções estudadas aquela que se pensa ser a melhor solução para lidar com os problemas definidos.

3. Implementar algumas variantes da solução escolhida de forma a poder compará-las entre si e poder averiguar qual será a melhor ou saber em que condições podem ser aplicadas para resolver o problema quando aplicado diferentes ambientes.

4. Avaliar os resultados: Depois de encontrados os resultados dos processos a realizar, serão discutidos como contribuíram para o estudo do problema e para a evolução do trabalho realizado.

3.2 *Extrema propagation*

A técnica de agrecação seguida neste trabalho é a *fast estimation with extrema propagation* [1],[2],[4],[12] para estimar o tamanho da rede, neste caso, porque esta técnica apesar de não ser muito precisa é simples e rápida e no caso de estudo em mãos a grande preocupação é a rapidez de acesso ao resultado numa rede dinâmica não estruturada. Para além disso é totalmente distribuída, com cada nó a computar o resultado e sem nenhum ponto de falha. Como foi visto no capítulo 2 existem diversas técnicas que calculam o tamanho de uma rede. A razão pela qual esta técnica foi escolhida reside no facto de ser a técnica que melhor combinada a tolerância a faltas com a rapidez de processo e foi por isso escolhida como ponto de partida para encontrar uma variante que lide melhor com a saída de nós(o maior problema desta técnica, que perde precisão com a saída de nós) e aumente a precisão do algoritmo ao longo do tempo mantendo a rapidez de execução.

3.2.1 Funcionamento

O sistema é tipicamente uma rede grande, não estruturada, com um tamanho desconhecido que se pretende determinar num determinado momento. Considera-se para caso de estudo que a rede pode ser num primeiro estudo estática, para passar posteriormente ter um comportamento dinâmico. Numa fase inicial considera-se que o SD é síncrono, operando em rondas. Contudo, os métodos são adaptáveis a sistemas assíncronos. Serão considerados três modos de funcionamento do algoritmo: estático, dinâmico na entrada de nós, dinâmica na entrada e saída de nós.

3.2.2 Estático

Neste primeiro modo, por uma questão de simplicidade, é considerada uma rede grande não estruturada que se mantém inalterada ao longo do processo, ou seja não há novos nós a entrar, nem nós a sair. Muitas vezes o cenário a considerar não é estático, e quando assim acontece acaba por propor

um desafio maior, daí os modos seguintes já contemplarem um ambiente dinâmico.

Para aplicar *extrema propagation* (EP), cada nó tem de ser gerado aleatoriamente um vetor x de tamanho k, com os k valores a serem gerados por $rExp(1)$, uma função que retorna um número aleatório com distribuição exponencial de razão 1. É portanto um requisito o acesso a um gerador de números aleatórios. O tamanho k do vetor é muito importante no processo, uma vez que é este que fixa a qualidade da precisão do processo. Quão maior for o k maior será a precisão, mas por outro lado, quão maior o for k maior será o tamanho da mensagem a enviar e como consequência maior será o *overhead* na transmissão.

Apesar desse problema, o facto de a precisão ser dada pelo tamanho do k revela que a precisão do processo de estimação é independente do tamanho da rede [14], o que é fundamental para escalar bem em redes grandes.

EP é um processo iterativo. Em cada iteração cada nó envia o seu vetor x aos vizinhos. Após enviar o seu vetor, recebe as mensagens dos nós vizinhos. Por cada mensagem recebida é verificado, para cada entrada do x do vizinho, se o valor presente é menor que o valor do próprio x. Caso seja menor, o valor do próprio x é substituído pelo do vizinho. Passadas d iterações, normalmente o diâmetro do grafo, todos os nós têm um vetor x igual, por se estar diante de um ambiente estático e sem faltas.

O último passo é fazer a seguinte computação:

$$Sum = \frac{K-1}{\sum_{i=1}^{K} x[i]} \tag{3.1}$$

com k o tamanho do vetor x. O valor obtido é a estimativa do tamanho da rede.

O algoritmo funciona muito bem em ambientes estáticos. Ao fim de d rondas (normalmente o diâmetro da rede), é calculada a solução e todos os nós estão sempre atualizados.

3.2.3 Dinâmica na entrada de nós

Considerando o modo de funcionamento secção anterior, pretende-se introduzir uma nova variável no sistema, que é a entrada de novos nós. O funcionamento deste modo é em todo semelhante ao modo de funcionamento previamente descrito, apenas com algumas alterações.

A cada nova entrada de um nó, é inicializado o seu vetor x. Para poder ser calculado de novo o tamanho da rede têm de esperar d rondas para que os vetores convirjam e que nós antigos da rede possam trocar os vetores com o novo nó, de forma ao mesmo tempo de atualizarem e atualizarem o novo nó. Não é necessário que uma paragem do algoritmo para que as valores convirjam, não é esse o objetivo desta técnica que prima pela rapidez do processo. Quando se diz que tem de se esperar d rondas, na verdade, quer-se dizer que o algoritmo só vai dar uma valor mais preciso e próximo do valor real, e que a informação do novo nó só atualizará os restantes nós e vice-versa passadas d rondas.

Para este modo existe solução, no entanto, o facto de o sistema demorar a ser atualizado pode ser constrangedor para o caso de uma constante entrada de novos nós numa rede grande, possivelmente com um diâmetro também grande, já que a atualização está dependente do diâmetro da rede.

Contudo o diâmetro da rede estabelece em geral um limite mínimo do número de rondas para a propagação de nova informação por toda a rede.

3.2.4 Dinâmica na entrada e saída de nós

Para a entrada de novos nós, a resposta é igual ao modo anterior. Agora se se considerar agora a saída de nós do sistema, existe um grave problema. À medida que vão saindo os nós da rede, os valores desses nós nos vetores x, não vão sair do vetor. Pode acontecer o caso um novo nó entrar no sistema e provocar a atualização de alguns desses valores, no entanto numa situação normal em que haja um número semelhante de entrada e saída de nós há o problema de os valores de nós que saíram da rede ficarem a poluir o sistema,

é a chamada memória de mínimo. Isto implica uma diminuição acentuada na precisão do estimador uma vez que se ao saírem nós do sistema este não atualiza os *sketches*. É fácil prever que o resultado final do estimador vai conter mais nós na rede do que realmente há. O capítulo seguinte aborda este problema e apresenta soluções concretas para a sua resolução.

3.3 Abordagens propostas

Embora EP seja um processo rápido, se a estimação não for precisa devido ao fenómeno de *churn*, a solução apesar de rápida pode não ser suficientemente precisa para que a estimação tenha algum valor. A técnica de EP num cenário em que ocorram faltas não é preciso, por isso o grande objetivo é combinar a simplicidade e rapidez da EP(sketches) com a robustez e precisão das técnicas de *averaging*, que é técnica que melhor lida com *churn* e com faltas.

É portanto um objetivo aumentar adaptabilidade, isto é, esquecer as entradas antigas que podem conter informação desatualizada.

A técnica estudada tem outro problema, quando a rede estabiliza o processo de estimação estagna. Isto acontece porque está previsto no algoritmo que o vetor x tenha um tamanho k fixo. Seria interessante obter uma variante deste algoritmo que possa tirar partido comunicação de forma a aumentar a precisão do estimador, bastando para isso que o vetor x não seja fixo mas sim que seja aumentado gradualmente o seu tamanho.

Quando forem alcançadas isoladamente as duas metas traçadas anteriormente, adaptabilidade e precisão, seria interessante também conjugar as duas versões numa nova solução e analisar os resultados.

3.3.1 Vetor Dinâmico

A medida encontrada por forma a aumentar a precisão do estimador foi garantir que o vetor x seja dinâmico. O processo começa com um valor k típico 387, valor teórico do k para um erro relativo máximo de 10% (com

confiança de 95%) [1]. O processo inicial corre como definido na secção 3.2.2. Assim que corram d rondas (o algoritmo já estabilizou), a cada ronda que passe, é acrescentada uma nova célula ao x. Essas novas células são enviadas aos nós vizinhos que as substituem caso sejam menores, tal como no processo base. Por uma questão de eficiência, poderão ser apenas enviadas as novas células em cada iteração. As k células base já são conhecidas por todos os nós, não é necessário que sejam enviadas de novo, isto no caso de não haver entrada de novos nós. Passadas d iterações, essas células passarão a fazer parte células estáveis e não será necessário que sejam enviadas outra vez. O número de células novas a gerar por cada ronda é parametrizável. Para tirar partido dos canais de comunicação da rede é mais vantajoso enviar blocos de células em vez de uma de cada vez alcançando assim mais rapidamente uma maior precisão. Foi dado o exemplo com uma célula por uma questão de simplicidade. Por outro lado, caso se enviem blocos de células já não é necessário que este envio seja efetuado em todas as rondas, pode-se por exemplo enviar de d em d rondas.

3.3.2 Janela Deslizante

Novamente é utilizado um vetor de tamanho k. O processo corre tal como definido na secção 3.2.2. Para aumentar a adaptabilidade, isto é, para atenuar o efeito dos nós que saíram, sempre seja detetado que um nó tenha saído, o vetor x é truncado num tamanho bastante inferior a k, fazendo esquecer parte das células e assim perdendo grande parte da informação contaminada. A cada ronda é gerada uma nova célula tal como na secção 3.3.1, e adicionada ao vetor x. Quando x voltar atingir o tamanho k os valores usados na truncatura vão sendo apagados a cada iteração com a consequente geração das novas células até que todas as células antigas sejam esquecidas. Quando este processo estiver concluído a operação para até que seja detetada nova saída de um nó. Pode dar-se o caso de ocorrerem novas saídas de nós antes de o processo estar concluído, podendo optar-se por continuar o processo caso ele esteja no início e no fim apagar os valores antigou ou por

reiniciar caso contrário.

Pode ser discutida a vantagem de utilizar uma truncatura em vez de começar do zero. Esquecendo todos os valores anteriormente, a convergência do algoritmo é atingida muito mais depressa (dependendo do tamanho da truncatura).

3.3.3 Solução combinada

O objetivo desta solução é combinar o aumento de precisão, usando um vetor dinâmico, com a adaptabilidade alcançada com uma janela deslizante. Este modo irá funcionar de dois modos, modo normal e modo de descontaminação.

No modo normal, os nós vão aumentando o tamanho do k gradualmente de forma a aumentarem a precisão. Assim que um nó detete que houve uma saída de um nó, é ativado o modo de descontaminação em janela deslizante, tal como é descrito na secção 3.3.2. Quando o algoritmo estabilizar, no fim da fase da descontaminação, é alterado de novo o modo para o modo normal para voltar a ganhar precisão. No caso de uma rede muito dinâmica, pode se dar o caso de nunca ser atingido o modo normal.

3.3.4

Capítulo 4

Trabalho realizado

Neste capítulo serão abordadas dois pontos essenciais do trabalho desenvolvido, tendo como ponto de partida a técnica de *Extrema Propagation* estático, até chegar à versão de *Extrema Propagation* dinâmico alcançada, descrevendo a evolução da estratégia seguida para partir da solução existente até à definição da que se propõe nesta dissertação.

4.1 *Extrema Propagation* - Estático

A primeira abordagem ao problema de EP foi feita através de um ambiente estático de forma a evitar ter de lidar com a entrada e a saída de nós, problema de complexa resolução. Essa abordagem consistiu em estabelecer um tamanho inicial do vetor x e ir aumentando a precisão do estimador através da introdução de novas células ao vetor x. Contudo, esta introdução de novas células tem de ser controlada porque senão a precisão que se pretende ou não é atingida ou demora mais a ser alcançada. Por isso, a cada vez que são introduzida novas células é necessário esperar D rondas, com D correspondendo ao diâmetro da rede, para que as células introduzidas tenham tempo de ser trocadas pelos nós e estarem atualizadas no momento da estimação. Este controlo do tempo de estabilização é efetuado com recurso a uma técnica que efetue o cálculo do diâmetro, existem várias técnicas que

o fazem como é o caso do artigo [4], o cálculo dessa função não faz parte do âmbito deste trabalho. Posta esta espera de D rondas é possível então gerar mais células de forma a aumentar a precisão do algoritmo. Pode ser por exemplo utilizado o mecanismo de do EP descrito em [4].

