Implementação de Difusão Atômica Baseada em Diagnóstico com Testes Imperfeitos

Edson Tavares de Camargo[1,2], **Elias P. Duarte Jr.**[2], **Weyne Cassou Pietniczka**[2]

[1]Universidade Tecnológica Federal do Parará - Câmpus Toledo (UTFPR)
CEP: 85902-490 – Toledo – PR – Brasil

[2]Universidade Federal do Paraná (UFPR) – Programa de Pós-Graduação em Informática
Caixa Postal 19081 – 81531-980 – Curitiba – PR – Brasil

edson@utfpr.edu.br, elias@inf.ufpr.br, wcp10@inf.ufpr.br

Abstract. *This paper describes two implementations of atomic broadcast, the first implementation is based on the Paxos consensus algorithm, while the other assumes a stable sequencer. The system is assumed to be composed of both unstable and stable processes. Stable processes are obtained and identified with a novel system-level diagnosis model which is based on imperfect tests. The performance of the strategy used to select stable nodes, as well as an experimental comparison of the two atomic broadcast strategies is shown though simulation.*

Resumo. *Este trabalho descreve duas implementações da difusão atômica em um conjunto de processos identificados como estáveis. A primeira implementação da difusão atômica é baseada no algoritmo de consenso Paxos. A segunda implementação assume um processo estável como sequenciador fixo. Os processos estáveis são identificados e mantidos por meio de uma estratégia de testes definida em um novo modelo de diagnóstico em nível de sistema que se apoia em testes ditos imperfeitos. Este trabalho descreve a implementação da estratégia de testes. Resultados de simulação mostram o desempenho da estratégia de seleção de nodos estáveis, bem como uma comparação experimental do número de mensagens e a latência das duas implementações da difusão atômica.*

1. Introdução

Identificar unidades que estão funcionando corretamente em um sistema e de acordo com a sua especificação é o objetivo do diagnóstico em nível de sistema (*system-level diagnosis*) [Duarte et al. 2011]. O diagnóstico é baseado no resultado de testes executados entre as unidades do sistema (neste trabalho usamos o termo "processo" como sinônimo de unidade). O primeiro modelo de diagnóstico é o PMC [Preparata et al. 1967], o qual assume que uma unidade sem-falha determina com precisão o estado de outra unidade testada.

Este trabalho apresenta duas implementações de difusão atômica as quais utilizam uma estratégia de monitoramento que se apoia em um novo modelo de diagnóstico onde os testes são, por definição, imperfeitos. A estratégia permite encontrar e manter um *núcleo de processos estáveis*. Em particular, um teste executado sobre um processo que não sofreu falha pode não indicar que está correto. Processos que executam os testes e

que se classificam como estáveis entre si é que formam o núcleo estável. Desta forma, definimos um processo como estável quando todos os testes executados sobre este processo indicam que é correto. Portanto, mesmo sendo imperfeitos, os testes definem o critério de estabilidade a ser aplicado no sistema.

A estratégia de monitoramento permite que processos classificados como instáveis apenas por um curto período de tempo retornem ao sistema. Um processo instável testado como sem-falha por ζ testes consecutivos por algum processo estável pode ter a sua classificação modificada após uma rodada de consenso entre os processos do núcleo de processos estáveis. Dessa forma, o núcleo estável decide se reintegra o processo classificado como instável.

A difusão atômica é um serviço distribuído fundamental e o seu objetivo é garantir que todos os processos corretos entreguem o mesmo conjunto de mensagens na mesma ordem total [Defago et al. 2004]. Este trabalho implementa a difusão atômica considerando duas abordagens, conforme [Defago et al. 2004]. Na primeira abordagem, a ordem da entrega das mensagens é definida através de sucessivas instâncias de consenso. O consenso, a grosso modo, permite que um conjunto de processos corretos concordem sobre um valor comum com base em valores propostos inicialmente. A segunda abordagem implementada para difusão atômica se apoia em um sequenciador fixo, ou seja, em um processo que assume a função de definir a ordem de entrega das mensagens. Nessa abordagem duas variantes são implementadas: a UB (*unicast-broadcast*) e a BB (*broadcast-broadcast*). O algoritmo de consenso Paxos foi implementado tanto na difusão atômica baseada no consenso quanto na reintegração de um nodo instável.

