Eleição de Líder com Qualidade de Serviço para o Modelo Falha-e-Recuperação

Vinícius Angiolucci Reis[1], **Gustavo Maciel Dias Vieira**[1*]

[1]DComp – CCGT – UFSCar
Sorocaba, São Paulo, Brasil

angiolucci@gmail.com, gdvieira@ufscar.br

Abstract. *Distributed systems use unreliable failure detectors to encapsulate the abstraction of time and to determine which processes have currently failed. Very often, these detectors are the basis for the creation of a leader election service. Many works are dedicated to analyze the quality of service (QoS) of failure detectors, but only few of them has analyzed the QoS of a leader election algorithm. In this work, we present the NFD-L leader election algorithm, designed to work on crash-recovery distributed systems and to follow the QoS specification defined by [Chen et al. 2002]. We used NFD-L to elect Paxos coordinators for a replication framework and compared the observed QoS for NFD-L with the behavior of the framework native leader election algorithm that is not designed to explicitly meet any QoS requirement.*

Resumo. *Um dos objetivos de um sistema distribuído é prover poder computacional e persistência de dados mesmo na presença de falhas de um subconjunto de enlaces e processos. Para determinar quais são os processos defeituosos deste sistema e abstrair o conceito de tempo, estes sistemas utilizam os serviços de um detector de falhas não confiável, encapsulado em um algoritmo de eleição de líder. Embora a literatura sobre a qualidade de serviço (QoS) oferecida por detectores de falhas seja abundante, ela é escassa quando se trata da QoS oferecida por algoritmos de eleição de líder. Nesta dissertação propomos um algoritmo de eleição de líder para o modelo falha-e-recuperação denominado NFD-L, que segue as especificações de QoS originalmente apresentadas em [Chen et al. 2002]. Utilizamos NFD-L em uma aplicação para replicação, como mecanismo de eleição de coordenador para Paxos e apresentamos uma análise da QoS observada, comparando o seu comportamento com um algoritmo de eleição de líder que não foi projetado explicitamente para prover garantias de QoS.*

1. Motivação

Determinar os processos defeituosos de um sistema distribuído é uma tarefa difícil, que resulta da impossibilidade de se diferenciar processos defeituosos de processos demasiadamente lentos [Fischer et al. 1985]. Esta dificuldade pode ser contornada com o uso de detectores de falhas *não confiáveis* [Chandra and Toueg 1996]. Mecanismos de replicação que utilizam consenso [Lamport 1998] dependem dos serviços de detectores de falhas

*Orientador

não confiáveis para a escolha de um processo distinto que coordena a replicação. Embora a não confiabilidade do detector de falhas não comprometa a correção destes algoritmos distribuídos [Guerraoui 2000], ela pode ter efeitos negativos em seu desempenho, pois o desempenho da replicação depende da velocidade com que o detector de falhas detecta e substitui um coordenador defeituoso, ao mesmo tempo que evita escolher como novo coordenador um processo defeituoso.

Como modo de limitar o comportamento eventual dos detectores de falhas não confiáveis em [Chen et al. 2002] os autores propuseram métricas para quantificar o seu comportamento em função de sua *velocidade* e *precisão*. Além disso, apresentaram algoritmos capazes de atender alguns requisitos de QoS, considerando o comportamento probabilístico da rede. Em [Nunes and Jansch-Porto 2004] os autores concluíram que a eficiência destes algoritmos depende, entre outros fatores, da correta predição do comportamento probabilístico da rede na forma da estimativa de chegada de mensagens futuras. Em [Sotoma and Madeira 2006] os autores utilizam cadeias de Markov para modelar um configurador que lida com perdas de mensagens em rajada. Em [Ma et al. 2010], os autores investigaram a aplicação das técnicas apresentadas por [Chen et al. 2002] para contemplar um modelo computacional onde os processos defeituosos podem voltar a operar corretamente. Em [Schiper and Toueg 2008] os autores utilizaram o algoritmo NFD-S proposto por [Chen et al. 2002] para criar dois algoritmos de eleição de líder, mas que necessita da existência de relógios sincronizados entre os processos.