Para calcular a estimação do tamanho da rede em qualquer momento basta computar a seguinte equação:

$$Sum = \frac{K-1}{\sum_{i=1}^{K} x[i]} \tag{4.1}$$

tendo como requisito, para que a estimação seja boa, utilizar os valores de x que estejam estáveis (que já tenham sido trocados por todos os nós em D rondas). É preciso portanto saber o valor do diâmetro da rede para poder gerar novas células de forma mais otimizada.

Inicialmente foi prevista inserção de uma célula de cada vez que houvesse expansão do vetor x mas achou-se que seria mais vantajoso tirar partido da largura de banda da rede e enviar blocos de células, de forma a agilizar o processo de aumento de precisão.

4.1.1 Descrição da Simulação

Por forma a permitir a avaliação das soluções optou-se, de uma forma simplista, por criar um grafo de uma topologia achada interessante para o caso de estudo, para poder correr n testes com i iterações cada um. De seguida correm-se os mesmos testes mas agora para topologias diferentes, com características distintas. São criados grafos com tamanho grande e para cada nó é gerado o vetor x de k valores aleatórios. Segue-se então uma sequência de rondas em que os nós vão trocando informação entre eles e produzindo estimações do tamanho da rede. Em cada ronda os nós trocam, com os respetivos vizinhos, os valores x_i a serem tratados no momento, por forma a, que todos os nós, no fim de D iterações, tenham vetores x iguais, que em cada um dos nós se possa estimar o tamanho da rede e possam ser geradas novas células x_i. Estas novas células passarão agora a ser a

informação trocada entre nós para assim aumentar a precisão do processo. Para guardar a informação que chega a cada nó por parte dos seus vizinhos é utilizado um vetor auxiliar, o vetor primário que é utilizado para atualizar por sua vez os vizinhos só é atualizado com os valores do secundário no início da iteração seguinte, para evitar que na simulação realizada não possa ocorrer uma propagação da informação mais rápida do que o possível num cenário real. Este mecanismo é feito por se tratar de uma simulação de uma rede utilizando um grafo, e como se trata de uma simulação, a troca de informação entre nós é imediata, não existindo um tempo de propagação da mensagem. Com esta introdução de dois vetores quer-se garantir que um vetor não está a utilizar informação que ainda não devia ter na ronda atual mas que só devia ter na ronda seguinte e assim garantir resultados mais credíveis.

Os resultados são guardados e por fim é apresentado um gráfico com o resultado da média dos valores obtidos por iteração, nos vários testes, juntamente com o valor correto da rede (em *GodMode*) e ver a convergência das linhas. Por *GodMode* assume-se a propagação instantânea da informação por todos os nós da rede, uma propagação em zero rondas.

4.2 Extrema Propagation - dinâmico

Agora considerando um cenário mais realista, em que a entrada e a saída de nós fazem parte do comportamento normal de um rede grande, a abordagem estática deixa de fazer sentido. Tomando primeiramente o caso da entrada de nós a estratégia anterior resulta em parte. Sempre que entrar um nó, este nó tem que trocar informação com os restantes nós e ao mesmo tempo se atualizar e atualizar os restantes. No entanto surge um problema, por questões de eficiência computacional e de tráfego da rede em circunstâncias normais só são enviados os valores acrescentados ao vetor x, mas como se viu quando há entrada de um novo nó tem que se trocar toda a informação de x tendo que se trabalhar agora em dois modos, o modo normal e modo manutenção. Apesar desta modificação, a solução acaba por convergir para

o correto valor esperado.

Contrariamente ao que ocorre na adição de nós, a remoção de nós, mesmo com a operação em dois modos, acaba por não convergir para os valores esperados porque quando os nós saem, a informação que transmitiram a todos os nós continua presente nesses mesmos nós, mesmo depois de terem saído da rede. Ficam portanto a contaminar toda a rede. É necessário então esquecer informação mais antiga, uma nova variável nesta equação.

4.2.1 Solução inicial

Inicialmente pensou-se em operar em dois modos, o modo normal e modo contenção. Em teoria quando a rede estivesse estável o algoritmo funcionaria em modo normal, ou seja, aumentando a precisão do estimador ao longo do tempo e mudando para o modo de contenção assim que notasse a remoção de algum dos nós. Num cenário de elevado *churn* seria difícil obter a precisão desejada devido a uma possível saída frequente de nós, já que o algoritmo passaria grande parte do tempo a alternar entre estados, com um custo associado a essa alternância entre estados mais o custo de determinar em que cenário é que se encontraria o algoritmo. Foi devido a esta conjetura que se tentou uma abordagem diferente, independente de modos de operação.

A partir deste ponto houve uma viragem na estratégia a utilizar. Depois de analisar o trabalho realizado e as várias abordagens propostas e implementadas chegou-se à conclusão que se podia utilizar uma nova abordagem mais vantajosa que consiste um agregar conjuntos de valores em blocos de n valores e a armazenar a sua soma (`count`) de forma a que cada bloco contenha uma enésima parte do vetor, para que pudessem ser atribuídos determinados pesos a cada bloco, de forma a conseguir que seja dada uma maior ênfase quer ao passado quer ao presente dependente das situações. Caso tenha sido já atingida estabilidade será dado um maior peso aos blocos mais antigos, no entanto quando houver casos, quer de entradas, quer de saídas de nós será mais vantajoso atribuir um maior peso aos novos blocos já que assim se conseguirá atingir valores precisos mais rapidamente, uma vez que apenas os

blocos mais recentes refletem o tamanho mais atual da rede.

Por outro lado estavam ainda previsto na secção 3.3 depois de alcançar uma maior precisão do algoritmo mantendo a rede estática, passar posteriormente para o modo janela deslizante em que a rede passa a ser dinâmica para aumentar a adaptabilidade. Numa fase final passar-se-ia a combinar as duas soluções numa solução híbrida. Com esta nova solução consegue-se uma solução híbrida diretamente, já para não dizer que com esta solução resolve-se não só o problema dos valores antigos desatualizados, como também não necessita que quando entrem novos nós se pare a geração de novas células, problema apresentado em 4.2. De facto até convém continuar a gerar novas células para que o novo nó receba blocos de informação dos vizinhos e esteja rapidamente apto para que possa também estimar o tamanho da rede, sem que interfira no normal funcionamento do algoritmo.

4.2.2 Média Pesada

A solução encontrada para, facilmente descartar ou tornar menos relevante informação relativa às estimações mais antigas, foi armazenar a informação em blocos e atribuir-lhe diferentes pesos. Tal como anteriormente, o aumento de precisão passa por a cada D rondas adicionar blocos de células ao vetor x. À medida que são adicionados essas células ao vetor x é também adicionado o seu somatório a um vetor de pesos.

Neste momento a estimação não será feita sobre o vetor x mas sim sobre o vetor de pesos. Sabe-se que este vetor reflete os valores que estão no vetor, x mas agora numa representação simplificada. Sabe-se o tamanho K do vetor x e o tamanho dos blocos utilizados para representar a informação no vetor dos pesos. A solução passa por adaptar a fórmula inicial do estimador para uma variante que utilize blocos e atribua pesos a cada bloco, podendo atribuir diferentes pesos aos diferentes blocos tendo apenas que garantir que a soma dos pesos é igual a 1, para preservar o invariante.

Sabendo que a função `SUM` é separável [18], sabe-se também que uma função separável pode ser composta pela somas das várias funções.

Partindo então da fórmula até então usada para obter a estimação:

$$estimation = \frac{K-1}{\sum_{i=1}^{K} x[i]} \tag{4.2}$$

Assume-se que por exemplo para 2 blocos:

$$estimation = \frac{\frac{K-1}{2}}{\sum_{i=1}^{\frac{K}{2}} x[i]} + \frac{\frac{K-1}{2}}{\sum_{i=\frac{K}{2}+1}^{K} x[i]} \tag{4.3}$$

Agora tendo 4 blocos n tem-se, partindo dos valores mais recentes da esquerda para a direita, e assumindo os pesos 1/4 para os presente (P_{Pres}) e 3/4 para o passado (P_{Pres}). Consultando a equação (4.6) atribui-se o peso $\frac{1}{4}$ (4.6a) ao primeiro(mais recente) e $\frac{3}{16}$ (4.6b), $\frac{27}{64}$ (4.6b), $\frac{3}{64}$ (4.6c) aos restantes por ordem:

$$estimation = \frac{\frac{K-1}{4}}{\sum_{i=1}^{\frac{K}{4}} x[i]} + \frac{\frac{3K-1}{16}}{\sum_{i=\frac{K}{4}+1}^{K/2} x[i]} + \frac{\frac{9K-1}{64}}{\sum_{\frac{K}{2}+1}^{\frac{3K}{4}} x[i]} + \frac{\frac{27K-1}{64}}{\sum_{i=\frac{3K}{4}+1}^{K} x[i]} \tag{4.4}$$

Generalizando para blocos de tamanho arbitrário n e peso relativo p_i tem-se:

$$estimation = \sum_{i=0}^{n-1} p_i . \frac{\frac{K-1}{n}}{\sum_{i*\frac{k}{n}}^{(i+1)\frac{K}{n}} x[i]} \tag{4.5}$$

$$\text{com } p_i = \begin{cases} P_{Pres} & \text{Para } i = 0 \quad (4.6a) \\ P_{Pass} * \frac{1}{n^{i-2}} & \text{Para } i = n-1 \quad (4.6b) \\ P_{Pass} * \frac{1}{n^{i-1}} & \text{Para os restantes} \quad (4.6c) \end{cases}$$

garantindo sempre que $\sum_{i=0}^{n-1} p_i = 1$, de forma a preservar o invariante, uma vez que a soma dos pesos atribuídos aos blocos tem que ser igual ao peso atribuído ao vetor x quando utilizado como único bloco na estimação. A distribuição dos p_i, utilizada em 4.2.2, não é obrigatória, apenas foi a utilizada

pelo algoritmo para se conseguir dar mais relevância ao presente ou passado bastante apenas definir os 2 valores desejados, preservando o invariante, sem ter de definir um peso específico para cada elemento do bloco. Como se trata de uma estimação aleatório o tamanho do vetor sobre o qual vai ser efetuada a estimação é variável, daí se ter tomado esta opção.

Com este novo mecanismo a solução consegue-se aumentar precisão e adaptabilidade sem ter de estar a alternar entre modos.

Tendo chegado a esta fórmula o passo seguinte foi pensar na atribuição de pesos a cada bloco, fazendo distinção entre os blocos mais recentes, com informação mais credível, e os blocos mais antigos mas ditos mais estáveis.

Foram feitas várias experiências com diferentes valores, atribuindo quer mais peso aos novos nós como por outro lado atribuir um peso maior aos mais antigos, com diferentes resultados. No entanto, nenhuns dos resultados atingidos foram considerados tão satisfatórios como inicialmente previsto porque existe um *trade-off* entre atribuir mais ou menos peso ao presente ou ao passado já que não se adaptam a entradas e saídas da mesma forma. No caso de se atribuir maior peso ao passado prima-se por uma maior precisão ao longo do tempo e maior estabilidade, mas atenuando a convergência em caso de *churn*. Em contrário no caso de atribuir maior peso ao presente, prima-se por uma maior convergência em caso de *churn* mas perde-se precisão em casos de estabilidade, além disso está sujeito a maiores picos de variação devido ao maior peso dado ao presente.

A solução poderia passar por obter um mecanismo que fizesse essa alternância entre precisão e adaptabilidade de forma automática, consoante as necessidades.

Capítulo 5

Avaliação

Este capítulo é dedicado à avaliação e discussão das diversas soluções encontradas. Antes de serem introduzidas as soluções, os respetivos resultados e a forma como estes foram alcançados é dedicada uma secção para identificar as tecnologias utilizadas para a obtenção das soluções.