A difusão atômica implementada neste trabalho considera que somente os processos que compõem o núcleo de processos estáveis enviam suas mensagens. Dessa forma, as mensagens são entregues na mesma ordem a todos os processos do núcleo estável. Para mostrar a viabilidade da nossa proposta, resultados de simulação da estratégia de monitoramento e das abordagens de difusão atômica são apresentados. Os resultados de simulação da estratégia de monitoramento visam apresentar a latência da obtenção e manutenção do núcleo estável em diferentes números de nodos. Os resultados de simulação da difusão atômica apresentam o número de mensagens e a latência das duas abordagens da difusão atômica implementada sobre o núcleo estável. Ao final, as duas abordagens de difusão atômica implementadas são comparadas quanto a latência e o número de mensagens.

O trabalho de iniciação científica consistiu da especificação e implementação dos algoritmos de difusão atômica sobre o novo modelo de diagnóstico baseado em testes imperfeitos. O trabalho realizado incluiu também a implementação do algoritmo de consenso Paxos. A estratégia de monitoramento incluindo a reintegração de um nodo instável ao núcleo estável também foi implementada no contexto da iniciação científica. A iniciação científica desenvolvida está inserida em um projeto maior, em desenvolvimento, que tem por objetivo construir o novo modelo de diagnóstico baseado em testes imperfeitos que visa identificar instabilidade em sistemas.

Este trabalho está organizado da seguinte forma. A Seção 2 apresenta o modelo de sistema e a estratégia de monitoramento. A Seção 3 realiza a conceituação da difusão atômica e apresenta o algoritmo Paxos. A Seção 4 aborda a implementação. A Seção 5

apresenta os resultados experimentais. A Seção 6 apresenta a conclusão.

2. Modelo de Sistema

O modelo de sistema assume um sistema S representado por um grafo completo $G = (V, E)$, onde V é o conjunto de N vértices e E o conjunto de arestas, sendo que existe uma ligação direta entre quaisquer dois vértices (n_i, n_j). Neste trabalho, nos referimos a um vértice n_i também como nodo i. A ligação entre dois nodos se dá por um canal confiável. Cada nodo i assume um de dois estados, *estável*, ou *instável*. Um *evento* é definido como a troca de estado de um nodo, de estável para instável ou vice-versa. O modelo é assíncrono, isto é, não há premissas temporais. O modelo de falhas de processos é o modelo *crash*.

A estabilidade de um nodo é definida pelo fato de que nenhum outro nodo que o testa o considera instável. Para ser considerado estável, ou sem-falha, o nodo precisa responder a requisições de acordo com a sua especificação e dentro de um limite de tempo. As informações sobre os resultados de testes são propagadas entre os processos, permitindo que um núcleo de *processos estáveis* se forme.

A estratégia de testes deste trabalho é definida a seguir. Um nodo i ao testar um nodo j como estável recebe de j as informações sobre os estados dos nodos testados por j. Sendo assim, se o nodo j testou um nodo k como instável, essa informação chegará a i. Mesmo que o nodo i considere o nodo k estável atualizará o estado de k devido à nova informação recebida de j. Dessa forma, o nodo k é excluído do núcleo de processos estáveis. A Figura 1 apresenta um grupo de oito processos identificados de 0 a 7 onde os nodos 0, 2, 5 e 7 formam um núcleo estável. Os demais nodos não fazem parte do núcleo estável pois foram identificados como instáveis por pelo menos um nodo estável.

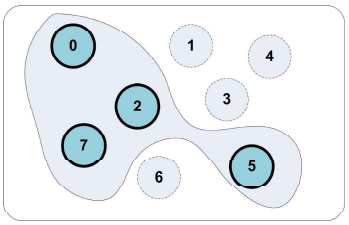


Figura 1. Núcleo estável formado pelos processos 0, 2, 5 e 7.

