Análise de algoritmos distribuídos híbridos para Consenso em rede

Fernando Augusto Medeiros Silva, Ricardo Augusto Rabelo de Oliveira
PPGCC - Programa de Pós Graduação em Ciência da Computação
UFOP - Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brasil
email: fernando.augusto@gmail.com, rrabelo@gmail.com

***Resumo*—Com o crescimento dos dispositivos móveis com capacidade de processamento, cria-se a possibilidade de criação de redes instantâneas para troca de informações de interesse comum. Surge daí a necessidade de se encontrar modelos de computação distribuída que possam gerar informações relevantes com baixo custo de tempo e consequentemente, menor custo energético. Nesse trabalho analizamos técnicas de busca de consenso distribuído com a utilização de *clusters* implementadas e simuladas numa ferramenta chamada DAJ *Distributed Algorithms for Java* e analizadas por complexidade de número de mensagens.**

***Keywords*-algoritmos distribuídos, complexidade, consenso distribuído.**

I. Introdução

Com a popularização do uso de dispositivos móveis pessoais com capacidade de comunicação sem fio, surge a possibilidade de criação de redes temporárias entre dispositivos com proximidade física para atingir objetivos comuns. Um possível cenário para essa utilização seria onde vários turistas, localizados em um ponto de encontro, tem em seu dispositivo uma rota de visitação para vários pontos de interesse. Cada uma dessas rotas é definida como uma sequência de coordenadas cartezianas entre os pontos de interesse do grupo.

Assumindo que um determinado perfil de turistas possui pontos de interesse semelhantes seria vantajoso a troca de informações entre eles para produzir uma rota consensual que inclua as informações trazidas por todos em torno de um objetivo específico. Como objetivo podemos citar por exemplo um grupo de turistas da terceira idade que pode ter como prioridade caminhos com menor distância ou um grupo de estudantes pode preferir caminhos onde os restaurantes tenham o menor custo possível. Outro cenário seria encontrar um conjunto mínimo de músicas disponível nos dispositivos de usuários num espaço fechado afim de montar uma lista de músicas única para todos. ou qualquer outro objetivo onde se procure um consenso entre dispositivos dispersos. Aqui, consideramos que não existe um acesso a internet ou servidores de apoio a este consenso e sim a comunicação *ad hoc"* entre os elementos.

Esse é um campo de pesquisa bastante explorado recentemente [1]. Também esta em estudo a utilização de algoritmos distribuídos como *Fixed Interaction* e *Random Gossip* para resolução do problema sem a utilização de um ponto central de controle [2] e [3] . Em Li [1], vemos uma estratégia híbrida para minimizar o custo de mensagens, a utilização de clusters de nós que trabalhando como um só, minimizariam o número total de elementos com necessidade de convergência. A análise de Liv [1] alega um ganho de $\Omega(\log n)$ sobre as abodagens tradicionais. Entretanto, para o funcionamento dessa estratégia é necessária a formação dos *clusters* entre os nós. Na presente análise pretendemos verificamos o custo e a viabilidade da formação de tais clusters em um ambiente mais próximo da realidade.

O artigo está organizado da seguinte forma. A SeçãoII apresenta uma breve formalização do problema. Em seguida, na Seção III apresentamos a descrição do método de formação de *clusters* proposto por Li em [1]. Na Seção IV apresentamos a implementação do método de formação de *clusters* e a sua complexidade seguido dos resultados obtidos. Por fim, na Seção VI apresentamos uma análise dos resultados obtidos e proposição de novos trabalhos.

II. Formulação do Problema

Um sistema distribuído como o do cenário estudado é um sistema de memória distribuída, ou seja, um sistema de troca de mensagens. Sistemas de memória distribuída compreendem um conjunto de processadores interconectados de alguma forma [4]. Para representar o algoritmo de consenso utilizamos um grafo $G(V, E)$ onde V são os nós (vértices), representados por cada tarefa e E os meios de comunicação (arestas), representados pelas trocas de mensagens [4]. Para análise do ambiente em rede sem fio, utilizamos o grafo genérico randômico $G(n, r(n))$ para modelar a rede, onde n nos são uniformemente independentes e distribuídos em um espaço e r(n) a distância comum de transmissão de cada um dos nós. É conhecido que sendo $r(n) > \sqrt{\frac{2logn}{n}}$ o grafo tem uma grande probabilidade de ser conectado [1].

A. Consenso Distribuído

O processo de se alcançar um consenso distribuído pode ser conseguido com um *flood* ou com um elemento centralizador, que faz o consenso.

