Análise e Modelagem de Algoritmos para Eleição de Líder em Sistemas Distribuídos

Émerson R. Silva[1], **Eduardo C. Julião**[1], **Patricia T. Endo**[1]

[1]GRupo de Estudos Avançados em Tecnologia da Informação e Comunicação (GREAT)
Universidade de Pernambuco (UPE) – Campus Caruaru – Caixa Postal 55014-908 – Caruaru – PE – Brazil

{emersonremigio,eduardocesarbj,patriciaendo}@gmail.com

Abstract. *Currently, the distributed systems area is expanding and concepts arise according to the need. In some distributed systems, a component must behave differently and this component is selected by an election process. This work aims to compare existing distributed solutions to elect leaders by modeling algorithms in different scenarios and making a comparison between the features and behaviors of them, showing their limitations and resources used in an election.*

Resumo. *Atualmente a área de sistemas distribuídos está em expansão e novos conceitos vão surgindo de acordo com a necessidade. Em alguns sistemas distribuídos, um componente precisa se comportar de maneira diferenciada e única. Este componente é escolhido através de um processo de eleição. Este trabalho tem como objetivo principal comparar soluções distribuídas existentes para escolha de líderes através da modelagem dos algoritmos em diferentes cenários e realizando uma comparação entre as funcionalidades e comportamentos deles, mostrando como resultado final as suas limitações e recursos utilizados na execução de uma eleição.*

1. Introdução

A evolução computacional que vem acontecendo nas últimas décadas, permitiu que vários computadores pudessem comunicar uns com os outros em alta velocidade e trocar grande quantidade de dados através de redes. Existiam sistemas centralizados que para ter uma grande capacidade de processamento eram necessários *mainframes* de alto custo, mas mesmo estes ficavam sujeitos a falhas, por estarem implementados em um único local, qualquer problema neste servidor central faria com que o serviço ficasse indisponível ao usuário. Isto viabilizou o surgimento de sistemas distribuídos, que são componentes com um comportamento autônomo que se comunicam através de mensagens. Colouris, Dollimore e Kindberg (2007, p. 1) definem que: "*um sistema distribuído é aquele no qual componentes localizados em computadores interligados em rede se comunicam e coordenam suas ações apenas passando mensagens*".

Uma questão muito importante a ser tratada em sistemas distribuídos é a concorrência dos recursos utilizados pelo sistema para que todos os computadores possam utilizar determinado recurso sem que exista um conflito ou inconsistência. Segundo Tanenbaum e Steen (2007), isso é realizado pelos algoritmos distribuídos de exclusão mútua, que organizam para que os processos acessem o recurso de acordo com

a requisição e não todos ao mesmo tempo. Em sistemas distribuídos existe um único algoritmo implementado em todos os nós, porém em determinados momentos existe a necessidade de que um nó se comporte de maneira diferente (ou seja, passe a atuar como um **líder**), e para escolher o líder é necessário realizar um **processo de eleição**.

Este artigo tem como principal objetivo comparar algoritmos de eleição de líder em sistemas distribuídos realizando uma análise das variáveis com relação ao tempo e aos recursos (quantidade de mensagens) utilizados em cada algoritmo. Após análise do funcionamento de cada algoritmo nas modelagens, espera-se identificar qual tem o melhor desempenho para determinado cenário ou topologia, visando também identificar as limitações, sejam elas de implementação ou de topologia, que cada um possui.

Para tanto, este artigo está organizado da seguinte forma: a Seção 2 descreve as especificações de cada algoritmo que será analisado e mostra o funcionamento; na Seção 3 são apresentados os resultados das análises comparativas entre os algoritmos, e a Seção 4 conclui o trabalho e apresenta modificações e trabalhos futuros que podem ser implementados neste artigo.