Uma *rodada de testes* é caracterizada como o período de tempo no qual cada nodo realiza seus testes. A latência do algoritmo é definida como o tempo necessário para que nodos estáveis recebam informações sobre um determinado evento. Diferentemente do modelo PMC [Preparata et al. 1967], que assume que todo nodo sem-falha identifica corretamente o estado do nodo testado, na nossa estratégia um teste não é perfeito, ou seja, é possível que o testador se engane em relação ao estado do nodo testado. Concretamente, isso se deve à impossibilidade de aguardar indefinidamente pela resposta a um teste executado, pois um nodo i que deixa de responder a requisições de um nodo j não necessariamente se encontra falho. O limite de tempo em que um nodo deve aguardar pela resposta de teste pode ser calculado de forma adaptativa de acordo com o algoritmo de Van Jacobson usado no protocolo TCP [Jacobson and Karels 1988].

Na estratégia proposta, inicialmente o estado de cada nodo é desconhecido. Cada nodo testa todos os demais nodos a cada intervalo. Um nodo j propaga a sua visão

sobre os estados dos demais nodos do sistema quando o nodo i o testa como estável. Um nodo que se mantém instável na avaliação de pelo menos um nodo pode ter o seu estado modificado para estável após uma rodada de consenso envolvendo todos os nodos estáveis. Um nodo i ao testar o nodo k como estável por um número consecutivo ζ de rodadas de testes pode acionar o algoritmo de consenso a fim de reingressar o nodo k ao núcleo de processos estáveis. Assume-se que processos não são reincorporados ao núcleo estável até completar o consenso.

3. Paxos e Difusão Atômica: Conceituação

O consenso e a difusão atômica são problemas fundamentais para construção de sistemas distribuídos confiáveis. Abaixo uma breve conceituação teórica sobre a difusão atômica e o algoritmo de consenso Paxos.

3.1. Difusão Atômica

A difusão atômica, (*atomic broadcast* ou *total order broadcast*), tem por objetivo garantir que todos os processos corretos entreguem o mesmo conjunto de mensagens na mesma ordem total. A difusão atômica é composta pelas três propriedades da difusão confiável (*reliable broadcast*) mais a propriedade de ordem total. As propriedades da difusão confiável asseguram que uma mensagem enviada por um processo correto será entregue, sem duplicação, a todos os demais processos corretos. Se um processo falhar após ter difundido uma mensagem, a mensagem será entregue a todos os demais se pelo menos um processo tiver entregue a mensagem. A propriedade de ordem total define que se dois processos corretos *p* e *q* entregam duas mensagens *m* e *m'*, então *p* entrega *m* antes de *m'* se e somente se *q* entrega *m* antes de *m'*, [Defago et al. 2004].

A difusão atômica é um serviço fundamental para manter a consistência em base de dados replicadas onde o comportamento pode ser modelado por uma máquina de estados. O trabalho de [Defago et al. 2004] apresenta diversas abordagens para a implementação da difusão atômica. Neste trabalho, duas dessas abordagens foram implementadas. A primeira é baseada em consenso. Nesse caso, a ordenação das mensagens é realizada por consecutivas execuções do consenso pelos processos com base nas mensagens recebidas.

A segunda abordagem considera que um processo é responsável por definir a sequencia das mensagens. Nessa abordagem foram implementadas duas variantes: a UB (*unicast-broadcast*) e a BB (*broadcast-broadcast*). Na variante UB, um processo envia sua mensagem ao sequenciador, que insere na mensagem uma etiqueta com seu número de sequência e a envia aos demais processos. Na variante BB, o processo envia sua mensagem a todos os processos. O sequenciador, ao receber a mensagem, envia aos demais apenas o número de sequencia daquela mensagem e não a mensagem em si. A segunda variante apesar de gerar mais mensagens diminui a carga sobre o sequenciador.

A abordagem baseada no consenso foi implementada através do algoritmo de consenso Paxos, descrito a seguir.