No *flood*, toda mensagem que chega em um nó é enviada para todos os outros nós com que este nó tem um canal de comunicação, menos para o que lhe enviou a mensagem. O consenso será formado, pois em algum momento do tempo, o nó destino vai receber a comunicação, talvez ateé mais de uma vez. Contudo, a quantidade de mensagens geradas na rede afetam a largura de banda e acarretam num aumento exponencial de uso de mensagens. Outra questão é a convergência deste, pois as mensagens podem ficar vagando indefinidamente na rede. A complexidade de mensanges neste caso é quadrática.

O segundo método, atravês de um elemento centralizador, é passível de falhas pois este elemento sair da rede ou mesmo ter um consumo energético alto devido a concentração na recepção de mensanges.

Posto isso, esses dois métodos não são desejados por motivos de consumo excessivo de energia devido a enorme quantidade de mensagens, ou por criar gargalos de comunicação e pontos único de falhas [5].

Assim os algoritmos de consenso distribuído que não definem um único ponto de falha são preferidos por minimizar o custo energético e, devido a sua natureza aleatória de comunicação, evitar pontos únicos de falha.

III. *Cluster* DISTRIBUÍDO

É assumido uma formação de *clusters* seguindo 2 princípios:

- cada nó pertence a a um e somente um cluster;
- em cada cluster existe um nó cabeça que é adjacente a todos os outros nós remanescentes;

Dois *clusters* são ditos vizinhos se existe um canal direto os unindo. Se existir mais de um canal, um deles é ativado. Os nós finais das conexões são chamados nós gateway. É dito que o conjunto de cabeças de *cluster* é assumido como um conjunto dominante, um subconjunto dos nós que estão a 1 hop de qualquer nó, ou seja, o *Minimum Dominant Set*. Sendo o MDS é um problema NP-Hard [1] e a complexidade da aproximação do MDS não se justifica em grandes grafos, é proposto uma versão não interativa de algoritmo de clusterizaçao distribuído.

Assume-se que um nó i tem uma semente inicial i_i, que é única na sua vizinhança. Isto pode ser conseguido através de um número aleatório ou unsando o id do nó. A partir do tempo 0, cada nó inicia um contador com tamanho $t_i = s_i$, que é decrementado por em um a cada instante de tempo até chegar a 0. Se o contador de um nó i expirar, ele se torna um *clusterhead* e envia um pedido de inicialização de *cluster clusterinitialize* para os outros nós. Cada um dos nós com tempo acima de 0, ao receber o pedido de *cluster*, envia sua intenção de se juntar ao cluster com uma mensagem *clusterjoin* e torna seu timer 0. Se um nó receber mais de um *clusterinitizalize* ao mesmo tempo, ele aleatoriamente escolhe um dos *clusters* a se unir. Ao fim, os *clusters* são formados com cada nó pertence a um e somente um *cluster*. Uma vez que há unicidade de semente na vizinhança, é garantido que os *cluster-heads* estão no mínimo a uma distância r de cada um dos outros.

A análise de [1], promete ganhos de $\Omega(\log n)$ sobre os algorítmos tradicionais, entretanto, assume-se que o custo de mensagens e de formação do cluster é desprezível.

IV. METODOLOGIA

Implementamos uma simulação do processo de formação de *clusters* proposto por Li no simulador DAJ *(*Distributet Algorithms for Java). Após a formação do *cluster* um nó pode se do tipo *HEAD* (Cabeça) ou do tipo *CHILD* (Nó a um *hop* de um *cluter*, podendo ao mesmo tempo ser *gateway*) .

O processo de formação pode ser dividido em 3 etapas:

1) formação do *cluster*, onde os nós se identificam como sendo *HEAD* (Cabeça) ou do tipo *CHILD* e quem são seus filhos ou cabeças respectivamente;
2) detecção de vizinhos, onde os nõs cabeça encontram os adjacentes através dos nós gateway;
3) os nós cabeça computam o consenso dos filhos e faz toda atualização;

A etapa de formação de cluster é subdividida em:

1) *CLUSTER_INIT*: onde cada um dos nós decrementa o seu contador interno até o limite 0 para se decidir se será cabeça ou filho.
2) Os nós cabeça enviam uma mensagem *CLUSTER_INIT* para os nós adjacentes e espera pelas respostas;
3) Os nós que recebem o pedido de inicialização respondem com uma intenção de se unir ao cluster com uma mensagem *CLUSTER_JOIN*
4) O nó cabeça devolve uma mensagem *CLUSTER_ACCEPTED* para o nó filho e adiciona-o a sua lista de filhos.