2. Algoritmos de Eleição para Sistemas Distribuídos

Os algoritmos distribuídos muitas vezes precisam que um nó se comporte de maneira diferente, desempenhando assim uma função especifica e única para todo sistema (neste artigo, este nó será chamado de **líder**). Para escolher o nó existe um processo chamado de eleição que é executado por um **algoritmo de eleição** por todos os nós do sistema, dentre todos apenas um torna-se o líder. Alencar (1998 p. 4) explica que: "*O principal requisito para algoritmos de eleição de líder é de que a escolha do elemento eleito seja única, ainda que vários nós estejam tentando eleger-se.*".

O líder pode ser escolhido de acordo com vários fatores, dentre eles pode-se citar como exemplo: um endereço IP, endereço físico do nó, quantidade de processamento ou qualquer identificação única, para que cada nó seja distinto. Ao abordar o assunto de algoritmos de eleição são tomados como base de estudos dois algoritmos tradicionais, pela simplicidade de processos para eleição: o algoritmo do valentão (Garcia-Molina, 1982) e o algoritmo em anel (Tanenbaum e Steen, 2007); além de um algoritmo mais recente, proposto por (Santoro, 2007), denominado Yo-Yo.

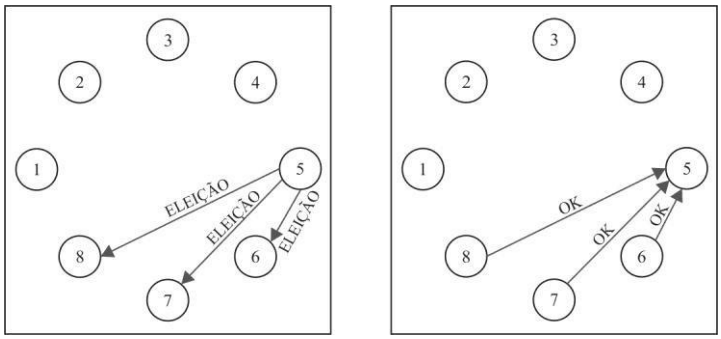
2.1. Algoritmo do valentão

O algoritmo do valentão recebe esse nome porque o nó mais robusto vence a eleição e torna-se líder. Antes que a eleição tenha inicio há duas premissas: cada nó tem uma identificação única e conhece os vizinhos maiores que ele próprio. Então, um nó ativo qualquer vai perceber se não há líder ativo para poder iniciar uma eleição.

Segundo Garcia-Molina (1982) o nó que inicia a eleição envia uma mensagem "ELEIÇÃO" para todos os nós maiores que ele, ativos ou não (Figura 1.a), e aguarda uma resposta positiva (Figura 1.b). Caso ele receba ao menos uma resposta, seu processo de eleição é finalizado e um outro nó vai continuar com a eleição. Porém, caso nenhum nó responda, ele torna-se o líder.

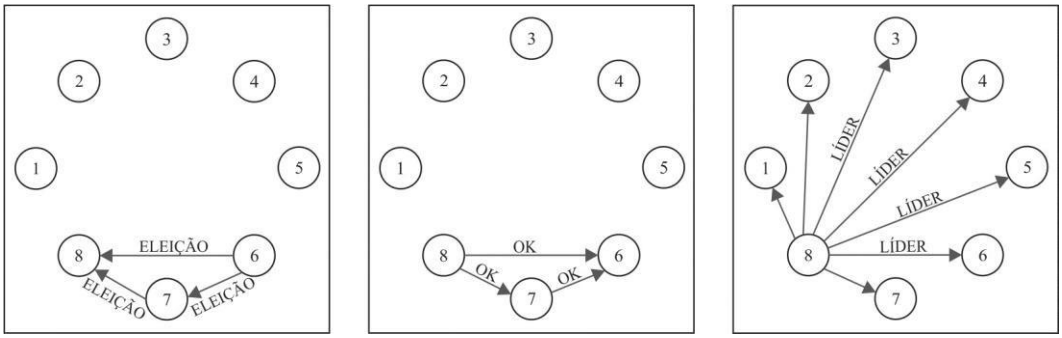
Um nó pode receber a mensagem de eleição a qualquer momento, quando recebe ele envia um "OK" de volta como resposta e toma o poder de fazer a eleição novamente.