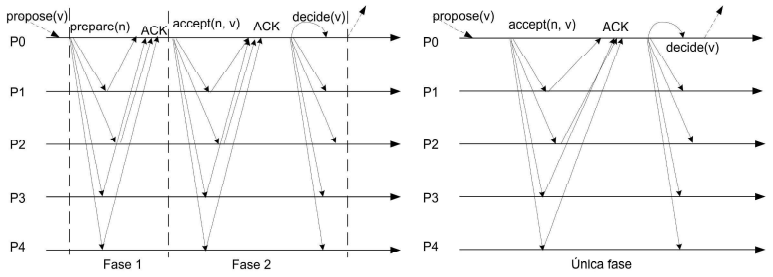
3.2. O Consenso e o Algoritmo de Paxos

O consenso, a grosso modo, permite que um conjunto de processos corretos concordem sobre um valor comum com base em valores propostos inicialmente, considerando que esses valores iniciais podem ser diferentes para cada processo. O consenso está alicerçado

em quatro propriedades, sendo uma propriedade de progressão (*liveness*) e três propriedades de segurança (*safety*): terminação, validade, integridade e acordo. A terminação assegura que cada processo correto em algum momento decide por algum valor, sendo esta a propriedade que garante a progressão. A validade significa que se um processo decide por um valor *v*, então *v* foi proposto por algum processo. A propriedade de integridade determina que nenhum processo decide duas vezes e a propriedade de acordo afirma que dois processos corretos não decidem valores diferentes [Guerraoui et al. 2011].

Paxos é um algoritmo de consenso projetado inicialmente para ser executado no contexto de replicação [Lamport 2001]. O algoritmo de consenso Paxos segue o modelo assíncrono. O modelo de falhas assumido é o de parada com recuperação (*crash-recovery*). Dessa forma, o algoritmo considera que tanto a execução dos processos quanto a entrega das mensagens possuem velocidades arbitrárias e que os processos podem reiniciar sua execução após uma falha por parada. Os processos assumem as seguintes funções no algoritmo: *proposers*, *acceptors* e *learners*. Os *proposers* propõem um valor, os *acceptors* escolhem um valor e os *learners* aprendem o valor decidido. Um processo pode assumir qualquer uma dessas funções e múltiplas funções simultaneamente.

A execução do Paxos ocorre em rodadas. Uma rodada é identificada por um número inteiro positivo *n*. Cada processo p_i possui um identificador único *i* que é acrescido ao número de rodada *n*. O consenso acontece em duas fases (Figura 2(a)). Basicamente, na primeira fase o processo *proposer* tenta garantir que o seu valor será escolhido solicitando ao conjunto de *acceptors* através de uma requisição *prepare(n)* que não aceitem um número de rodada menor do que *n*. Após obter a resposta afirmativa da maioria através de *ACKs*, o *proposer* segue para a segunda fase. Na segunda fase o *proposer* submete um valor para ser decidido juntamente com o número de rodada aceita na fase anterior por meio de uma solicitação *accept(n,v)*. Ao aceitar a proposta da segunda fase, os *acceptors* comunicam os *learners* sobre o valor aceito.



(a) Fases do Paxos (b) Paxos em uma fase

Figura 2. Algoritmo Paxos

Para evitar que os *acceptors* precisem comunicar diversos *learners*, um processo *learner* distinto pode ser escolhido para receber a decisão. Os demais *learners* podem ser informados posteriormente por esse *learner*. Na Figura 2(a), o processo *propose* e o processo *learner* são os mesmos.

Para garantir a progressão do algoritmo e evitar um cenário onde dois ou mais *proposers* concorrem indefinidamente pela escolha do seu valor, um processo líder é escolhido. Somente o líder pode submeter valores. Em [Lamport 2001] é descrito um cenário de otimização do Paxos onde o líder permanece o mesmo durante múltiplas rodadas de

consenso (Figura 2(b)). Essa otimização permite que o consenso seja realizado somente executando a segunda fase.

Em relação a implementação do algoritmo Paxos, destaca-se o Ring-Paxos [Marandi et al. 2014]. O Ring-Paxos é um protocolo para difusão atômica derivado do Paxos e projetado para atingir altas taxas de transferência (*throughput*). A partir do Ring-Paxos dois outros protocolos são propostos: o M-Ring Paxos e o U-Ring Paxos.