5.1 Tecnologias utilizadas

5.1.1 *Python*

A escolha da linguagem de programação para implementar os algoritmos propostos recaiu na linguagem *python* por ser uma linguagem de rápida aprendizagem e por ainda disponibilizar uma biblioteca sobre grafos, com implementações de diversas topologias e de métodos úteis para trabalhar com este tipo estruturas, permitindo agilizar o processo, já que não foi necessário despender tempo na implementação de tais algoritmos. Houve apenas um investimento inicial para estudar a biblioteca e perceber que tipo de soluções disponibilizava e se estas poderiam vir a ser úteis. Como de facto vieram a ser úteis, foi utilizada. A biblioteca em questão é *NetworkX*. Para além desta biblioteca foi relevante também o uso da biblioteca *matplotlib* para a obtenção da visualização gráfica do grafo nas diferentes topologias.

5.1.2 *RStudio*

Após o desenho dos vários algoritmos e consequente aplicação dos mesmos em vários testes houve uma grande quantidade de dados para serem tratados. A ferramenta escolhida para esse tratamento de dados foi o IDE (Integrated development environment) *RStudio* por ser versátil e permitir, de forma rápida, tratar os dados para obter métricas de comparação ou para desenho de gráficos. A utilização deste IDE foi de facto importante, já que todos os resultados e gráficos subjacentes apresentados neste capítulo de avaliação, foram tratados com o recurso ao *RStudio*.

5.1.3 Topologias de rede utilizadas

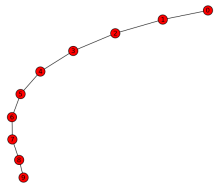
Durante a realização dos testes para o algoritmo desenvolvido foram utilizadas várias topologias de rede com diferentes propriedades, para analisar o comportamento do algoritmo em ambientes distintos. Nesta secção são apresentadas as várias topologias utilizadas.

5.1.3.1 Linha

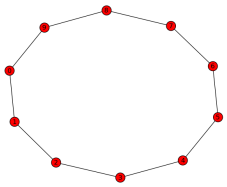
A topologia em linha é uma topologia, que como o nome indica tem a forma de uma linha. Cada nó tem como vizinhos apenas o nó anterior e sucessor na linha, menos o primeiro e último nó que apenas têm um vizinho. O diâmetro da rede é igual ao tamanho da rede.

5.1.3.2 Ciclo

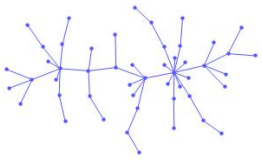
A topologia em ciclo é idêntica à topologia em linha, difere apenas na conectividade dos nós das extremidades. Na prática é uma topologia em linha, em que o primeiro e último nó estão diretamente conectados e por consequência são vizinhos, contrariamente ao que acontece na topologia em linha. Nesta topologia o diâmetro é igual a metade do número de nós que compõem a rede, esta redução para metade do diâmetro da rede é uma



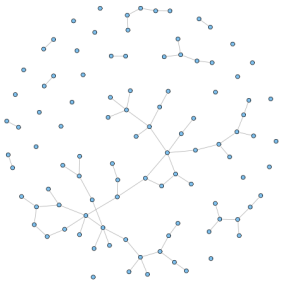
(a) Topologia de rede em linha



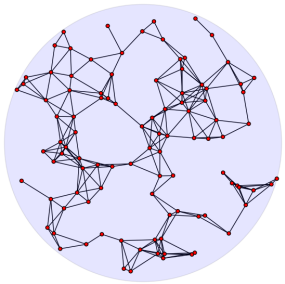
(b) Topologia de rede em ciclo



(c) Topologia de rede *Barabási-Albert*



(d) Topologia de rede *Erdős-Rényi*



(e) Topologia de rede Random-Geometric

Figura 5.1: Topologias de rede

consequência da transformação de linha em ciclo e pelo "primeiro" e "último" nós estarem conectados.

5.1.3.3 *Barabási-Albert*

A topologia de rede assente no modelo de *Barabási-Albert* pode ser vista, de uma forma simplificada, como uma topologia de n nós, em que há medida que vão sendo acrescentados nós, estes vão estabelecer ligações a um número definido de nós, com maior probabilidade de serem anexados a nós que já tenham um elevado grau de conectividade (muitas ligações a outros nós), como se pode ver na figura 5.1c. É um modelo aleatório que tende a ter zonas densamente povoadas, em contraste com outras zonas mais isoladas. A composição da rede depende sempre de quão elevado é o número de ligações para cada nó em comparação com o tamanho da rede, se for relativamente grande, o diâmetro da rede é pequeno.

5.1.3.4 *Erdős-Rényi*

Tal como o modelo *Barabási-Albert*, visto anteriormente, a topologia em *Erdős-Rényi* também assenta num modelo aleatório. Neste caso para cada nó existe uma probabilidade p de ser criada uma aresta para os restantes nós. Se esta probabilidade for relativamente grande o grafo associado vai ter muitas ligações e o diâmetro da rede vai ser pequeno, caso contrário há poucas ligações e o diâmetro vai ser grande.

5.1.3.5 *Random Geometric*

Cada nó é associado a um ponto no referencial cartesiano através de uma distribuição uniforme. É estabelecido um valor r para o raio de alcance de cada ponto, os pontos que estiverem ao alcance do raio são ligados através de uma aresta. É também uma topologia aleatória, na qual o diâmetro depende do tamanho do raio de alcance dos pontos.

5.2 Ambiente estático

Para testar a implementação do algoritmo desenvolvido para ambientes estáticos foram realizados vários testes ao algoritmo com diferentes topologias de redes previamente apresentadas, *Random Geometric*, em linha, ciclo, *Barabási-Albert* e *Erdős-Rényi*.

Inicialmente pensou-se em apresentar um gráfico que mostrasse numa linha o tamanho da rede real e a estimação, e que ambas serviriam para comparar os resultados entre as várias topologias. Devido à natureza estática das redes, por definição, os resultados dos algoritmos são muito bons, ou seja muito próximos do valor real expectado e por isso à vista desarmada não havia grande diferença entre topologias. Por isso para além desse gráfico foi necessário encontrar outra métrica que fosse mais esclarecedora e que revelasse a aproximação da estimação ao resultado correto, foi utilizada a *normalized root-mean-square deviation*, explicada na secção 5.2.1 . Após a secção dedicado à introdução da NRMSD são analisados individualmente os

resultados obtidos para cada topologia a que se recorreu para realizar este estudo.

5.2.1 *Root-Mean-Square Deviation*

A *root-mean-square deviation* (RMSD) deriva da *root-mean-square* (RMS), em português 'valor quadrático médio' ou ainda 'valor eficaz' que calcula a média quadrática. A RMSD é utilizada para calcular a diferença entre os valores estimados e o resultado. É uma boa medida para calcular a eficácia do estimador.

Para cada valor estimado é calculado o seu resíduo. Este resíduo consiste no quadrado da diferença entre a estimativa, designado por $\widehat{x}$, e o resultado correto x. É calculado o somatório dos resíduos e dividido pelo número de amostras.

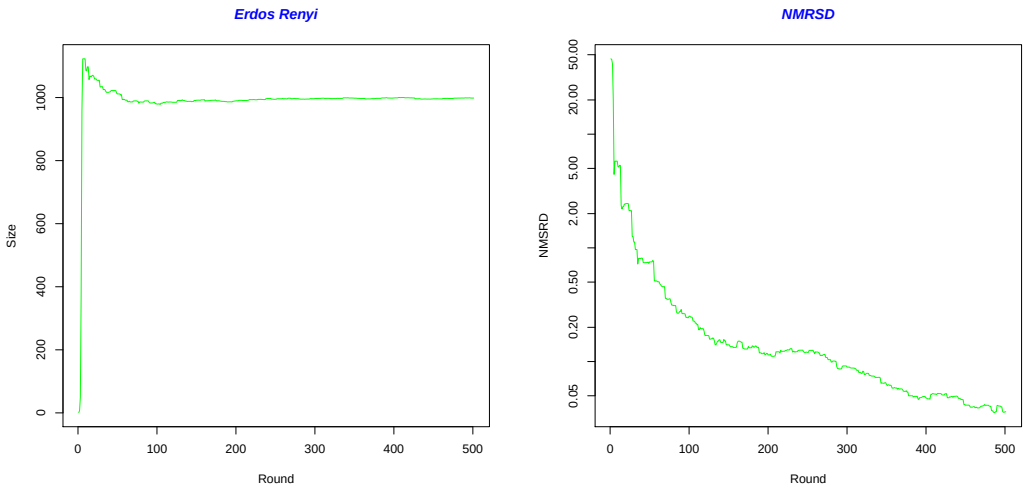
$$RMSD = \sqrt{\frac{\sum_{t=1}^{n}(x_t - \widehat{x}_t)^2}{n}} \tag{5.1}$$

5.2.2 Análise de resultados

5.2.2.1 Configuração dos testes

Os testes realizados foram uniformizados para que possam comparar eficientemente as topologias testadas. Para cada topologia foram corridas várias experiências diferentes com grafos de 1000 nós, cada uma com 500 iterações. O bloco x sobre o qual vão ser calculadas as estimações tem inicialmente 50 (k) células, e a cada D (diâmetro), que varia entre 6 e 7, rondas são gerados mais k células para juntar ao bloco x. São realizados várias experiências de forma a obter resultados mais fiáveis do que se forem executadas experiências isoladamente. Desta forma, os resultados das várias experiências são acumulados e por fim é calculada a sua média, que é o valor final apresentado como estimativa para o tamanho da rede. Os valores de RMSD apresentados são calculados a partir desta média acumulada.

5.2.2.2 *Erdős-Rényi*



(a) Estimativa do tamanho de rede (b) RMSD

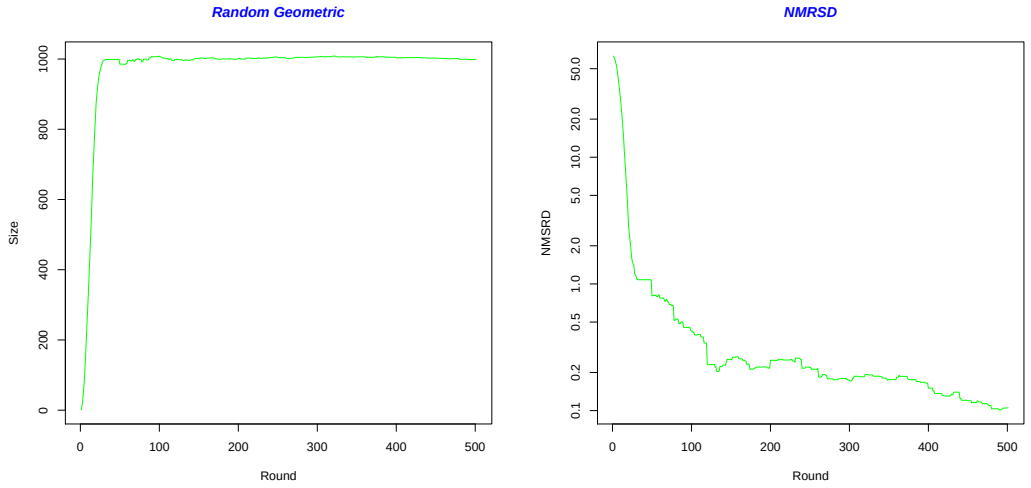
Figura 5.2: Resultados para uma rede *Erdős-Rényi* com 1000 nós.

Como se pode ver analisando a figura 5.2a, as estimações iniciais não só convergem muito rapidamente até ao valor como ultrapassam consideravelmente o resultado esperado. No entanto com o passar do tempo a estimativa baixa consideravelmente situando-se relativamente perto do valor esperado (1000), como se pode ver analisando agora a figura 5.2b, o valor dos resíduos vão diminuindo ao longo da execução do algoritmo ainda que de forma algo lenta. É ainda relevante notar que apesar de a aproximação, em muitos pontos do algoritmo, estar um pouco desfasada da realidade, não é expectável que este algoritmo seja exato, já que está sempre prevista uma percentagem de erro de aproximação que por muito pequeno que seja, no caso de uma rede grande, a diferença entre resultado ótimo e estimação pode estar na ordem das dezenas.

5.2.2.3 *Random Geometric*

Analisando os gráficos de estimação do tamanho da rede verifica-se que o comportamento da topologia *Random Geometric* difere bastante do com-

portamento da topologia *Erdős-Rényi*, é relativamente mais lento a chegar a valores próximos do valor ótimo. Em relação à qualidade do estimador a partir do *setup* inicial do algoritmo pela figura 5.3a nada mais é possível concluir. Agora olhando para a figura 5.3b e para a figura 5.2b pode-se ver que os resultados estimados aproximam-se menos do resultado correto que em comparação com a rede *Erdős-Rényi* no entanto ainda podem convergir mais.