Assim em algum momento todos os nós maiores do que o nó que iniciou a eleição vão ter tomado o poder e iniciado também a eleição (Figura 2.a), após receberem as respostas "OK" (Figura 2.b) o nó maior uma hora vai realizar o mesmo processo e então vencer a eleição e mandar a mensagem "LÍDER" a todos os nós do sistema (Figura 2.c).



(a) (b)

Figura 1. Iniciando a eleição com o Algoritmo do Valentão



(a) (b) (c)

Figura 2. Nó sendo eleito líder do sistema

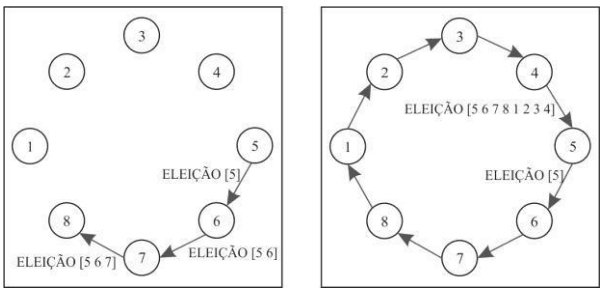
2.2. Algoritmo em anel

O algoritmo em anel, segundo Tanenbaum e Steen (2007), consiste na utilização de um anel em que os nós estão ordenados por ordem física ou lógica, e ao terminar a eleição elege o nó mais robusto como líder.

A eleição é iniciada quando qualquer nó percebe que não há líder ativo, porém mais de um nó pode perceber isso simultaneamente e iniciar eleições em paralelo; de toda forma, o resultado é o mesmo. O nó que percebe a ausência do líder cria uma mensagem "ELEIÇÃO" para ser repassada no anel (como mostra a Figura 3.a), que contém inicialmente a identificação do próprio nó, e a envia para o próximo nó do anel. Este acrescenta na mensagem a sua identificação e a envia novamente. Este processo ocorre até que todos os nós do anel tenham realizado esta tarefa. A mensagem de eleição para de circular quando o nó inicial recebê-la novamente (Figura 3.b) e, comparando o primeiro item da lista, perceber que a primeira identificação é a dele.

Ao fim da volta no anel, o nó que iniciou a eleição tem uma lista com a identificação de todos os nós ativos, então ele escolhe o nó com a maior identificação e realiza o processo de passar a mensagem pelo anel novamente, mas agora é uma mensagem "LÍDER" com a identificação do nó que se tornou o líder. Esta mensagem

para de circular no anel quando chegar ao nó que a criou, e então o mesmo finaliza o processo de eleição.



(a) **(b)**

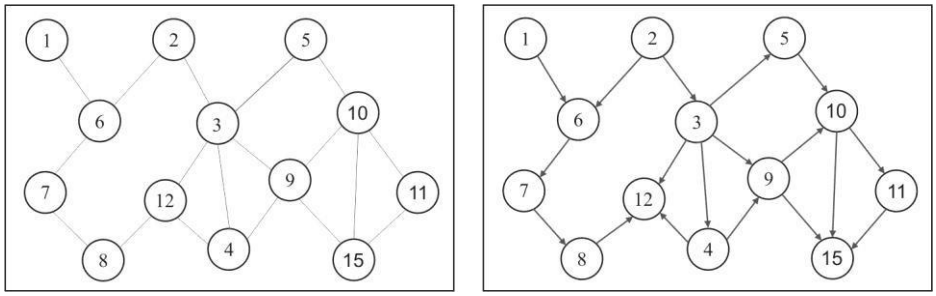
Figura 3. Iniciando eleição em anel e repassando a mensagem de eleição

2.3. Algoritmo Yo-Yo

O Yo-Yo (Santoro, 2007) é um algoritmo de busca mínima, onde o nó com menor identificação (ID) é escolhido como líder. Ele consiste em duas partes: préprocessamento e sequência de iterações.

2.3.1. Setup

A fase de pré-processamento é chamada de *setup*, onde todos os componentes tem um ID único e conhecem os de seus vizinhos, onde são criados *links* diretos apontando do menor ID para o maior, construindo assim um grafo direcionado.