4. Implementação

Neste trabalho, a difusão atômica é implementada sobre um núcleo de processos estáveis, conforme apresenta a Figura 3. Uma das abordagens implementadas faz uso do algoritmo de consenso Paxos. O algoritmo Paxos também foi usado no contexto da estratégia de testes para decidir se um nodo instável reintegra o núcleo estável.

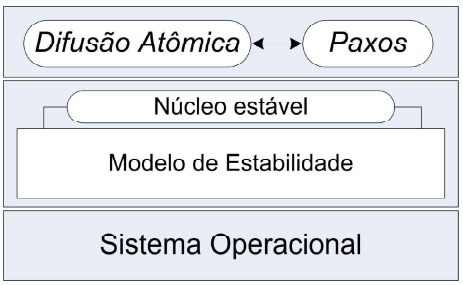


Figura 3. Arquitetura do sistema implementado.

O algoritmo Paxos é composto pelas primitivas *propose(v)* e *decide(v)*, onde v é um valor arbitrário. A difusão atômica é definida pelas primitivas *broadcast(m)* e *deliver(m)*. Todas as primitivas implementadas consideram apenas os processos do núcleo estável. Dessa forma, a primitiva *broadcast(m)*, por exemplo, funciona como uma primitiva *multicast(m)*, onde as mensagens são enviadas apenas a processos do núcleo estável. A difusão atômica foi implementada conforme a descrição a seguir.

Cada processo p_i do núcleo estável envia sua mensagem aos demais processos do núcleo estável via *broadcast(m)*. As mensagens recebidas pelos processos do núcleo estável são armazenadas em um vetor de mensagens recebidas. Há uma constante chamada *k* que determina a quantidade de mensagens recebidas que o processo deve aguardar para então invocar o consenso. O líder do algoritmo de consenso será o primeiro nodo do núcleo estável ao atingir a quantidade de mensagens recebidas igual a *k*. Constantemente cada processo verifica se há uma rodada de consenso em andamento.

Uma rodada de consenso inicia quando o líder envia uma mensagem *prepare(n)*, de acordo com a Figura 2, onde *n* é o número da rodada do consenso. Se o líder receber resposta da maioria dos processos do núcleo estável e não receber uma resposta *NACK* como resultado do *prepare(n)* poderá então propor o seu valor através da primitiva *propose(v)*, onde *v* é composto pela sequencia dos identificadores das mensagens recebidas pelo líder. Caso o líder receba um *NACK* como resposta do *prepare(n)* deverá então atualizar o seu valor da rodada *n* para $n + x * q$, onde *q* é a quantidade total de processos e *x* o número da rodada. O novo valor da rodada deverá ser maior que o número da rodada recebido no *NACK*. Após atualizar o valor da rodada, o processo líder envia uma nova solicitação *prepare(n)*. Uma vez que a maioria dos processos do núcleo estável aceita

o *propose(v)* do líder, então o líder comunica a decisão com uma mensagem *decide(v)* a todos os processos do núcleo estável. Ao finalizar a rodada de consenso, os processos do núcleo estável realizam a entrega das mensagens conforme o vetor de mensagens recebidas do líder do consenso.

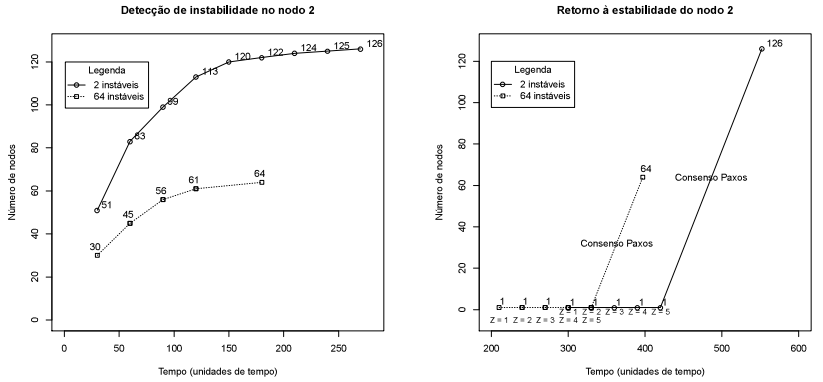
Na segunda abordagem da difusão atômica implementada, o primeiro processo do núcleo estável assume a função de sequenciador das mensagens.