Alguns requisitos são assumidos. Os nós possuem relógio sincronizado pois devem iniciar ao mesmo tempo sua contagem regressiva para atingir o ponto de início de formação, como a unicidade de semente deve ser garantida, inicialmente os nós devem ter alguma informação do número de elementos totais da rede para obter um contador único. Após a formação do *cluster*, o nó cabeça deve esperar a formação dos outros para iniciar o processo de procura de vizinhos num tempo proporcional ao número de elementos na rede.

Na etapa de Detecção de vizinhos, os nós *gateway* devem enviar um *broadcast* pedindo informação de quem são os adjacentes ao seu cluster, todos os que recebem a mensagem devem processá-la, verificar se são provenientes do seu cluster ou de outro e, se forem do mesmo *cluster* descartar. Se a mensagem originar-se de outro *cluster* eles devolvem informação sobre seu cluster. Cada mensagem deve ser analizada pelo nó que recebe, portanto deve ser contabilizada como mensagem do sistema, mesmo que seja enviada como

broadcast. Após finalizar a detecção de vizinhos, o nó cabeça deve novamente esperar a finalização das detecções de vizinhos de todos os outros para iniciar o processo de consenso. Apesar de Li [1] garantir um máximo de 48 *clusters* em qualquer formação, o tempo de formação e o número de mensagens enviadas não é dependente do número de *clusters* e sim do grau dos nós filhos sendo o número de mensagens enviadas $\Theta((n-1)^2)$ se todos os nós forem conectados e somente um *cluster* se formar.

V. Experimentos

Os testes foram realizados considerando o número de nós e o raio de alcance de transmissão. Durante a construção do cenário o algoritmo de geração de nós no simulador verifica a cada nó se algum outro n[o está no seu raio de transmissão e cria uma conexão bidirecional entre eles. O broadcast é tratado como envio de uma mesma mensagem para cada um dos nós adjacentes ao nó originador.

Fizemos testes com cinco raios de transmissão, r: 100, 150, 300, 600 e 850. O valor de r influencia na conectividade do gravo, como descrito na seção II. A quantidade de nós foi variada de 10 a 80 com acréscimo de 10.

Os cenários com menos de 10 nós e raio abaixo de 100 não satisfaziam a premissa de grafo conectado e não foram utilizadas. A convergência e a parada final dos testes acontece no momento em que os nós cabeça possuem conhecimento de seus vizinhos e é calculado o valor de consenso entre os seus elementos pelo método centralizado.

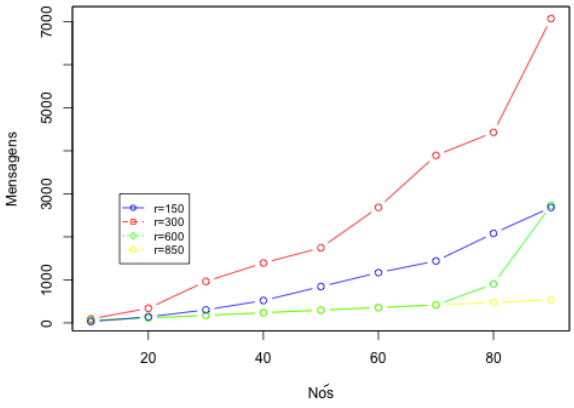

Figura 1. Mensagens versus Nós

A Figura 1 apresenta o resultado da execução da formação dos clusters, contabilizando as mensagens formadas. Como podemos ver no gráfico 1, analizando o número de mensagens geradas no sistema, vemos um crescimento quadrático ligado ao aumento do número de nós e da diminuição do raio que altera a topologia da rede representada pelo número de clusters formados. A curva para o raio 300 termina com uma transmissão de 7072 mensagens e a de raio 850 com 536. Clusters com mais vizinhos geram mais mensagens devido a natureza de busca flood do sistema, entretanto essa relação tem um decaimento quando o raio é 150. Nesse raio de transmissão, o número de gateways cai bastante mas, o número de vizinhos não cresce na mesma proporção. Uma vez que só há devolução de mensagens quando esta vem de outro *cluster* o numero de mensagens cai. O cenário com raio 850 foi utilizado por ser a diagonal do espaço utilizado, 600, o que levou a formação de somente um *cluster*. Esse cenário dminui o número de mensagens porque, sem vizinhos, o número de mensagens de retorno de *cluster* encontrado é 0.