(a) **(b)**

Figura 4. Organização do cenário no *setup*

Na Figura 4.a é mostrado o sistema inicial e na Figura 4.b o *setup* onde, após criado o grafo, cada nó já sabe o papel que irá desempenhar na eleição. Neste cenário os nós 1 e 2 serão *sources*, os nós 15 e 12 serão *sinks* e os demais nós serão *internals*. Segundo Santoro (2007) o grafo resultante é acíclico e neste grafo existem três tipos de nós: *source*, *sink* e *internal*. O *source* é um nó que é possui ID menor que todos os seus vizinhos e onde os *links* só saem dele (ele é um mínimo local); *sink* é um nó que é o maior que todos os vizinhos, onde os links só chegam nele e nenhum sai dele (ele é um máximo local); e *internal* é um nó que não é um *source* e nem um *sink*.

2.3.2. Iteração

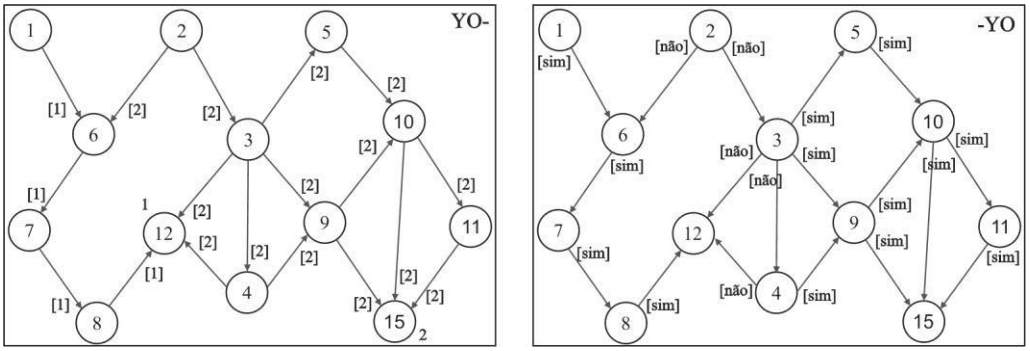
A essência do algoritmo é uma sequência de iterações onde cada iteração consiste em um estágio da eleição onde os aptos a líder são os *sources*.

2.3.2.1 YO-

Segundo Santoro (2007), esta fase é iniciada pelos *sources* para que através dos vizinhos ele possa mandar a sua mensagem (seu próprio ID) até os *sinks*. Inicialmente, um *source* manda o valor para os *links* de saída, os nós *internals* aguardam até receber o valor de todos os *links* de entrada e em seguida pega o valor mínimo e passa adiante para os *links* de saída (Figura 5.a). Os nós *sinks*, por sua vez, aguardam receber o valor de todos os vizinhos com *links* de saída, escolhe o menor ID e inicia a segunda fase, -YO.

2.3.2.2. -YO

Segundo Santoro (2007), esta fase é iniciada pelos *sinks* para eliminar alguns candidatos a líder e transformar alguns sources em *sinks* ou *internals*. Um *sink* vai mandar para todos os vizinhos com *links* de entrada pra ele um "sim", cuja mensagem possui um ID menor, e um "não" para aqueles que tiverem mensagem com ID maior. Um nó interno espera receber a mensagem de todos os *links* de saída e se receber um "sim" de todos, ele manda "sim" para os vizinhos que tem *links* de saída para ele com menor ID e um "não" para os demais. Por fim, os *sources* esperam até receber mensagens de todos os seus *links* de saída. Se todos os votos forem "sim", ele continua ativo para a próxima iteração, se pelo menos um voto for "não", ele não mais é um candidato (Figura 5.b).