5. Resultados Experimentais

A seguir são apresentados dois conjuntos de resultados. O primeiro se refere à simulação da estratégia de testes. O segundo conjunto de resultados é da difusão atômica executada sobre um núcleo de processos estáveis. A simulação é efetuada através da linguagem SMPL (*Simple Portable Simulation Language*) [MacDougall 1987].

5.1. Obtenção do Núcleo Estável

Nesta primeira simulação o objetivo é apresentar a formação e manutenção do núcleo estável respectivamente na Figura 4(a) e na Figura 4(b). O gráfico da Figura 4 apresenta dois cenários de execução com 128 nodos. O primeiro cenário considera a execução com 2 nodos instáveis e o segundo considera 64 nodos instáveis. Em ambos os cenários, um nodo, o nodo 2, será reincorporado ao núcleo estável, conforme a Figura 4(b). O retorno de um nodo ao núcleo estável ocorre com $\zeta = 5$. A partir de então todos os nodos alteram o estado do nodo instável para estável. Na simulação os nodos executam a estratégia de testes a cada 30 unidades de tempo. Os nodos se tornam instáveis no tempo 20.



(a) Detecção da instabilidade (b) Retorno do nodo à estabilidade

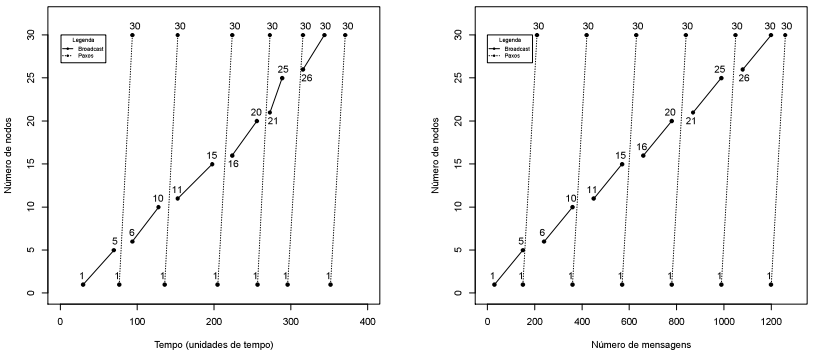
Figura 4. Simulação com 128 nodos.

No cenário com 2 nodos instáveis (linha contínua) no tempo 30, 51 nodos detectam a instabilidade. Na rodada de testes seguinte, são 83 nodos que detectam os nodos instáveis. Por fim, no tempo 270, todos os 126 nodos estáveis detectam que os 2 nodos estão instáveis. No tempo 300 (Figura 4(b)), um nodo detecta que um dos nodos instáveis se comporta como estável. Então, após 5 rodadas de testes sequenciais recebendo retorno aos testes executados, o nodo estável chama o algoritmo de consenso envolvendo o núcleo estável no tempo 420 para decidir se o estado do nodo antes instável deve ser modificado para estável. Na sequência o nodo instável tem o seu estado modificado para estável por

todos os processos estáveis. O tempo da execução do consenso e atualização do estado do nodo pelo núcleo estável é indicado na linha diagonal da Figura 4(b). No cenário com 64 nodos instáveis (linha pontilhada), os nodos estáveis identificam em menos tempo os nodos instáveis uma vez que há apenas 64 nodos estáveis.

