

Resolvendo o problema de snapshot em redes DTN utilizando algoritmos distribuídos

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

Resumo—Quando falamos em transmissão de dados utilizando uma rede, esperamos que exista um caminho entre a origem e o destino para que a comunicação seja viável. Esta é uma condição para que seja possível utilizar o protocolo de redes TCP/IP. Mas quando pensamos em alguns cenários no mundo real, como por exemplo uma rede interplanetária, não podemos garantir que todos os dispositivos da rede estarão disponíveis a todo momento. Para resolver este problema surgiu uma nova arquitetura de redes conhecida com DTN (*Delay Tolerant Networks*). Neste tipo de rede geralmente os dispositivos não estão disponíveis durante todo o tempo, sendo necessário a utilização de algumas técnicas diferentes para alcançar a origem. Este artigo faz uma abstração desta rede para um grafo temporal, e propõe uma solução em algoritmos distribuídos para resolver o problema de snapshot em grafos, visando garantir que a mensagem alcance seu destino mesmo em uma rede que mude a todo momento e que o tempo de propagação desta mudança seja lento.

Keywords—grafos, algoritmos, complexidade, snapshot, *Delay Tolerant Network*.

I. INTRODUÇÃO

Sistemas de memória distribuída são sistemas em que múltiplos processadores são interconectados por um canal que permite a comunicação ponto-a-ponto, nos quais os processadores não compartilham fisicamente qualquer tipo de memória. A falta de uma região de memória compartilhada exige que a comunicação entre os processadores seja feita através da troca de mensagens. Como os recursos computacionais são limitados, neste caso, além do limite de memória e de tempo que são exigidos naturalmente por qualquer algoritmo, temos que levar em consideração também o custo que o algoritmo distribuído terá para realizar a troca de mensagens entre os processadores [1].

Em um grafo G com n vértices V_i conectados por m arestas A_j onde os nós mudam de posição a todo momento e o tempo de propagação da mudança seja lento, seria necessário construir a árvore geradora do grafo depois de cada mudança para conhecer sua configuração, possibilitando assim a transmissão de informações de forma direta e com o menor custo possível. Ou seja, as mensagens passariam somente pelos nós que estivessem entre a origem e o destino, evitando que a rede fique sobrecarregada com excesso de mensagens.

Outra possibilidade, um pouco mais simplista, seria enviar as mensagens por inundação, onde cada nó que receber a mensagem, a encaminha para todos os seus vizinhos, caso esta não esteja endereçada a ele.

Naturalmente, para que cada vértice V_i conheça os V_{n-1} vértices do grafo, é necessário que cada vértice V_i receba uma mensagem de cada um dos V_{n-1} vértices do grafo, ou seja, k mensagens são necessárias para se conhecer o estado global, ou snapshot, do grafo. Desta forma, o número de mensagens que trafegarão pelo grafo será sempre linear e sua complexidade é expressa pela fórmula $k = n.m$. Onde, n é o número de vértices e m é o número de arestas.

Aplicando um algoritmo distribuído para resolver o problema do snapshot, conseguimos dividir o processamento para os V_n vértice do grafo possibilitando um processamento mais robusto, e focamos a análise de complexidade [2] somente no algoritmo, levando em consideração não somente a análise de tempo e espaço mas também a análise de troca de mensagens k .

Ao trabalharmos com a possibilidade do grafo G não ser conexo durante todo o tempo, tendo pelo menos um vértice V_i sem nenhuma aresta A_j que ligue V_i a G , uma alternativa de alcançar V_i em um determinado tempo, considerando que qualquer V_i será conexo a G em algum instante de tempo qualquer (grafo temporal) seria a utilização de protocolos sob a arquitetura DTN (*Delay tolerant Network*). Para termos uma visão prática sobre o cenário descrito acima podemos pensar em uma rede interplanetária, que é descrito em maiores detalhes em [3].

Uma técnica de fazer com que uma mensagem k parta de um vértice V_j e chegue até V_i pertencente a G , seria replicar k e enviar para todos os vizinho de V_j e assim sucessivamente até alcançar V_i . Desta forma a mensagem k se propagaria $n.m$ vezes, caracterizando uma transmissão por inundação, onde todas as mensagens passam por todas as arestas do vértice. Apesar da garantia de que a mensagem k sempre alcance o vértice V_i , esta técnica pode provocar grande degradação da rede e grande gasto de energia, sendo inviável para utilização em dispositivos móveis onde o consumo de energia tem que ser controlado.