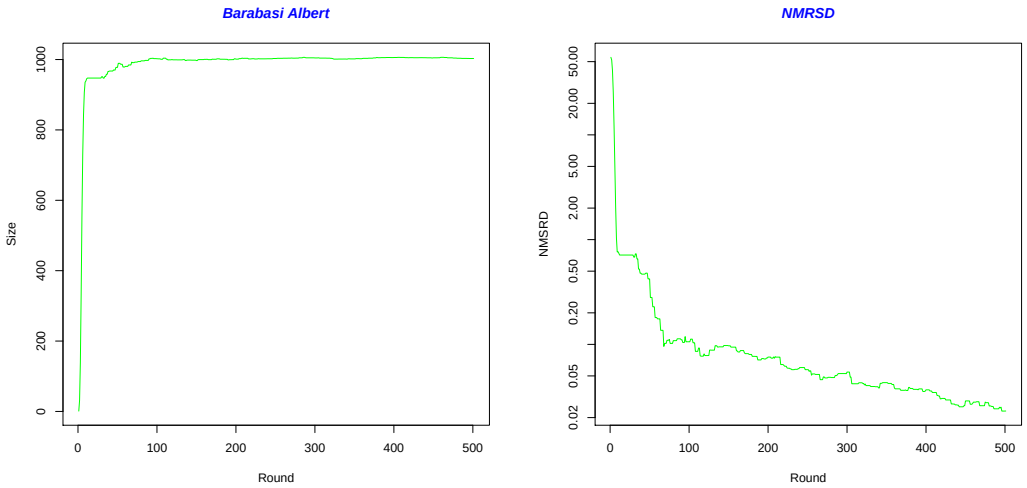
(a) Estimativa do tamanho de rede (b) RMSD

Figura 5.3: Resultados para uma rede *Random Geometric* com 1000 nós.

5.2.2.4 *Barabási-Albert*

A rede em grafo *Barabási-Albert* apresenta um comportamento muito similar com o da rede com grafo *Random Geometric* se se compararem gráficos dos valores estimados. Analisando como mais pormenor os valores dos resíduos de ambos os grafos, 5.4a e 5.4b , consegue-se ver que o grafo *Barábasi-Albert* converge mais rapidamente. Apesar de a curva dos valores estimados para o tamanho da rede se assemelhar mais ao grafo *Random Geometric* a rapidez com que a curva de convergência dos resíduos deste grafo se aproxima de 0 assemelha-se mais com a curva correspondente ao grafo *Erdős-Rényi*.

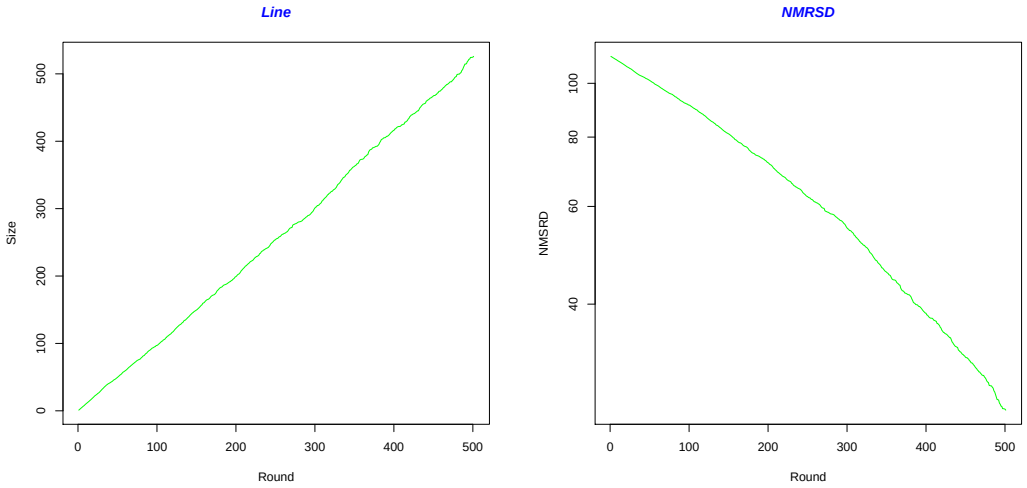
Apesar de tudo, não converge tão bem como o referido grafo.



(a) Estimativa do tamanho de rede (b) RMSD

Figura 5.4: Resultados para uma rede Barabási-Albert com 1000 nós.

5.2.2.5 Linha



(a) Estimativa do tamanho de rede (b) RMSD

Figura 5.5: Resultados para uma rede em Linha com 1000 nós.

A diferença entre a topologia em linha e as restantes topologia analisadas e discutidas é notória à primeira vista. Enquanto que, para outros exemplos, existe uma curva de aproximação que converge muito rapidamente para um valor próximo do resultado correto a estimar e por fim há uma aproximação mais lenta mas mais precisa, como se pode ver nas figuras 5.4a e 5.3a, neste caso a aproximação é muito lenta com uma curva de aproximação similar a uma reta de 45° de declive, observando a figura 5.5a. A diferença no comportamento das curvas de aproximação é facilmente explicável, enquanto que nas outras redes o diâmetro da rede é pequeno, na ordem das dezenas, neste caso, como o grafo é em linha, o diâmetro da rede é igual ao seu número de nós, 1000. Sabe-se pela secção 5.2.2.1 que o tamanho da rede é 1000 e no entanto o estimador só apresenta no final das 500 iterações o valor 500. Isto acontece porque a precisão do estimador é limitada pelo diâmetro da rede como referido em 2.2.3, ou seja só com 1000 iterações do algoritmo é que a estimativa iria convergir para valores na ordem das mil unidades.

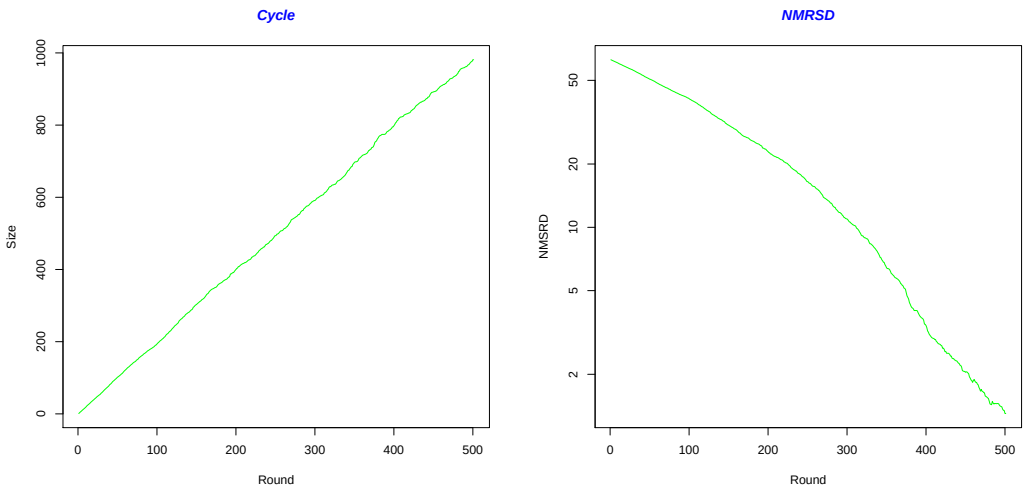
Os valores apresentados correspondem aos valores extraídos de uma das extremidades do grafo, tal como nos restantes grafos. Poder-se-ia pensar que, se se extraísse informação de nó situado no meio da rede, este conseguiria obter resultados em metade das iterações por receber informação das duas extremidades mais rapidamente mas, no entanto, é limitado pelo diâmetro do grafo para gerar e receber novos valores dos restantes nós. Foi escolhido um valor nas extremidades como poderia ter sido outro valor qualquer, como a técnica depende do diâmetro da rede, é indiferente o nó que se escolha. Como se dá tempo ao estimador para estabilizar e que todos os nós recebam a informação trocada, estes vão ter todos o mesmo valor estimado, por isso não é uma questão central nesta discussão.

Apesar de a técnica testada não apresentar um comportamento adequado para este tipo de topologia, por outro lado, em redes grandes esta topologia não é utilizada por razões óbvias. Para propagar informação entre extremidades da rede tem de haver D comunicações, com D o diâmetro da rede, o que não é viável. Está técnica é portanto utilizável em topologia mais

comummente utilizadas para redes de larga escala.

5.2.2.6 *Ciclo*

Olhando para os resultados da topologia em ciclo estes parecem idênticos aos anteriormente analisados para a topologia em linha. E realmente são muito parecidos já as topologias diferem pouco uma da outra. Enquanto que uma é uma linha, a outra é um círculo. Se a topologia em linha demora n rondas a propagar informação de um nó para os restantes, com esta topologia esse tempo de propagação reduz-se para metade. Assim sendo leva $n/2$ rondas a propagar informação. Daí os gráficos serem idênticos, as curvas são idênticas os valores estimados são o dobro enquanto que os valores dos resíduos calculados são naturalmente menores.



(a) Estimativa do tamanho de rede (b) RMSD

Figura 5.6: Resultados para uma rede em Ciclo com 1000 nós.

5.3 Dinâmico

Para realizar a avaliação do algoritmo desenvolvido em ambiente dinâmico foram descartadas à partida duas topologias, a topologia em linha e em

ciclo, porque já se viu, em 5.2.2.5, que estas topologias não se adequam bem a este tipo de técnicas para ambientes estáticos, logo pode-se inferir que o seu comportamento ainda vai piorar para ambientes dinâmicos. Foram utilizadas as 2 topologias que melhores resultados apresentaram e cujos grafos se consideraram mais interessantes para o estudo, a topologia *Erdős-Rényi* e *Random Geometric*.

5.3.1 Configuração dos testes

O teste do algoritmo em ambientes dinâmicas é o grande mote para a dissertação desenvolvida, foram realizados testes mais específicos a cada topologia de forma observar o comportamento da mesma quando submetida a diferentes instâncias do algoritmo com diferentes propriedades. Foi observado o comportamento do algoritmo quando o tamanho do bloco do estimador é alterado, os valores do bloco utilizados foram 50, 100 e 200.Na secção 3.3.1 foi introduzido o k típico de 387, mas neste caso está-se a definir o tamanho de um bloco. O vetor, na soma dos blocos será sempre maior que esse k típico para um vetor que não sofre alterações de tamanho. Como agora o vetor x vai sendo sempre aumentado não faz sentido definir um k.

Na figura (4.6) foi definido como é feita a distribuição dos pesos, por uma questão de notação vai se considerar ρ como o peso atribuído ao presente enquanto que o peso atribuído ao passado será $1-\rho$ (para preservar o invariante), como valores a serem introduzidos para calcular os p_i finais. Ao mesmo tempo que foram testados vários tamanhos de bloco foi também testado o comportamento do estimador caso o parâmetro ρ fosse apertado ou alargado. Foram utilizados 3 valores diferentes para o ρ que dão respetivamente 5% 10% e 20% de peso ao bloco mais recente.

Aquando da realização dos testes, para poder inferir sobre o comportamento do algoritmo em ambiente estático, foram efetuadas um número considerável de experiências com 500 iterações cada. Para o ambiente dinâmico foram utilizadas as mesmas 30 experiências mas agora o número de iterações foi aumentado para 1000, devido ao caráter dinâmico das experiências

efetuadas, para dar tempo ao algoritmo de estabilizar e assim poder melhor aferir os resultados obtidos. Foi utilizado esse número de experiências com o intuito de tornar mais uniformes os resultados obtidos, já que estes resultam de um método aleatório e podem variar em termos de comportamento geral do estimador em cada experiência.