(a) **(b)**

Figura 5. Sequência de iterações YO- e -YO

A Figura 5 mostra como vão ocorrer as fases YO- e -YO. Na fase YO- a mensagem com a identificação dos *sources*, nós 1 e 2, é passada vizinho a vizinho e só pode ser repassada após todos vizinhos que direcionam um *link* para o nó mandarem suas respectivas mensagens, então é feita uma comparação e segue a menor mensagem. As mensagens vão seguir por todos os nós até os *sinks* e lá é iniciada a fase -YO que é responsável por reduzir o número de *sources*. É feita uma comparação entre a mensagem que o *sink* possui e a mensagem dos vizinhos, para os que tiverem a mesma mensagem recebem um "sim" e os que não tiverem recebem um "não", como o nó 2 recebeu "não" como resposta então ele não está mais apto, acaba a eleição com o nó 1 sendo eleito líder.

3. Resultados

Utilizou-se a ferramenta NetLogo para a modelagem e simulação dos três algoritmos de eleição apresentados anteriormente: valentão, anel e Yo-Yo. O objetivo principal das

simulações é analisar como os mesmos se comportam em tempo de execução, com relação à quantidade de tempo para finalizar uma eleição e quantidade de mensagens trocadas entre os nós. Para tanto, foram utilizadas topologias de rede com 15, 30 e 60 nós com 30, 60 e 12 repetições, respectivamente.

3.1 Quantidade de tempo para execução de uma eleição

Os gráficos das Figuras 6.a, 6.b e 6.c, mostram a média da quantidade de tempo (em ticks, que representam a unidade de tempo da ferramenta NetLogo) de cada algoritmo para finalizar seu processo de eleição de líder, nos cenários com 15, 30 e 60 nós, respectivamente.

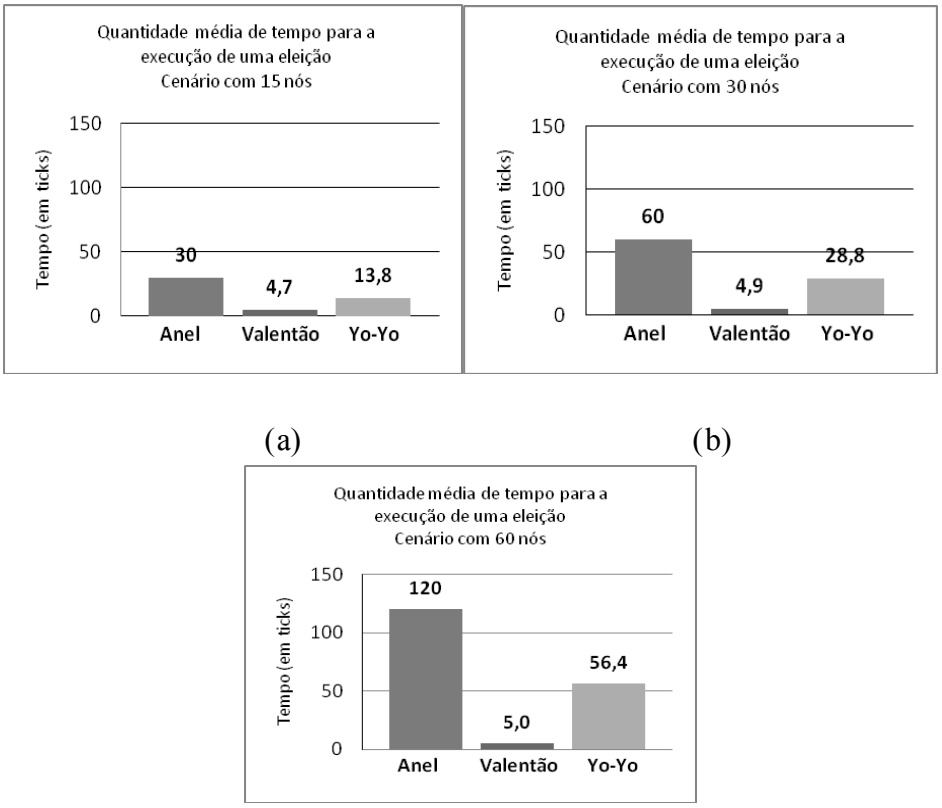


Figura 6. Gráficos de quantidade média de tempo para execução de uma eleição

Para concluir a eleição no cenário com 15 nós, em média, o algoritmo em anel demorou 30 *ticks*, o valentão demorou 4,7 *ticks* e o yo-yo demorou 13,8 *ticks*. Já no cenário com 30 nós, o algoritmo em anel demorou 60 *ticks*, o valentão demorou 4,9 *ticks* e o yo-yo demorou 28,8 *ticks*. E no cenário com 60 nós, o algoritmo em anel demorou 120 *ticks*, o valentão demorou 5 *ticks* e o yo-yo demorou 56,4 *ticks*.