Para resolver este problema. Em [4] propôs uma técnica de transmissão baseada em poucas cópias de mensagens.

Considerando um cenário estocástico onde a rede se movia com frequência e poderia ser desconectada por algum tempo, um nó envia uma mensagem para o um de seus vizinhos (a escolha deste vizinho era baseada na probabilidade dele conhecer o destino), e atualiza a probabilidade da origem conhecer o destino com um valor alto, desta forma ele evita que a mensagem volte para a origem. Após alguns experimentos o autor chegou a conclusão de que o método não é eficiente, pois a mensagem pode nunca chegar ao seu destino, ou no melhor caso, chegar com um atraso muito grande.

Utilizando o mesmo cenário, e em uma tentativa de resolver o problema da mensagem não atingir o destino, o mesmo autor propôs em [5] uma técnica que nomeou de “Pulverizar e esperar” do inglês “*Spray and wait*”. Esta técnica consiste em pulverizar a mensagem entre os vizinhos do nó até que se tenha um número de mensagens suficientes para garantir que ela seja entregue ao seu destino, e menor do que o número de mensagens enviadas pelo método de inundação. Conforme apresentado em seus experimentos, o método se mostrou mais eficiente do que o método proposto anteriormente e o método de inundação, pois como resultado ele obteve um bom desempenho na transmissão e uma pequena sobrecarga de mensagens.

As redes de computadores utilizam como padrão a pilha de protocolos TCP/IP, que funciona perfeitamente bem para o ambiente pelo qual foi projetada. Por outro lado, quando saímos de suas condições ideais de uso, como por exemplo, redes sem fio entre dispositivos móveis onde se tem restrição de consumo de energia e falta de garantia de conectividade entre o destino e a origem. É necessário utilizarmos outras técnicas de transmissão de dados, o que deu origem a uma nova arquitetura de transmissão, conhecida como DTN.

Apesar de redes DTN terem sido projetadas com foco em redes interplanetárias, eles têm uma boa aplicação em redes para dispositivos móveis onde as propriedades do TCP/IP geralmente não são satisfeitas, pois os limites de armazenamento são muito restritos e os dispositivos não estão conectados durante todo o tempo. Em [6] é mostrado que é viável a utilização de redes DTN neste ambiente, utilizando as tecnologias de comunicação que são bluetooth, Wi-Fi, 3G e interfaces de rádio.

Esta nova arquitetura de redes prevê a utilização de uma técnica de comutação e de armazenamento persistente de dados em uma nova camada que fica logo abaixo da camada de aplicação, que é denominada camada de agregação ou em inglês *Bundle Layer*. Esta camada é responsável de certa forma por isolar as camadas inferiores da camada de aplicação garantindo interoperabilidade de dispositivos de rede. Redes que utilizam arquitetura DTN tem as seguintes características:

- Atrasos longos: Uma DTN não possui tempo estimado para o pacote chegar ao seu destino.
- Frequentes desconexões: As desconexões podem ocorrer

pela mobilidade, que provoca constantes mudanças na topologia da rede.

- Capacidade de Armazenamento: A principal característica das DTNs é a propriedade de “armazenar e enviar” do inglês “*store-and-forward*”. Este processo de transmissão é exemplificado na Figura 1

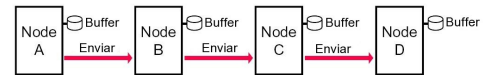


Figura 1. Exemplo de *Storage-and-forward* de uma DTN

As aplicações DTNs enviam mensagens de tamanhos variáveis chamadas de unidades de dados da aplicação (Application Data Units - ADUs). As mensagens são transformadas pela camada de agregação em uma ou mais unidades de dados de protocolo (Protocol Data Units - PDUs) denominadas agregados (bundles), que são armazenados e encaminhados pelos nós DTN. Esta camada de agregação suporta a retransmissão nó a nó através do mecanismo denominado *Transferência de Custódia*, que tem como objetivo passar a responsabilidade da entrega de uma mensagem de um nó para outro nó, iniciando na origem e sendo completada no destino.

Estas características exigem que sejam estabelecidas alguns critérios de transmissão entre os nós DTN, e uma boa política de roteamento pode diminuir o número de mensagens na rede ou até mesmo o tempo que as mensagens gastam para atingir seus destinos. As principais técnicas de roteamento são:

Contato Direto: O nó de origem só transmite uma mensagem para uma mula, (nome dado aos nós pertencentes a uma rede DTN), se o próximo contato da mula for diretamente com o destino.