Nesta dissertação nós apresentamos o algoritmo de eleição de líder NFD-L, baseado no algoritmo NFD-E apresentado em [Chen et al. 2002]. Nossos resultados apontam que NFD-L é um algoritmo de eleição de líder capaz de atender requisitos de QoS em uma rede com comportamento probabilístico conhecido. Além disso, nossa solução considera um modelo computacional onde os processos defeituosos podem voltar a operar e não depende da existência de relógios sincronizados entre os processos, uma suposição muito comum em sistemas distribuídos implementados na prática. Uma primeira versão desse trabalho foi publicada na trilha principal do SBRC 2017 [Reis and Vieira 2017]. A dissertação defendida apresenta uma especificação melhorada do algoritmo NFD-L e uma análise experimental mais completa [Reis 2017].

2. O algoritmo de eleição de líder NFD-L

NFD-L é um algoritmo de eleição de líder baseado no algoritmo de detecção de falhas NFD-E. Herda dele as etapas de configuração com base em requisitos de QoS e no comportamento probabilístico da rede. Como diferencial, expande seu modelo computacional para falha-e-recuperação e provê um serviço de eleição de líder, indicando um único processo como confiável. O algoritmo NFD-L está especificado no Algoritmo 1.

Em NFD-L um único processo atua como líder enviando mensagens periódicas de *heartbeat* aos demais processos, que são apenas observadores do líder. Os observadores esperam pela mensagem ou a ocorrência de um *timeout*. Este comportamento é idêntico ao algoritmo NFD-E. Quando um observador percebe um *timeout*, ele suspeitará que o processo líder falhou e irá iniciar a difusão dos próprios *heartbeats* para informar aos demais processos que é o novo líder. Se um segundo processo observador também suspeitar que o líder atual falhou, irá difundir seus *heartbeats* e os dois processos disputarão a liderança seguindo o algoritmo de *bully* [Garcia-Molina 1982]. Durante a disputa de

algoritmo 1 Algoritmo NFD-L

1: **procedimento** INICIALIZAÇÃO
2: $pid_{lder} \leftarrow \perp$; $uptime_{lder} \leftarrow 0$; $\ell \leftarrow 0$; $\tau_{\ell+1} \leftarrow 0$
3: $uptime_p \leftarrow 0$
4: $recupera(zerotime)$
5: **se** $zerotime = \perp$ **então**
6: $zerotime \leftarrow agora()$
7: $armazena(zerotime)$
8: **fim se**
9: $i \leftarrow \frac{agora()-zerotime}{\eta}$
10: **fim procedimento**
11:
12: **ao ocorrer** $i \cdot \eta \leq agora()$ **faça**
13: **se** $pid_p = pid_{lder}$ **então**
14: envia heartbeat m_i a todos os processos
15: **fim se**
16: $i \leftarrow i + 1$
17: $uptime_p \leftarrow uptime_p + 1$
18:
19: **ao ocorrer** recebimento de mensagem m_j de q **faça**
20: **se** $pid_{lder} = pid_q$ **então**
21: **se** $j > \ell$ e $uptime_q > uptime_{lder}$ e $pid_{lder} \neq pid_p$ **então**
22: $\ell \leftarrow j$
23: $uptime_{lder} \leftarrow uptime_q$
24: $\tau_{\ell+1} \leftarrow EA_{\ell+1} + \alpha$
25: **fim se**
26: **senão**
27: **se** $uptime_q > uptime_{lder}$ ou
$(uptime_q = uptime_{lder}$ e $pid_q > pid_{lder})$ **então** ▷ Desempate
28: $\ell \leftarrow j$
29: $pid_{lder} \leftarrow pid_q$
30: $uptime_{lder} \leftarrow uptime_q$
31: $\tau_{\ell+1} \leftarrow EA_{\ell+1} + \alpha$
32: $output \leftarrow pid_{lder}$
33: **fim se**
34: **fim se**
35:
36: **ao ocorrer** $\tau_{\ell+1} \leq agora()$ e $pid_{lder} \neq pid_p$ **faça**
37: $pid_{lder} \leftarrow pid_p$
38: $uptime_{lder} \leftarrow uptime_p$
39: $output \leftarrow pid_{lder}$

liderança, o processo com mais tempo sem falhas é selecionado, uma estratégia adotada originalmente em Treplica [Vieira and Buzato 2008].