5.2. Difusão Atômica

Nesta simulação o objetivo é apresentar a latência e o número total de mensagens das implementações da difusão atômica. Nos resultados a seguir os nodos iniciam a transmissão de suas mensagens na unidade de tempo 30 após a formação do núcleo estável. Todos os nodos estáveis enviam uma mensagem entre si. Assume-se também que as mensagens enviadas sofrem atraso, porém não são perdidas. Os processos podem apresentar velocidades de execução diferentes e permanecem estáveis durante a execução da difusão atômica. Em todos os cenários há dois processos instáveis.



(a) Latência para k=5 (b) Número de mensagens para k=5

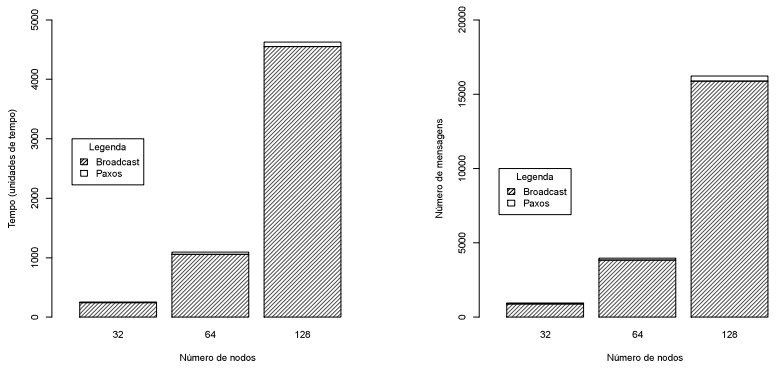
Figura 5. Difusão atômica com 32 nodos, 2 nodos instáveis e k = 5.

O primeiro gráfico, Figura 5, é de um cenário com 32 nodos e $k = 5$. Conforme descrito anteriormente, na Seção 4, a constante k define a quantidade de mensagens que um processo deve receber para então iniciar uma rodada de consenso. As Figuras 5(a) e 5(b) apresentam a latência e o número de mensagens respectivamente considerando as sucessivas instâncias de consenso. O gráfico destaca a latência e o número de mensagens da difusão (*broadcast*) e a latência e o número de mensagens do consenso (Paxos).

O gráfico da Figura 5(a), informa que o consenso, em cada execução, leva aproximadamente 16 unidades de tempo e a transmissão das mensagens em media 33 unidades de tempo. A latência total foi de aproximadamente 380 unidades de tempo. A Figura 5(b), que apresenta o número de mensagens, informa que após 5 nodos estáveis terem transmitidos suas mensagens o consenso é chamado. Como são 30 processos do núcleo estável, 150 mensagens são transmitidas ao todo até a execução do consenso. O consenso em si demanda 60 mensagens a cada execução.

A Figura 6 apresenta a latência e o número de mensagens da difusão atômica em cenários com 32, 64 e 128 nodos totais com a constante k igual ao número de processos estáveis. Também há a distinção entre a latência e o número de mensagens da difusão (*broadcast*) e do consenso (Paxos). De acordo com a Figura 6(a), percebe-se que 258

unidades de tempo são necessárias para a execução completa da difusão atômica pelo núcleo estável para o cenário de 32 nodos, enquanto que 1097 unidades são requisitadas no cenário de 64 nodos e 4627 para o cenário de 128 nodos. Como pode ser visto, o custo do consenso para a latência total da difusão atômica é mínimo.



(a) Latência (b) Número de mensagens

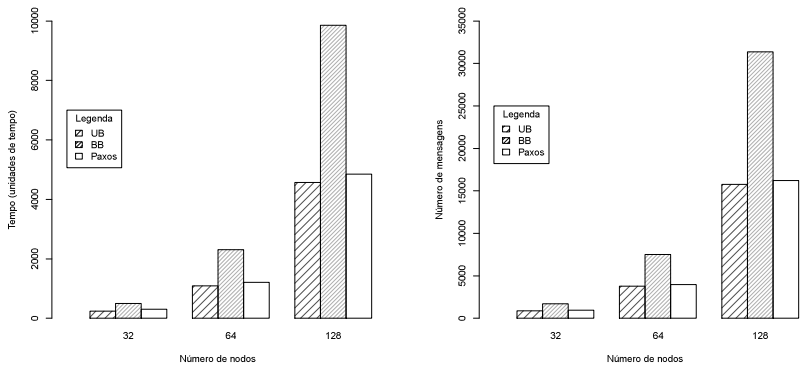
Figura 6. Difusão atômica baseada em consenso.