Primeiro Contato: O Nó de origem envia a mensagem para a primeira mula com a qual vier a estabelecer contato. A mula, por sua vez, envia a mensagem para a primeira região que estabelecer contato e assim conseqüentemente.

Epidêmico: É a principal proposta para cenário estocástico, pois suporta a entrega eventual de mensagens a destinos arbitrários com suposições mínimas relativas ao conhecimento da topologia de rede.

Probabilístico: Nos contatos previsíveis, apesar do momento exato do estabelecimento de cada contato entre dois nós da rede ser desconhecido, existe uma previsão do intervalo dentro do qual cada contato irá acontecer. É estabelecida uma probabilidade de ocorrência de cada contato.

Levando em consideração que o tempo para a rede se estabilizar é lento e a rota se modifica com muita frequência (características presentes em redes DTN), temos o cenário ideal para aplicarmos um algoritmo distribuído com a finalidade de conhecer o estado global do grafo. Pois, para

que se chegue ao estado global do grafo é necessário que haja troca de mensagens entre todos os nós da rede. Assim podemos analisar o impacto causado pela ausência de um nó em uma rede DTN, comparando seus resultados com a troca de mensagens para se chegar ao estado global de um grafo onde todos os nós são conexos durante todo o tempo.

As redes DNT tem ampla aplicabilidade em cenários reais, isto se dá pelo fato de que tem se tornado necessário estarmos conectados a uma rede em grande parte do tempo. Como exemplo, temos as redes de dispositivos móveis (celulares, *smartphones*, *ipads* e diversos outros dispositivos que têm sistema embarcado), além de redes militares, redes interplanetárias e sistemas de rastreo de animais selvagens.

II. MÉTODOS

Imaginando um cenário onde o algoritmo distribuído faria a troca de mensagens para determinar um estado global do grafo. Estas mensagens seriam enviadas de forma assíncrona, pois não se pode estabelecer limites para a troca destas mensagens. Descrevendo o cenário a ser analisado, temos um algoritmo distribuído e assíncrono que realiza a troca de mensagens entre os nós gerando eventos, até atingir o estado final e chegar ao estado global do grafo.

Cada evento pode ser representado por uma tupla de 6 elementos:

$$\xi = \{n_i, t, \varphi, \sigma, \sigma', \Phi\}$$

Onde:

- n_i é o nó no qual o evento ocorre.
- t é o instante do tempo, de acordo com o relógio local de n_i , em que o evento ocorreu.
- φ é a mensagem, se existir, que disparou o evento, quando de sua recepção, em n_i .
- σ é o estado de n_i antes da ocorrência do evento.
- σ' é o estado de n_i depois da ocorrência do evento.
- Φ é o conjunto de mensagens, se existir, enviado por n_i como consequência da ocorrência do evento.

Um conjunto de eventos (Ξ) pode ser representado por:

$$\Xi = \sum \sigma_n$$

Para determinarmos o estado global, o algoritmo é executado de forma assíncrona em cada nó, com isto, a execução do algoritmo se torna independente e as informações registradas ficam esparramadas na rede. Supondo que as mensagens enviadas no pulso $s - 1$ cheguem ao seu destino no pulso s , desta forma desconsideramos a possibilidade de perda de mensagem durante o envio. Os nós armazenam seu próprio estado e o estado de suas arestas e enviam uma mensagem especial para seus vizinhos, esta mensagem é chamada *Marker*.

Um nó N_0 presente em G , registra seu estado e envia a mensagem de *Marker* para seus vizinhos, quando qualquer N_i , vizinho de N_0 , recebe a mensagem de *Marker* enviada

por N_0 , ele registra seu estado global e envia mensagens de *Marker* para seus vizinhos, este processo continua até que cada N_i pertencente a G receba em todas suas arestas as mensagens dos seus N_{i-1} vertices do grafo G . Este procedimento é melhor ilustrado na Figura II.

Algoritmo A Record Global State

```

▷ Variables:
  node.statei = nil;                                {Estado local do subtrato}
  edge.statei =  $\forall n_j \in Neig_i$ ;                    {Estado da aresta ( $n_j \rightarrow n_i$ )}
  recordedi = false;                                {Indica se o estado local já foi registrado}
  receivedi = false  $\forall n_j \in I.Neig_i$ .              {Indica se marker já foi recebido de  $n_j$ }

▷ Input:
  msgi = nil.
Action if  $n_i \in N_0$ :
  {Estado do subtrato é registrado}
  node.statei  $\leftarrow \sigma_i$ 
  {Marca que registrou o estado local}
  recordedi  $\leftarrow$  true;
  {Envia marker, com o estado local, em todas as arestas de saída}
  Send marker to all  $n_j \in O.Neig_i$ .