3.2 Quantidade de mensagens trocadas

Já os gráficos das Figuras 7.a, 7.b e 7.c mostram a média da quantidade de mensagens trocadas entre os nós de cada algoritmo para finalizar seu processo de eleição de líder, nos cenários com 15, 30 e 60 nós, respectivamente.

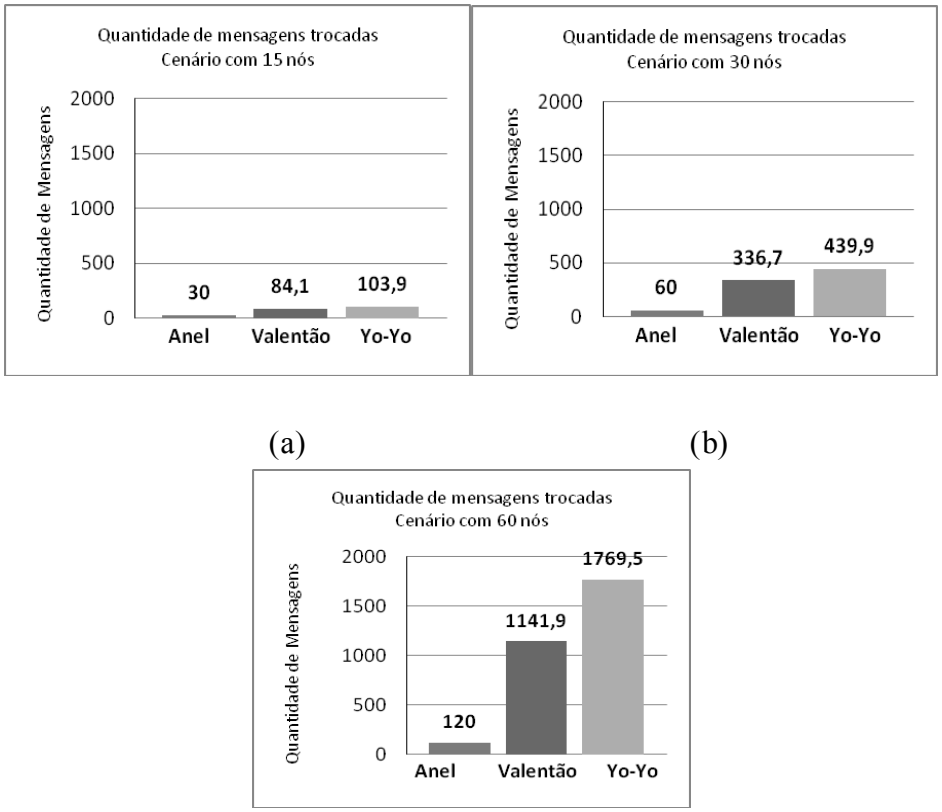


Figura 7. Gráficos de quantidade média de mensagens trocadas para eleição

Para concluir a eleição no cenário com 15 nós, em média, o algoritmo em anel trocou 30 mensagens, o valentão trocou 84,1 mensagens e o yo-yo trocou 103,9 mensagens. Já no cenário com 30 nós, o algoritmo em anel trocou 60 mensagens, o valentão trocou 336,7 mensagens e o yo-yo trocou 439,9 mensagens. E, por fim, no cenário com 15 nós, o algoritmo em anel trocou 120 mensagens, o valentão trocou 1141,9 mensagens e o yo-yo trocou 1769,5 mensagens.