Em NFD-L um processo confiará em um único líder em dado momento. Em momentos de disputa processos distintos poderão confiar em líderes distintos, mas nenhum observador ou processo disputante confiará em dois líderes simultaneamente. Uma vez agindo como líder, um processo enviará mensagens rotuladas m_i em intervalos de tempo iguais a η. Toda a configuração do detector de falhas e estimativa do comportamento probabilístico da rede seguem as definições do algoritmo NFD-E [Chen et al. 2002].

3. Validação Experimental

Para validar o comportamento do algoritmo NFD-L, o utilizamos como mecanismo de eleição de líder em um *framework* de replicação, Treplica [Vieira and Buzato 2008]. Comparamos então o comportamento de NFD-L com o algoritmo de eleição de líder já existente em Treplica. Este detector de falhas não é projetado para atender quaisquer requisitos de QoS, mas é um detector de falhas não confiável que permite que a replicação ativa aconteça à partir de várias rodadas de consenso. Seu funcionamento se dá pelo envio de *heartbeats* regulares a cada η unidades de tempo. O detector de falhas aguarda então $2 \cdot \eta$ para que o processo observado responda ao *heartbeat*, classificando o processo como defeituoso caso não receba uma resposta dentro deste intervalo de tempo.

O procedimento de configuração do algoritmo NFD-L é idêntico ao procedimento utilizado para o algoritmo NFD-E, como definido em [Chen et al. 2002]: é necessário que se observe o comportamento probabilístico da rede e se definam requisitos de QoS para encontrar valores compatíveis para η e α. Para comparar NFD-L com o algoritmo de eleição de líder existente em Treplica, configuramos ambos os algoritmos para enviar seus *heartbeats* em intervalos de tempo iguais a η.

Executamos dois conjuntos de experimentos. O primeiro conjunto, sem injeção de falhas, teve como objetivo medir a precisão dos detectores de falhas em termos das métricas de tempo entre enganos (T_{MR}) (Figura 1) e tempo para correção de engano (T_M) (Figura 2). O segundo conjunto, teve como objetivo medir a velocidade dos detectores de falhas. Falhas controladas foram injetadas no sistema para medir o tempo de detecção de falha (T_D) (Figura 3) e tempo de detecção de recuperação (T_{DR}) [Ma et al. 2010] (Figura 4) de cada detector de falhas. Nas figuras, as linhas pontilhadas representam o limite máximo para os requisitos de QoS.

4. Conclusão

Nesta dissertação nós apresentamos um algoritmo de eleição de líder, NFD-L, modelado a partir do algoritmo de detecção de falhas NFD-E originalmente apresentado por [Chen et al. 2002]. Utilizamos o algoritmo NFD-L em uma aplicação baseada em Paxos com uma alta carga propositalmente inserida para analisar o atendimento aos requisitos de QoS em uma ambiente sob *stress*. Pudemos verificar a diferença nítida comportamental do algoritmo NFD-L em relação ao algoritmo de eleição de líder já existente na aplicação de replicação utilizada. O algoritmo NFD-L apresentou um comportamento menos variado e foi capaz de atender às métricas de QoS.

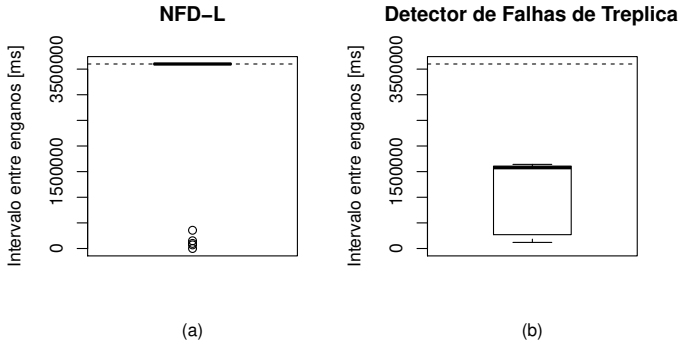

Figura 1. Intervalo entre enganos

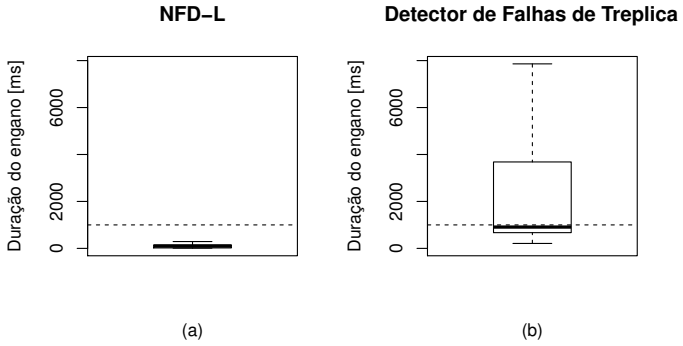

Figura 2. Duração de enganos

Referências

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267.