```

Algoritmo A Record Global State

```

▷ Input:
  msgi = marker | origin(msgi) = ( $n_j \rightarrow n_i$ ).
Action:
  {Indica que recebeu marker de  $n_j$ }
  receivedi  $\rightarrow$  true;
  {Faz o registro do estado local como descrito na ação (1), se ainda não fez}
  if not recordedi
  then begin
    node.statei  $\leftarrow \sigma_i$ 
    recordedi  $\leftarrow$  true;
    Send marker to all  $n_k \in O.Neig_i$ .
  end

```

Figura 2. Pseudo-Código do algoritmo de Cland-Lamport

III. ANÁLISE DE COMPLEXIDADE

Considerando que para o algoritmo chegar ao seu estado final de execução ele depende diretamente da trocar de mensagens. A análise de complexidade [7] do Programa passa a ter mais sentido se consideramos o número de mensagens que transita na rede. Fazendo a análise da complexidade:

$$O = n.m$$

Onde:

- n é a quantidade de nós presentes no grafo.
- m é o número de arestas do grafo.

Tendo como exemplo um grafo conexo G com 19 vértices e 84 arestas, ilustrado pela Figura III. Para conhecer o estado global do grafo são necessárias:

$$\begin{aligned}
 O &= n.m \\
 O &= 19 \times 84 \\
 O &= 1596 \quad \text{mensagens}
 \end{aligned}$$

Esta análise de complexidade pode ser comprovada consultando na Tabela I o resultado dos experimentos.

A análise de complexidade deixa de ser simples quando feita sobre redes DTN. O fato de não sabermos quantos nós

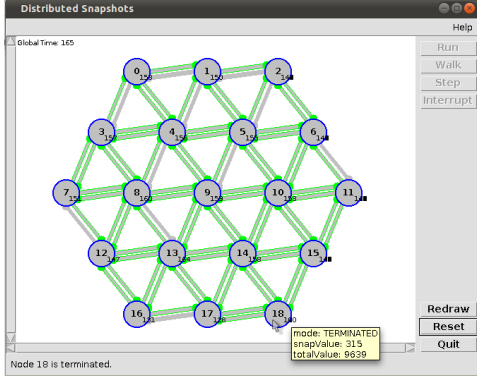


Figura 3. Grafo conexo com estado global definido

N_i estarão desconexos a G em um instante de tempo qualquer e nem por quanto tempo permanecerão, torna impossível fazermos uma análise de complexidade de tempo. E o fato de um nó N_i qualquer não estar conectado a G em um instante de tempo, provoca várias retransmissões de mensagens. Se consideramos o pior caso, quando nenhum N_i é conexo em nenhum instante de tempo, teríamos infinitas mensagens sendo enviadas para a rede.

Como não podemos prever o tempo que N_i será desconexo a G , a complexidade da troca de mensagens seria uma análise estatística da Esperança (E) da probabilidade de N_i permanecer conexo a G por um instante de tempo qualquer. Para determinarmos a E precisamos de uma população (termo estatístico para nomear um conjunto de dados que será analisado), que poderia ser construída a partir de dois cenários. O cenário determinístico, onde a população é feita sobre um histórico de ocorrências e o cenário estocástico, onde a população é construída através de milhares de testes. A construção de ambos os cenários não está no escopo deste *Paper*. Por este motivo, na Seção IV apresentamos os resultados dos experimentos realizados.

Para a realização das simulações foi utilizado o simulador DAJ [8], que é uma ferramenta visual e iterativa para o estudo de algoritmos distribuídos, pois deve-se apresentar explicitamente cada passo da execução do algoritmo e o DAJ se encarrega de exibir o estado de sua execução. Como algoritmos distribuídos geralmente são de difícil compreensão, o DAJ ajuda no estudo deste algoritmos permitindo a criação de logs e a reprodução do cenário até que os conceitos sejam compreendidos.

IV. EXPERIMENTOS

Os experimentos foram realizados sob o simulador DAJ utilizando a implementação do algoritmo de Cland-Lamport adaptado para resolver o problema de snapshot de grafos em redes DTN.