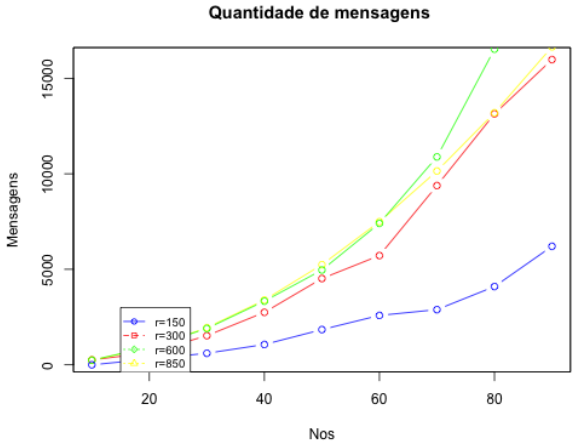

Figura 2. Mensagens recebidas versus Nós

A Figura 2 apresenta uma outra visão do custo de formação, contabilizando as mensagens quando recebidas pelos nós. Essa visão faz sentido uma vez que o custo enérgético não se limita ao envio mas também a recepção de mensages e seu processamento. É possível notar um aumento significativo do número de mensagens em relação ao grafico 1, ficando visível a ordem de complexidade Θn^2 do número de mensagens pelo número de nós.

A Figura 3 correlaciona a quantidade de clusters formados dados os raios propostos e os nós. A alta densidade de nós faz com que a quantidade de *clusters* aumente consideravelmente. Isso implica em mais mensagens de formação de *clusters*.

A Figura 4 mostra como a quantidade de mensagens enviadas foram necessárias para a criação dos *clusters*. Nota-se que o raio 850 e 600 se coincidem até um certo numero de mensagens, isso acontece porque o raio 600 gera somente um cluster em alguns casos.

VI. Conclusão

Neste artigo, vemos que o custo de formação de *clusters* da técnica Li [1] não é desprezível. No caso de uma

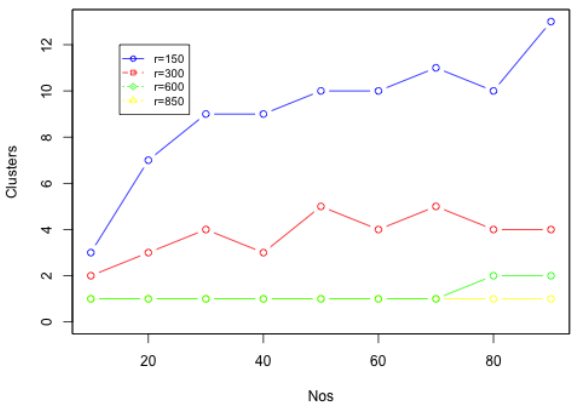

Figura 3. Nós versus Clusters formados

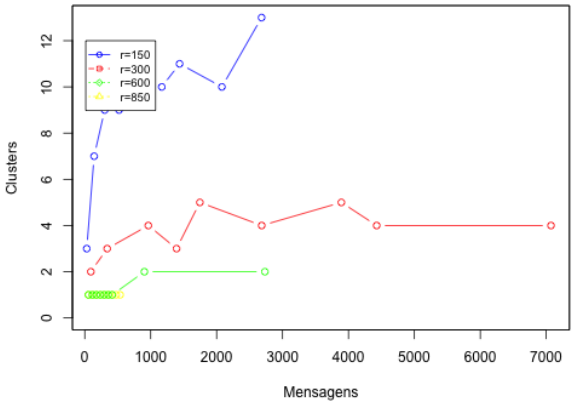

Figura 4. Mensagens versus Clusters formados

topologia totalmente conectada, esta complexidade será da ordem de $\Theta(n^2 * m)$. Essa topologia é o caso mais comum como por exemplo uma sala com todos dentro do raio de alcançe de *wi-fi*.

Em relação a complexidade de tempo sugerido pelos autores, o tempo de formação de cluster é (n) mas com uma constante dependente do tempo de processamento de cada nó. Outro ponto considerado é que, apesar dos autores limitarem sua proposta em uma rede estável, a a utilização de algoritmos distribuídos como o *gossip* faz sentido para configurações de rede não confiáveis [5] que convergem mesmo com perda de nós. O método de utilização de *clusters* cria pontos de falha. A perda dos nós cabeça faz com que o sistema perca $O(n)$ elementos de informação o que exigiria a recriação dos *clusters* e consquentemente, a perda da computação já realizada.

Referências

[1] W. Li and H. Dai, "Cluster-based distributed consensus," *Trans. Wireless. Comm.*, vol. 8, pp. 28–31, January 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1657616.1657622

[2] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems Control Letters*, vol. 53, no. 1, pp. 65 – 78, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167691104000398

[3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2508 – 2530, june 2006.

[4] V. C. Barbosa, *An introduction to distributed algorithms*. Cambridge, MA, USA: MIT Press, 1996.

[5] A. Dimakis, S. Kar, J. Moura, M. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847 –1864, nov. 2010.