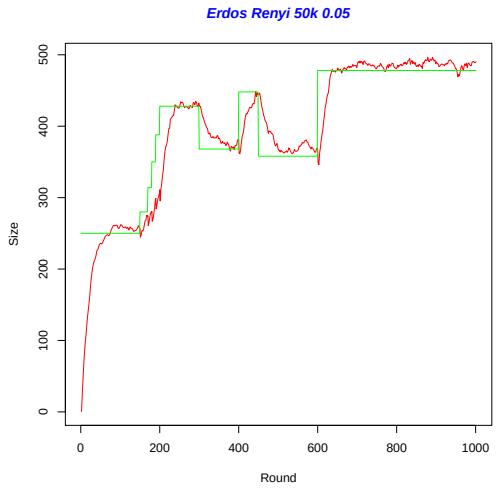
Para que os testes, com diferentes parâmetros como *input* do algoritmo, sejam conclusivos, quer na comparação entre topologias como na comparação de instâncias da mesma topologia mas com propriedades diferentes, foram bem definidos os momentos de atualização de tamanho do grafo e também fixados o número de inserções ou remoções para cada momento. Assim é possível analisar cada gráfico e compará-lo com os restantes.

5.3.2 Análise de Resultados

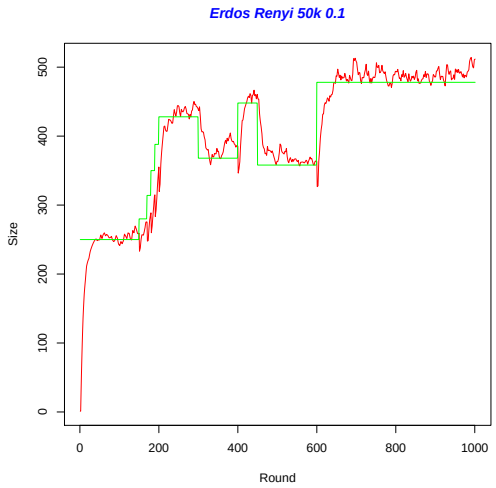
5.3.2.1 *Erdős-Rényi*

Os parâmetros das diferentes experiências apenas variaram no tamanho do bloco e no peso atribuído ao presente. Por forma a estruturar a análise aos resultados obtidos vão ser analisadas começando pelo menor tamanho de bloco e menor peso, aumento gradualmente o peso até aumentar o tamanho do bloco. Cada grupo de figuras tem em comum o mesmo tamanho de bloco, variando de forma crescente valor do peso atribuído ao presente e identificáveis pelas letras a, b e c. É relevante ainda assinalar que para todos os testes relativos a esta topologia o diâmetro da rede utilizado foi de 2 e 3, valores bastante pequenos. Para serem obtidos estes valores para o diâmetro tiveram de ser descartadas redes com diâmetros diferentes.

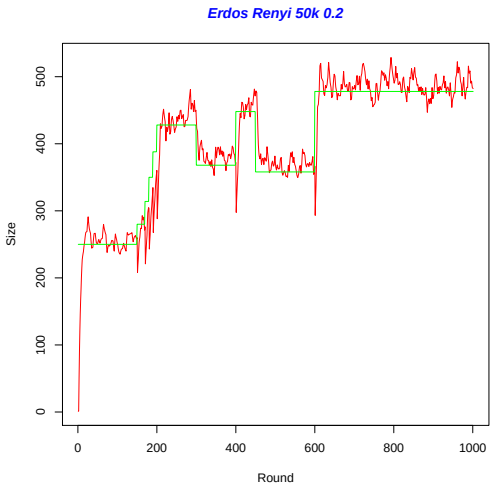
Tendo como ponto de partida então a figura 5.7a com tamanho de bloco 50 e o peso do presente 0.05, como se pode ver pela legenda, pode se ver que o comportamento do estimador, representado pela linha vermelha, é bastante satisfatório, quando comparado com o valor instantâneo representado pela linha verde. O estimador consegue convergir tanto em situações de aumento de número de nós como em remoções de nós, algo que anteriormente não se



(a) Peso 0.05



(b) Peso 0.1

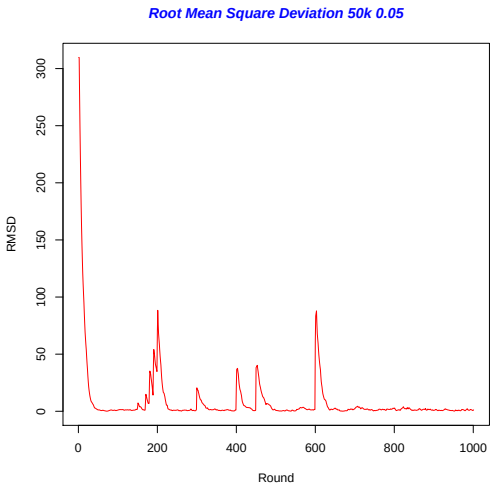


(c) Peso 0.2

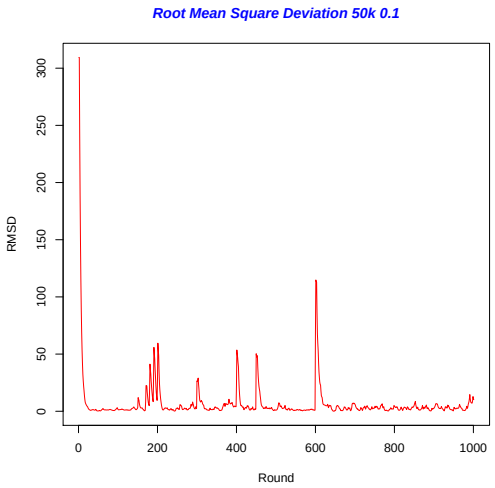
Figura 5.7: Estimadores para uma rede *Erdős-Rényi* de tamanho de bloco 50

havia conseguido.

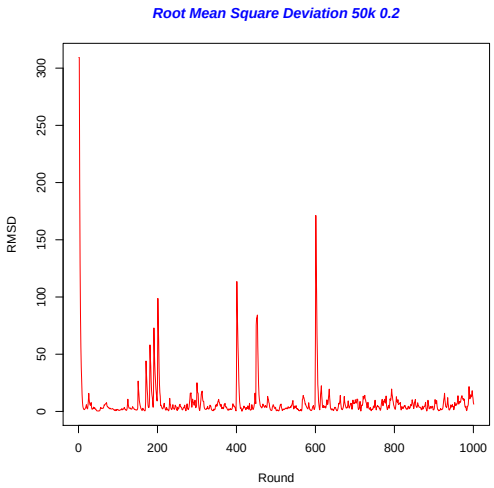
Passando agora para a figura 5.7b com peso 0.1 pode-se reparar numa mudança do comportamento do gráfico. Por um lado a convergência do estimador em situações de *churn* é alcançada mais rapidamente que na figura 5.7a mas por outro lado está mais sujeito a picos de variação em situações



(a) Peso 0.05



(b) Peso 0.1



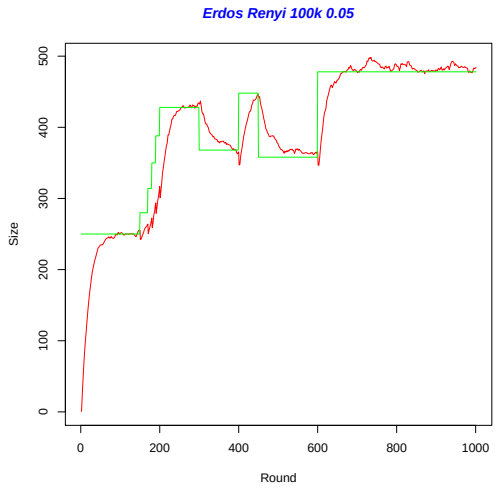
(c) Peso 0.2

Figura 5.8: RMSD numa rede *Erdős-Rényi* de tamanho de bloco 50

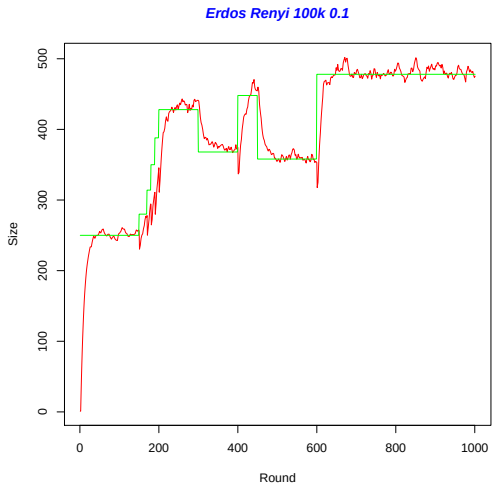
de repouso da rede.

Se se olhar agora para a figura 5.7c. agora com peso igual a 0.2 pode ver-se, de forma ainda mais acentuada, este comportamento mais instável do estimador, que a este nível já apresenta um desvio demasiado grande em relação ao valor ótimo.

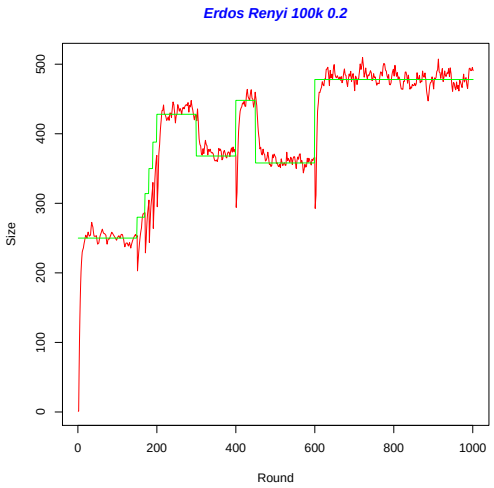
Se se atentar agora na RMSD em 5.8 as diferenças já não são tão notórias.



(a) Peso 0.05



(b) Peso 0.1

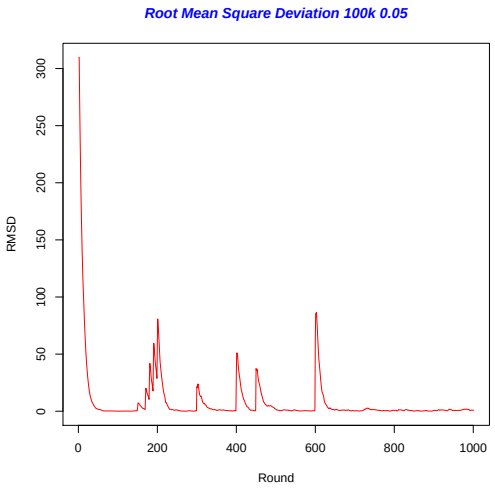


(c) Peso 0.2

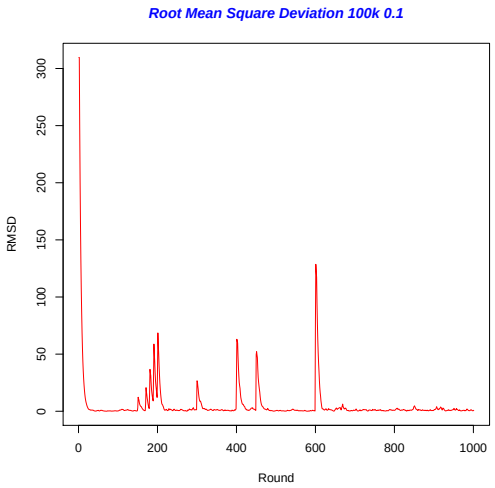
Figura 5.9: Estimadores para uma rede *Erdős-Rényi* de tamanho de bloco 100

Na parte inicial do gráfico a figura 5.8a é mais precisa que as 2 restantes enquanto que na segunda metade do gráfico a balança já pende por vezes para a figura 5.8b, no entanto o primeiro caso mantém-se mais estável.