Todos os testes foram realizados em um grafo G com as seguintes propriedades:

- G não é um grafo dirigido;

- Todos os v_n vértices de G têm que ser conexo por um instante de tempo qualquer;
- Não acontece perda de mensagens no canal de transmissão;
- A capacidade de armazenamento de mensagens no vértices é ilimitada;

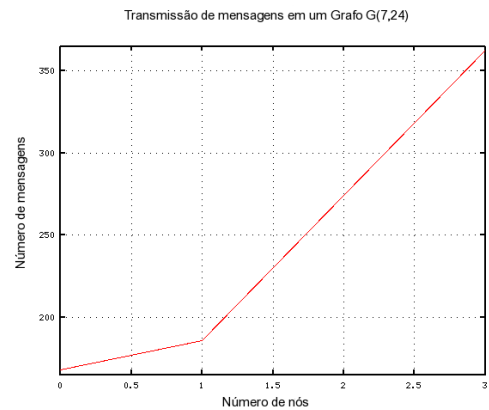
Seguindo estas propriedades os experimentos foram realizados em 2 grafos de tamanhos diferentes, sendo o primeiro um grafo com 7 vértice e 24 arestas $G(7,24)$, e o segundo um grafo com 19 vértices e 84 arestas $G(19,84)$, e seus resultados são exibidos na Tabela I.

Tabela I
TABELA COM OS RESULTADOS OS EXPERIMENTOS

Tamanho do grafo	Número de mensagens na rede			
	Número de nós inativos			
	0	1	3	10
$G(7,24)$	168	186	363	7
$G(19,84)$	1596	6006	2718	12146

Para o primeiro grafo $G = (7,24)$ o algoritmo foi executado para resolver 3 instâncias diferentes. Na primeira instância foi considerado que o grafo é conexo durante todo o tempo. Apesar desta instância não ilustrar um exemplo de DTN, ela se torna um importante critério de comparação com as demais. Já na segunda instância levamos em consideração que somente um vértice será instável, isso que dizer que ele será conectado e desconectado do grafo G em instantes de tempo que são determinados por uma variável randômica. Finalmente, na ultima instância teremos 3 vértices instáveis.

A troca de mensagens necessárias para resolver este problema pode ser visto com mais detalhes no Gráfico IV



No segundo grafo $G = (19,84)$, o problema foi resolvido para 4 instância distintas. Sendo a primeira, com o grafo conexo durante todo o tempo. Como no caso anterior, a solução desta instância serve como critério de compração entre as demais, e sua análise de complexidade, foi detalhada na Seção III.

Ao observarmos o Gráfico IV, conseguimos identificar um desvio de padrão para a solução da segunda instância, que considera que no grafo tem somente um vértice instável. Isto acontece pelo fato do vértice instável ter ficado muito tempo inativo, causando uma grande replicação de mensagens na rede. Esta solução representa de forma clara as dificuldades citadas na Seção III para calcularmos a análise de complexidade deste algoritmo.

A terceira e quarta instância do problema, consideram que no grafo existem 3 e 10 vértices instáveis respectivamente, e os resultados obtidos a partir da solução destas instâncias mostram uma aumento no número de mensagens necessárias para resolver o problema semelhante a curva de um problema exponencial.



Os resultados dos experimentos e o código fonte utilizado para resolver os problema proposto por este *Paper* estão disponíveis para download no link <http://dl.dropbox.com/u/4297019/DTN-Snapshot.zip>

REFERÊNCIAS

- [1] C. V. Barbosa, *An Introduction to Distributed Algorithms*, 1st ed. The MIT Press, 1996, iSBN-10: 0262024128.
- [2] N. Ziviani, *Projeto de Algoritmos com Implementaes em Java e C++*. Cengage Learning, 2006, iSBN-10: 8522105251.
- [3] S. Burleigh, "Delay-tolerant networking: An approach to interplanetary internet," *IEEE Communications Magazine*, 2003.
- [4] T. Spyropoulos and C. S. Raghavendra, "Efficient routing in intermittently connected mobile networks: The single-copy case," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 63–76, 2008.
- [5] —, "Efficient routing in intermittently connected mobile networks: The multiple-copy case," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 77–90, 2008.
- [6] C. Caini, P. Cornice, R. Firrincieli, and M. Livini, "Dtn meets smartphones: future prospects and tests," *International Symposium on Wireless Pervasive Computing (ISWPC)*, 2010.
- [7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. The MIT Press, 2009, iSBN-13: 978-0-262-53305-8.

- [8] W. Schreiner, *DAJ - A Toolkit for the Simulation of Distributed Algorithms in Java*, Research Institute for Symbolic Computation (RISC-Lins), <ftp://ftp.risc.unilinz.ac.at/pub/parlab/daj/report/report-main.ps.gz>, October 1998.