Chen, W., Toueg, S., and Aguilera, M. (2002). On the quality of service of failure detectors. *Computers, IEEE Transactions on*, 51(5):561–580.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382.

Garcia-Molina, H. (1982). Elections in a distributed computing system. *IEEE Trans. Comput.*, 31(1):48–59.

Guerraoui, R. (2000). Indulgent algorithms (preliminary version). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '00, pages 289–297, New York, NY, USA. ACM.

Lamport, L. (1998). The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169.

Ma, T., Hillston, J., and Anderson, S. (2010). On the quality of service of crash-recovery failure detectors. *IEEE Trans. Dependable Secur. Comput.*, 7(3):271–283.

Nunes, R. C. and Jansch-Porto, I. (2004). QoS of timeout-based self-tuned failure detec-

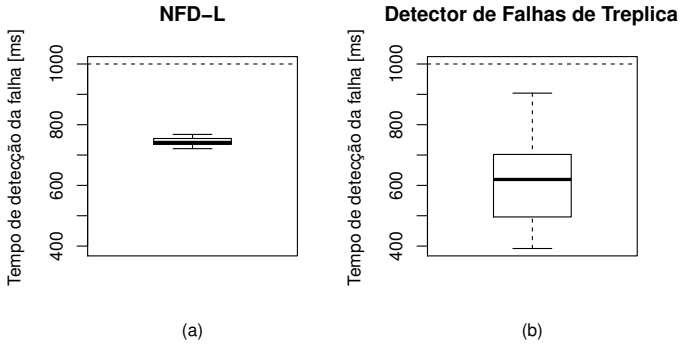

Figura 3. Tempo de detecção de falha

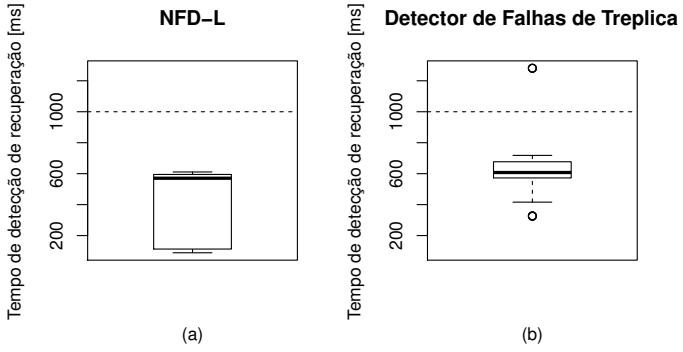

Figura 4. Tempo de detecção de recuperação

tors: The effects of the communication delay predictor and the safety margin. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, DSN '04, pages 753–, Washington, DC, USA. IEEE Computer Society.

Reis, V. A. (2017). Eleição de líder com qualidade de serviço para o modelo falha-e-recuperação. Master's thesis, Universidade Federal de São Carlos, Sorocaba, Brasil.

Reis, V. A. and Vieira, G. M. D. (2017). Quality of service of an asynchronous crash-recovery leader election algorithm. In *SBRC '17: Proc. of the 35th Brazilian Symposium on Computer Networks and Distributed Systems*, pages 1089–1102, Belém, Brazil.

Schiper, N. and Toueg, S. (2008). A robust and lightweight stable leader election service for dynamic systems. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, pages 207–216. IEEE.

Sotoma, I. and Madeira, E. R. M. (2006). A markov model for providing quality of service for failure detectors under message loss bursts. Technical Report IC-06-013, Institute of Computing, University of Campinas.

Vieira, G. M. D. and Buzato, L. E. (2008). Treplica: Ubiquitous replication. In *SBRC '08: Proc. of the 26th Brazilian Symposium on Computer Networks and Distributed Systems*, Rio de Janeiro, Brasil.