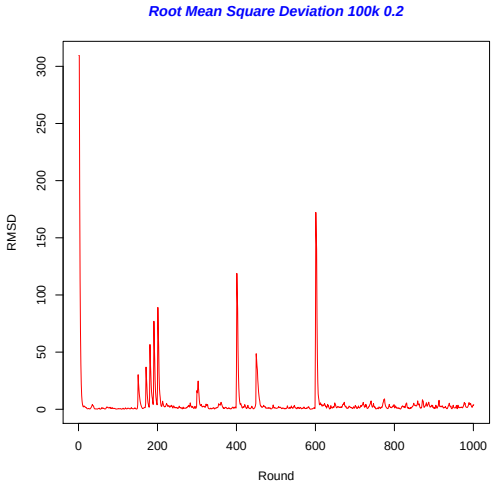
Passando agora à análise ao comportamento ao blocos de tamanho 100, comparando a figura 5.9a com peso 0.05 com a 5.7a, análoga de bloco 50, o



(a) Peso 0.05



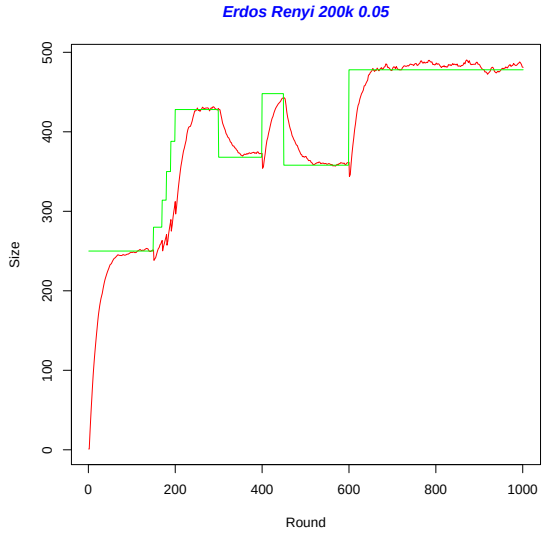
(b) Peso 0.1



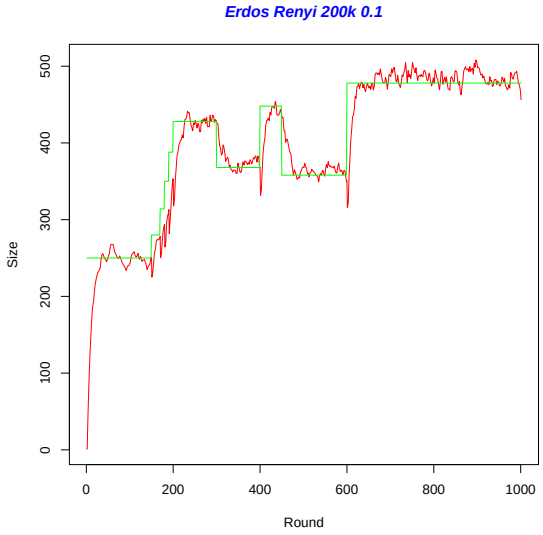
(c) Peso 0.2

Figura 5.10: RMSD numa rede *Erdős-Rényi* de tamanho de bloco 100

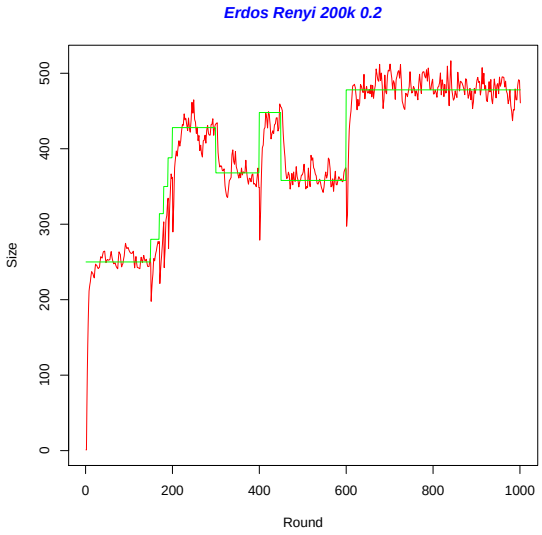
comportamento é similar. Pode se ver agora na presente figura um aproximar ao resultado correto de forma mais rápida mas mantendo-se a curva de forma suave quando a rede estagna. Tal como acontecera com o tamanho de bloco a 50, com o bloco a 100 se se aumentar o peso do presente para 0.1 e 0.2, figuras 5.9b e 5.9c respetivamente, há uma maior convergência em caso de entrada e saída de nós na rede, mas há picos no estimador quando a rede



(a) Peso 0.05



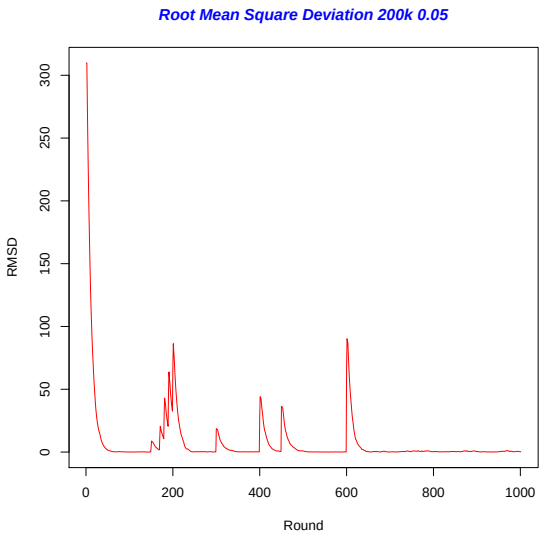
(b) Peso 0.1



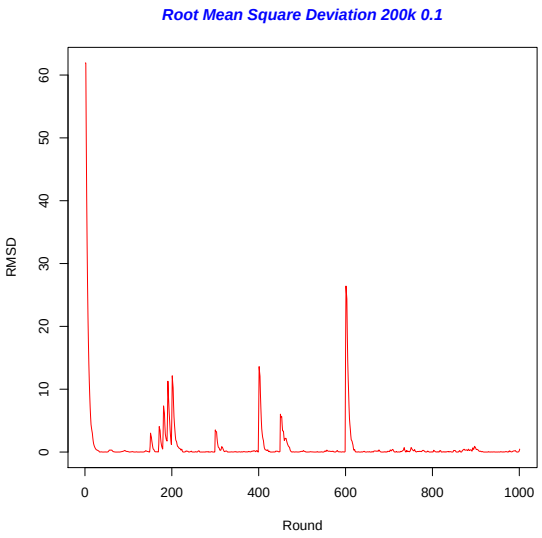
(c) Peso 0.2

Figura 5.11: Estimadores para uma rede *Erdős-Rényi* de tamanho de bloco 200

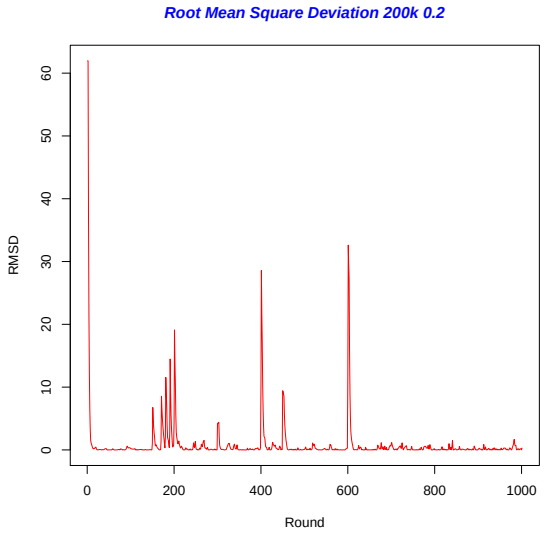
se apresenta estável por se atribuir um maior peso ao presente. No entanto para este tamanho de bloco há menor variância nos picos, ou seja menor diferença em relação ao valor expectável. Para o peso 0.1 também ainda



(a) Peso 0.05



(b) Peso 0.1



(c) Peso 0.2

Figura 5.12: RMSD numa rede *Erdős-Rényi* de tamanho de bloco 200

se pode aceitar o comportamento do estimador, para peso 0.2 o estimador também ainda produz à primeira vista algo instáveis. No entanto, para este bloco, se se observar a terceira figura relativa à análise à RMSD, 5.10c, já se nota um aproximar aos gráficos 5.10a e 5.10b que se comportam de forma

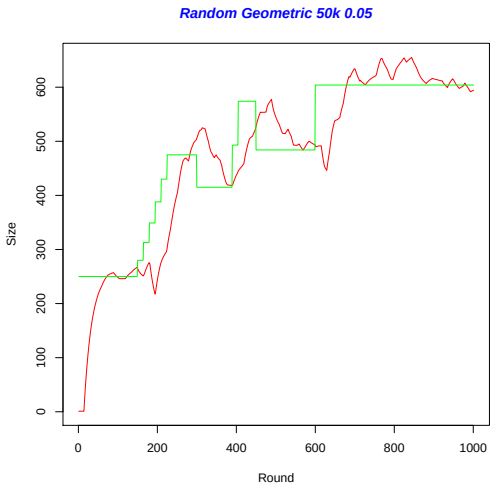
análoga para blocos de tamanho 50..

Observando agora para o tamanho de bloco a 200 já se pode ver melhorias significativas em todos os pesos, para 0.05 na (5.11a) continua a apresentar um comportamento bom mas ainda melhor que nas anteriores versões. Para 0.1, na figura (5.11b), já é bastante aceitável neste caso, já que se nota mais suavidade na evolução do gráfico do estimador em momentos de maior acalmia. A grande diferença para este bloco é que, para o peso 0.2 (5.11c), o gráfico deixa de se comportar tão irregularmente, como aconteceu nos casos anteriores, para poder ser utilizado caso se ache conveniente. Esta evolução é acompanhada pelo RMSD, desta vez para o peso 0.1, figura 5.12a fica para trás em relação ao peso 5.12b que apresenta o melhor comportamento global e em relação ao surgimento do bloco de tamanho 200 (5.11c) que é quase tão bom como para o de tamanho 100.

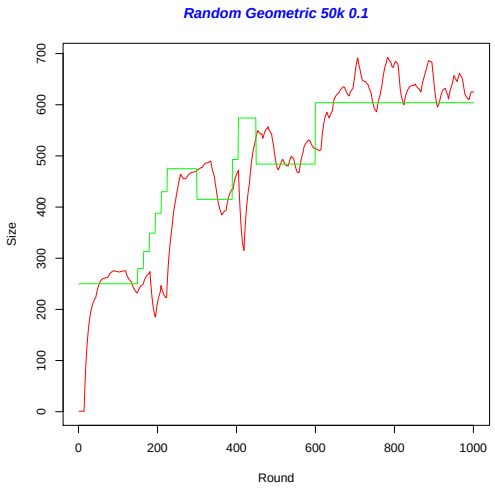
5.3.2.2 *Random Geometric*

Após terminada a análise à topologia *Erdős-Rényi*, vai se passar agora à análise da topologia *random geometric*. De uma forma análoga à secção 5.3.2.1, os parâmetros das experiências realizadas apenas variaram apenas no tamanho do bloco e no peso atribuído ao presente. Foram estruturados os resultados obtidos por ordem crescente de tamanho de bloco e peso, respetivamente. Os grupos de figuras tem em comum o mesmo tamanho de bloco, variando de forma crescente o peso atribuído ao presente e identificáveis pelas letras a, b e c. Desta vez de forma a diversificar os resultados foi utilizado o valor base de 15 como diâmetro, para poder retirar conclusões sobre o comportamento do algoritmo numa rede com os nós mais distantes entre si. Tal como foi referido em 5.3.2.1, os valores diferentes ao diâmetro pretendido foram descartados.