3.3 Análise e Discussão

A partir dos resultados das simulações, observou-se que o algoritmo em anel é o que mais consome tempo para finalizar uma eleição, porém a quantidade de mensagens utilizada pelo mesmo é a menor. Percebeu-se também que, ao aumentar a quantidade de nós na topologia, o tempo de uma eleição do algoritmo em anel aumenta consideravelmente em relação aos outros dois algoritmos.

O algoritmo do valentão demora menos para executar uma eleição e o tempo utilizado se mantém constante independente da quantidade de nós, mas o número de mensagens enviadas para o pouco tempo que ele passa sendo executado mostra que ele

consome muito do sistema deixando a rede sobrecarregada durante a eleição. Com o tempo constante, quanto maior a quantidade de nós, mais a rede fica sobrecarregada.

O algoritmo Yo-Yo, em todas simulações, apresentou o maior número de mensagens de acordo com a quantidade de nós, de maneira que, ao dobrar a quantidade de nós, as mensagens aproximadamente triplicam. Ao analisar o funcionamento é visto que ele é um algoritmo muito síncrono, e sincronia é o principal problema para ser resolvido em sistemas distribuídos.

A Tabela 1 mostra uma comparação entre os três algoritmos com relação a tempo de execução e quantidade de mensagens.

Tabela 1. Comparação entre os algoritmos

Algoritmo	**Topologia**	**Tempo de execução** (em ticks)	**Quantidade de mensagens**
Valentão	Totalmente conectada	Menor dentre os três	Possui quantidade média dentre todos
Anel	Em anel	Maior dentre os três	Menor dentre os três
Yo-Yo	Em árvore	Possui tempo médio	Maior dentre os três

O algoritmo do valentão é restrito a uma topologia na qual todos os nós possuem conexões com todos; o algoritmo em anel utiliza a topologia em anel e limita-se a esta, pois para que funcione necessita de que os nós estejam ordenados em anéis físicos ou lógicos; já o algoritmo Yo-Yo utiliza topologia em árvore e limita-se a topologias acíclicas, porque ele precisa de um grafo para a execução do algoritmo de eleição. Assim, o algoritmo do valentão apresenta o melhor tempo de execução e o algoritmo em anel apresenta o melhor resultado com relação a quantidade de mensagens enviadas.

4. Conclusão e Trabalhos Futuros

Este artigo descreveu o funcionamento de três algoritmos para eleição de líder em sistemas distribuídos e mostrou as especificações e limitações de cada um. O objetivo principal deste artigo foi servir como fonte de estudo inicial de algoritmos de eleição de líder em sistemas distribuídos, mostrando que pode ser escolhido de diversas formas.

Os resultados apresentados mostraram que os algoritmos foram desenvolvidos para cada tipo de topologia e que dificilmente funcionariam em outra, então isso representa uma grande limitação dos algoritmos.

Como trabalhos futuros, pretende-se aprimorar o estudo e as modelagens dos algoritmos, comparando-os com outros algoritmos distribuídos. Também pretende-se implementar e estudar a questão de tolerância a falhas dos algoritmos e avaliar o desempenho dos mesmos.

Referências

Alencar, J. F. “Algoritmos para eleição de líder em sistemas distribuídos”, Universidade Estadual de Campinas - Instituto de Computação. Campinas, 1998.

Colouris, G.; Dollimore, J.; Kindberg, T. “Sistemas Distribuídos Conceitos e Projeto”, 4. ed. Porto Alegre: Bookman Editora, 2007.

Garcia-Molina, H. “Elections in a Distributed Computing System.”, IEEE Transactions on Computers. v. C-31. n. 1. p. 48-59, 1982.

Santoro, N. (2007) "Design and Analysis of Distributed Algorithms", Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Tanenbaum, A. S. e Steen, M. V. (2007) "Sistemas distribuídos: princípios e paradigmas", 2. ed. São Paulo: Pearson Prentice Hall, 2007.

Wilensky, U. (1999) "NetLogo", http://ccl.northwestern.edu/netlogo, Último Acesso em: Setembro de 2013.