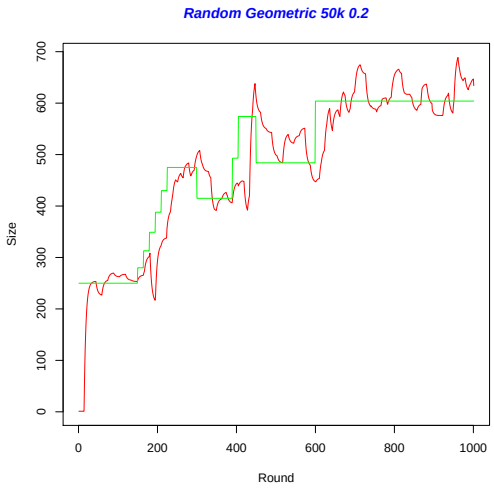
Tendo como ponto de partida então a figura 5.13a com tamanho de bloco 50 e o peso do presente 0.05, como está indicado na legenda, pode se ver que o comportamento do estimador, representado pela linha vermelha, é satisfatório, quando comparado com o valor instantâneo representado pela



(a) Peso 0.05



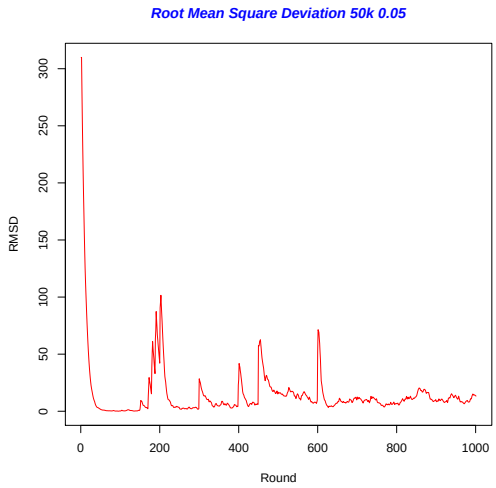
(b) Peso 0.1



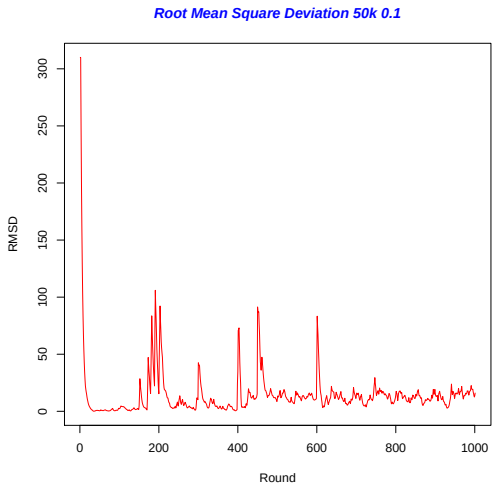
(c) Peso 0.2

Figura 5.13: Estimadores para uma rede *Random Geometric* de tamanho de bloco 50

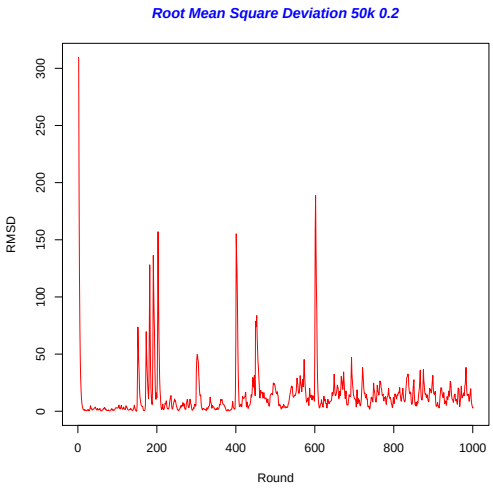
linha verde se se tiver em conta que o diâmetro da rede aumento muito em comparação a análise anterior. No entanto, o comportamento difere do comportamento apresentado pela 5.7a. O estimador leva mais tempo a convergir para os valores ótimos. As restantes figuras referentes à topologia *Random Geometric* também diferem de modo semelhante às homólogas da topologia



(a) Peso 0.05



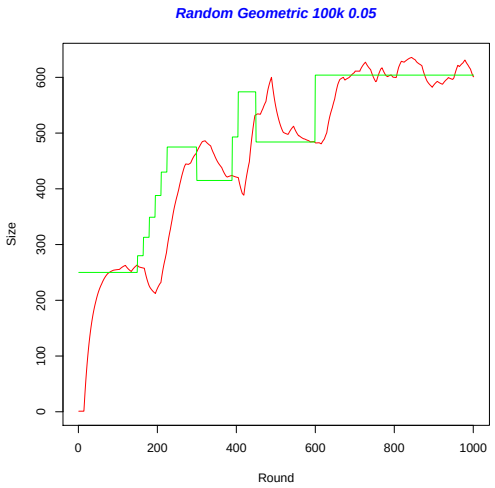
(b) Peso 0.1



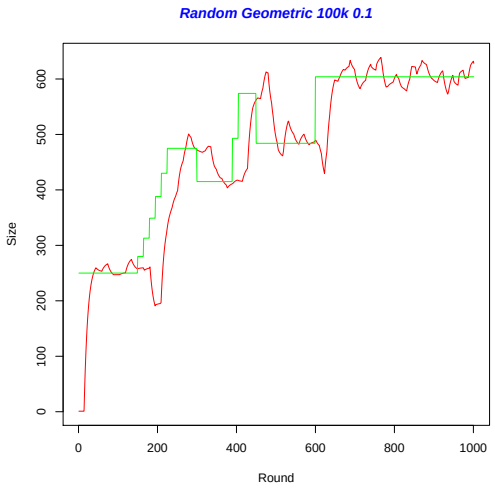
(c) Peso 0.2

Figura 5.14: RMSD numa rede *Random Geometric* de tamanho de bloco 50

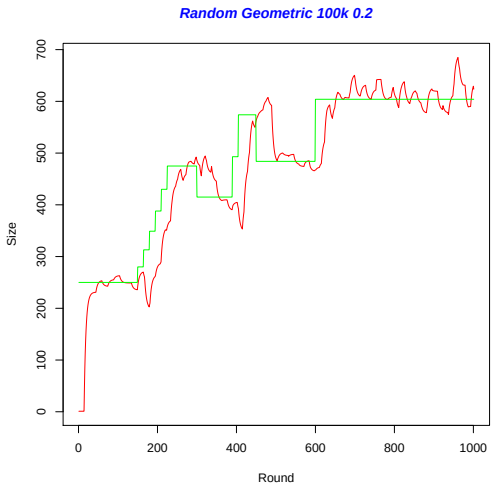
Erdős-Rényi, com relativo atraso devido ao aumento do diâmetro da rede, que passou de sensivelmente 2 ou 3 para um diâmetro entre 12 e 15. O diâmetro parte do valor 15 e pode vir a diminuir à medida que vão entrando nós, sendo pouco provável que seja inferior a 12. Seria interessante apresentar o padrão de evolução do diâmetro, no entanto como se trata de uma experiência aleatória, os valores variam de experiência para experiência. Os



(a) Peso 0.05



(b) Peso 0.1

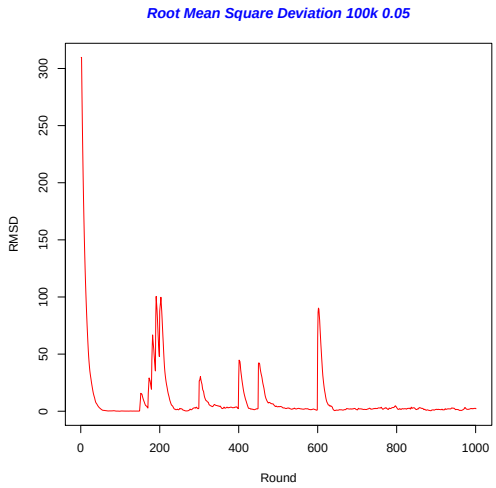


(c) Peso 0.2

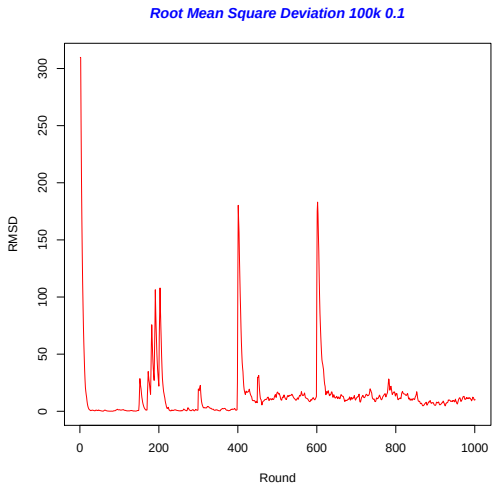
Figura 5.15: Estimadores para uma rede *Random Geometric* de tamanho de bloco 100

diâmetros não são constantes, como se trata de uma simulação dinâmica, a rede expande e retrai de forma aleatória e à medida que vai evoluindo pode ou não o diâmetro acompanhar essa evolução.

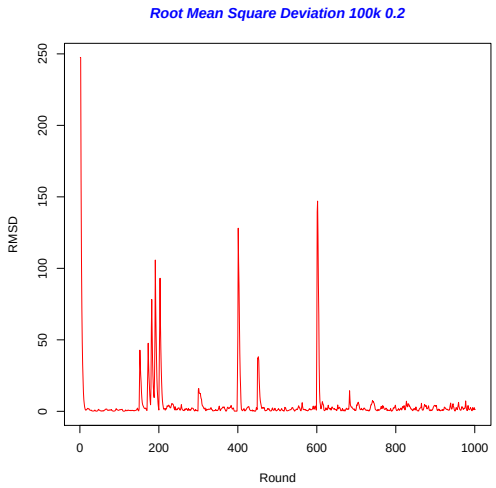
Neste caso como as diferenças são bastante notórias não é necessário conferir os valores RMSD. A análise de RMSD poderá ser antes utilizada para



(a) Peso 0.05



(b) Peso 0.1

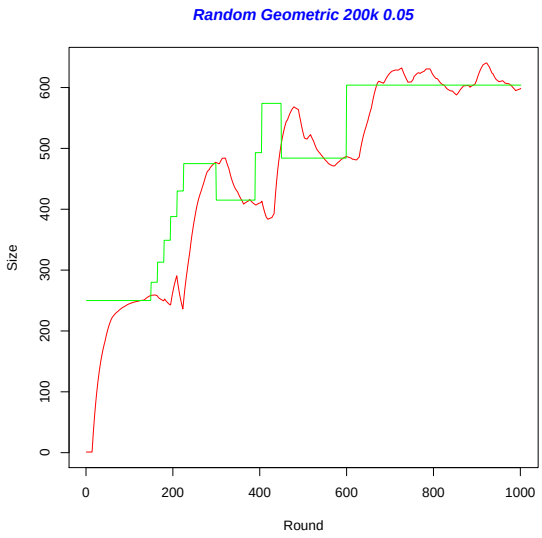


(c) Peso 0.2

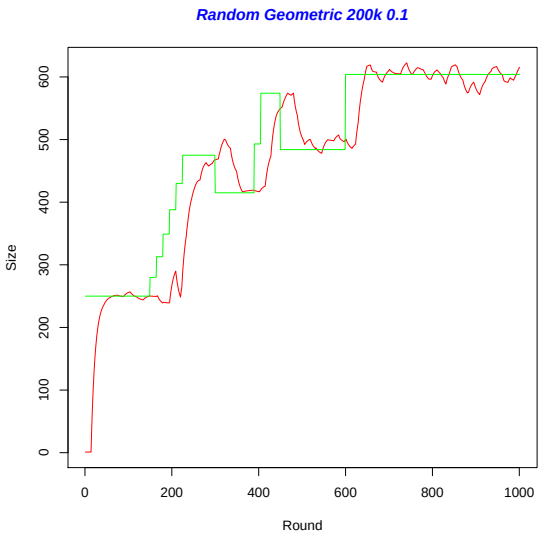
Figura 5.16: RMSD numa para uma rede *Random Geometric* de tamanho de bloco 100

comparar as diferenças entre testes diferentes na topologia *random geometric*.

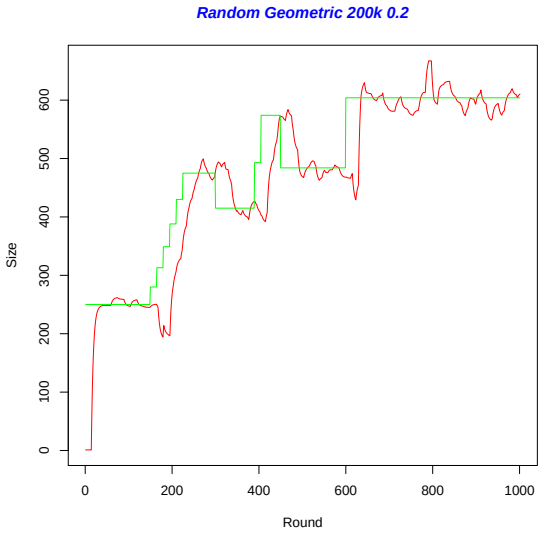
As diferenças entre experiências com o mesmo tamanho de bloco mas com pesos diferentes para esta nova topologia são semelhantes às do mesmo caso da topologia anterior. Com peso mais pequeno o estimador é muito mais estável enquanto que, quando o peso é maior, o mesmo é muito mais reativo.



(a) Peso 0.05



(b) Peso 0.1

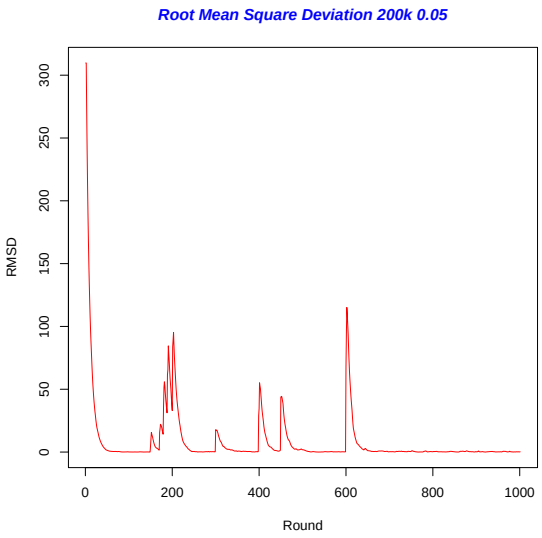


(c) Peso 0.2

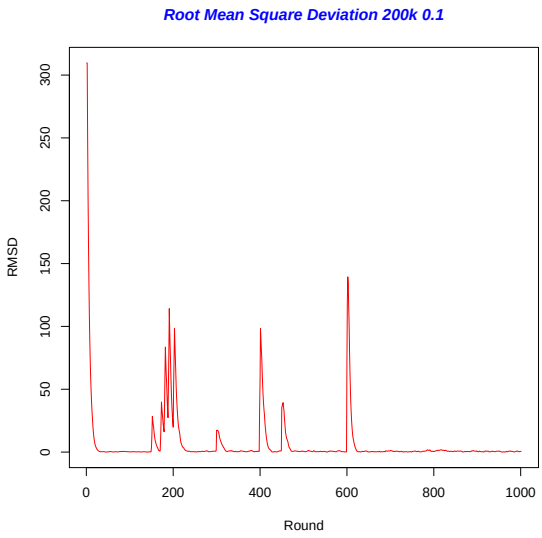
Figura 5.17: Estimadores para uma rede *Random Geometric* de tamanho de bloco 200

Tem-se então peso 0.05 o mais estável, 0.2 o mais reativo e 0.1 o intermédio.

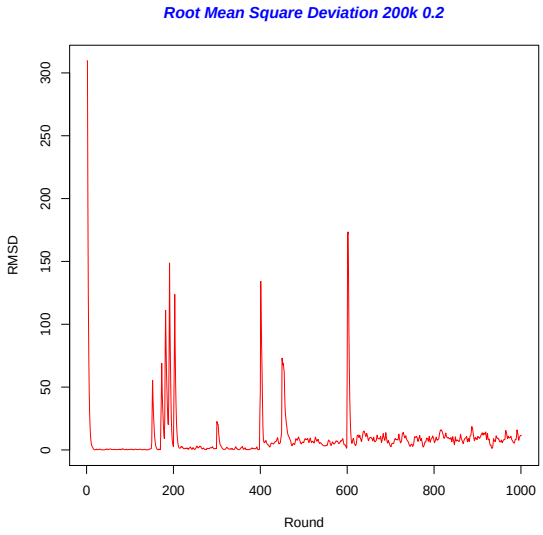
O comportamento do estimador é coerente com o anteriormente analisado com maior nível de detalhe, nesta experiência o que mais salta à vista são



(a) Peso 0.05



(b) Peso 0.1



(c) Peso 0.2

Figura 5.18: RMSD numa rede *Random Geometric* de tamanho de bloco 200

os efeitos atenuadores provocados pelo aumento do diâmetro. Ainda assim o estimador acompanha a evolução da rede, por isso o seu comportamento é satisfatório.

Da mesma forma, com aumento do tamanho do bloco também se verifica

uma convergência maior para o caso de blocos maiores mas acentuando mais os picos de variação, no entanto tanto neste caso como para o fenómeno relativo à influência do peso este comportamento é em parte atenuado pelo aumento do diâmetro da rede.

5.3.2.3 Conclusões

Tendo em conta as topologias utilizadas, os diferentes parâmetros utilizados e os resultados obtidos o uso desta técnica pode ser utilizada em ambientes de elevado *churn* desde que, estejam devidamente ajustados, porque como foi explicado em 5.3.2.1 e 5.3.2.2 há casos em que a técnica não deve aplicar, quando é atribuído demasiado peso ao presente. Como se pode ver pelos resultados obtidos, à medida que vai atribuindo maior peso ao presente mais oscilações a curva do estimador vai ter, em muitos casos os valores da RMSD não são aceitáveis se o estimador tiver demasiadas oscilações, já que passa a ter um erro percentual bastante grande. Por outro lado se se aumentar o tamanho do bloco esse aumento, acaba por compensar, em parte, o aumento do peso. Só se aconselha portanto o aumento do peso do presente caso se aumente também o tamanho do bloco. No entanto o aumento do tamanho do bloco traz outras implicações, nomeadamente o aumento da computação em cada nó e poderá provocar algum congestionamento da rede, para casos em que possa haver limitações nesse aspeto, como é o caso das redes *WSN*.

Não foram exploradas mais redes porque o que realmente define o comportamento do algoritmo não é a topologia em si, mas sim o diâmetro da rede. Quanto menor for a rede, em termos de diâmetro, melhor será a adaptabilidade da curva do estimador e aumentando a rede a propagação dos valores contidos em cada nó vai ser mais demorada contribuindo para um acompanhamento mais distante do estimador face ao valor ótimo. Foram inicialmente definidos diâmetros pequenos nas redes utilizadas porque tipicamente nas redes de larga escala utilizadas, o diâmetro da rede é relativamente pequeno mas foram também testados diâmetros de rede maiores, que apesar de abrandarem o ritmo do estimador continuaram a produzir resultados

aceitáveis.

Capítulo 6

Conclusão

A agregação de dados é uma ferramenta essencial no desenvolvimento e manutenção dos Sistemas Distruibuídos, no entanto obter esta agregação de dados não é uma tarefa trivial. As técnicas utilizadas acabam sempre por apresentar certas limitações. Foi necessário escolher, de entre as várias opções, aquela que melhor se adaptasse à realidade assumida como ponto de partida do problema, que consiste numa rede tipicamente grande, com comportamento dinâmico, com elevado grau de churn e que ainda oferecesse uma solução rápida sem ser necessariamente muito precisa.

Após esta fase de escolha, e partindo então da técnica de *Extrema Propagation*, a qual lida muito mal com a entrada e a saída de nós, avançou-se para uma solução que se adaptasse a constantes alterações na rede.

O trabalho passou então por adaptar o *EP* para se tornar mais preciso ao longo do tempo bem como para não lidar tão mal com o *churn*. Após a obtenção do novo algoritmo foi necessário aplicá-lo de modo a perceber o seu comportamento. Para tal, este foi testado inicialmente num ambiente estático a fim de analisar o aumento de precisão. Depois de se verificar que o estimador aumentava, de facto, a sua precisão ao longo do tempo, tentou verificar-se se o mesmo acontecia num ambiente dinâmico. Analisou-se o seu comportamento para comprovar esse aumento, quer na entrada quer na saída de nós.

Os resultados obtidos pelo novo estimador vieram a revelar-se bastante satisfatórios, já que analisando o seu gráfico de comportamento, se nota claramente um acompanhamento da linha do estimador à evolução da rede, quando analisados os melhores casos. Em suma, conclui-se portanto que têm de ser bem analisados e escolhidos os valores para os parâmetros ρ e para o tamanho do bloco para a execução do algoritmo, com o objetivo de se atingir bons resultados.

O bom comportamento do estimador implementado veio demonstrar que é possível haver alguns avanços nesta área partindo do trabalho desenvolvido até então. Dotar a técnica de um mecanismo que consiga determinar, através dos valores estimados no momento, se a rede poderá estar a aumentar ou a diminuir (e a partir daí decidir se se atribuirá mais ou menos peso aos valores presentes), seria um exemplo de um ponto importante a aprofundar no futuro. Por exemplo, poderia utilizar-se a derivada discreta sobre os últimos valores trocados para assim perceber através dos valores obtidos se a rede poderá estar a aumentar ou a diminuir.

SD	**S**istema **D**istribuído
AD	**A**gregação de **D**ados
EP	**E**xtrema **P**ropagation
RMSD	**R**oot **M**ean **S**quare **D**eviation
IDE	**I**ntegrated **D**evelopment **E**nvironment

Bibliografia

[1] Carlos Baquero, Paulo S. Almeida, and Raquel 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.

[2] Carlos Baquero, Paulo S. Almeida, Raquel Menezes, and Paulo Jesus. Extrema propagation: Fast distributed estimation of sums and network sizes. *IEEE Transactions on Parallel and Distributed Systems*, 23(4):668–675, April 2012.

[3] Mayank Bawa, Hector Garcia-Molina, Aristides Gionis, and Rajeev Motwani. Estimating aggregates on a peer-to-peer network. Technical Report 2003-24, Stanford InfoLab, April 2003.

[4] Jaime Cardoso, Carlos Baquero, and Paulo S. Almeida. Probabilistic estimation of network size and diameter. In *Dependable Computing, 2009. LADC '09. Fourth Latin-American Symposium on*, pages 33–40, sept. 2009.

[5] Laukik Chitnis, Alin Dobra, and Sanjay Ranka. Aggregation methods for large-scale sensor networks. *ACM Transactions on Sensor Networks*, 4(2):1–29, April 2008.

[6] Jeffrey Considine, Feifei Li, George Kollios, , and John Byers. Approximate aggregation techniques for sensor databases. In *Proceedings of the 20th International Conference on Data Engineering*, pages 449 – 460, march-2 april 2004.

[7] Yao-Chung Fan and Arbee L.P. Chen. Efficient and robust sensor data aggregation using linear counting sketches. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–12, 2008.

[8] Ayalvadi J. Ganesh, Anne-Marie Kermarrec, and Laurent Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2):139–149, February 2003.

[9] Keren Horowitz and Dahlia Malkhi. Estimating network size from local information. *Information Processing Letters*, 88:237–243, 2003.

[10] Paulo Jesus, Carlos Baquero, and Paulo S. Almeida. Fault-tolerant aggregation for dynamic networks. In *Reliable Distributed Systems, 2010 29th IEEE Symposium on*, pages 37–43, 2010.

[11] Paulo Jesus, Carlos Baquero, and Paulo Sérgio Almeida. Fault-tolerant aggregation by flow updating. In *9th IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 73–86, 2009.

[12] Paulo Jesus, Carlos Baquero, and Paulo Sérgio Almeida. A survey of distributed data aggregation algorithms. *Computing Research Repository*, abs/1110.0725, 2011.

[13] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual*

IEEE Symposium on Foundations of Computer Science, FOCS '03, pages 482–491, Washington, DC, USA, 2003. IEEE Computer Society.

[14] Dionysios Kostoulas, Dimitrios Psaltoulis, Indranil Gupta, Ken Birman, and Alan J. Demers. Decentralized Schemes for Size Estimation in Large and Dynamic Groups. In *Network Computing and Applications*, pages 41–48, 2005.

[15] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. *SIGOPS - Operating Systems Review*, 36(SI):131–146, December 2002.

[16] Gurmeet Singh Manku. Routing networks for distributed hash tables. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing*, pages 133–142. ACM, 2003.

[17] Laurent Massoulié, Erwan Le Merrer, Anne-Marie Kermarrec, and Ayalvadi Ganesh. Peer counting and sampling in overlay networks: Random walk methods. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '06, pages 123–132, New York, NY, USA, 2006. ACM.

[18] Damon Mosk-Aoyama and Devavrat Shah. Computing separable functions via gossip. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '06, pages 113–122, New York, NY, USA, 2006. ACM.

[19] André Schiper, Alexander A. Shvartsman, Hakim Weatherspoon, and Ben Y. Zhao, editors. *The Importance of Aggregation*, volume 2584 of *Lecture Notes in Computer Science*. Springer, 2003.

[20] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: New aggregation techniques for sensor networks. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 239–249, New York, NY, USA, 2004. ACM.