Em relação ao número de mensagens, Figura 6(b), conclui-se que 960 mensagens totais são trocadas pelo núcleo na execução da difusão atômica no cenário de 32 nodos, 3968 mensagens são necessárias no cenário de 64 nodos e 16218 são trocadas no cenário de 128 nodos. Considerando somente o Paxos, o número de mensagens enviadas chega a 60, 124 e 342 para os cenários de 32, 64 e 128 nodos, respectivamente.

Conforme pode ser observado, a latência e o número de mensagens nos cenários apresentados cresce consideravelmente à medida que aumenta o número de nodos do sistema. Vale observar que os cenários simulados consideram o pior caso, onde todos os nodos enviam uma mensagem a todos os demais.

A Figura 7(a) e a Figura 7(b) fazem um comparativo da latência e do número de mensagens totais da difusão atômica implementada através das três abordagens: UB, BB e consenso (k igual número de processos estáveis). Os cenários comparados são os de 32, de 64 e de 128 nodos totais, considerando 2 nodos instáveis. Em relação a latência (Figura 7(a)), percebe-se que a variante UB do sequenciador fixo e o consenso caminham muito próximas entre si. No entanto a latência da variante BB do sequenciador fixo aumenta consideravelmente a medida que cresce o número de nodos totais. Em números, no contexto de 128 nodos, a latência da abordagem BB chega ao patamar de 9854 unidades de tempo, o dobro da latência das abordagens UB (4564 unidades de tempo) e consenso (4849 unidades de tempo).

Em relação ao número de mensagens nas três abordagens (Figura 7(b)), verifica-se que o mesmo mais do que quadruplica a medida que o número de nodos aumenta. Em termos numéricos, a abordagem UB começa com 870 mensagens no contexto de 32 nodos, sobe para 3782 ao considerar 64 nodos e encerra com 15750 mensagens no contexto de 128 nodos. Já a abordagem BB inicia com 1711 mensagens com 32 nodos, aumenta para 7503 com 64 nodos e finaliza com 31375 mensagens trocadas ao considerar 128 nodos. O consenso tem número de mensagens próxima a variante UB.



(a) Latência (b) Número de mensagens

Figura 7. Comparação das abordagens de difusão atômica.

6. Conclusão

Neste trabalho, apresentamos resultados de implementação e de simulação de duas abordagens da difusão atômica: uma baseada em consenso e a outra em sequenciador fixo. A difusão atômica baseada em consenso fez uso do algoritmo de consenso Paxos. Também foi apresentada a implementação e simulação da estratégia de testes para obtenção do núcleo de processos estáveis proveniente de um novo modelo de diagnóstico em nível de sistema.

As especificações e implementações desenvolvidas para os algoritmos de difusão atômica no novo modelo se deram no contexto do trabalho de iniciação científica. A proposição, especificação e implementação do modelo de diagnóstico continua em desenvolvimento em um projeto maior.

Referências

Defago, Schiper, and Urban (2004). Total order broadcast and multicast algorithms: Taxonomy and survey. *CSURV: Computing Surveys*, 36.

Duarte, E. P., Ziwich, R. P., and Albini, L. C. P. (2011). A survey of comparison-based system-level diagnosis. *ACM Comput. Surv*, 43(3):22.

Guerraoui, R., Cachin, C., and Rodrigues, L. (2011). *Introduction to Reliable and Secure Distributed Programming (2. ed.)*. Springer.

Jacobson, V. and Karels, M. J. (1988). Congestion avoidance and control. *ACM Computer Communications Review*, 18(4):314–329.

Lamport (2001). Paxos made simple. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 32.

MacDougall, M. H. (1987). *Simulating Computer Systems. Techniques and Tools*. Computer Systems Series. MIT. Discrete Event Simulation mittels SMPL.

Marandi, P. J., Primi, M., Schiper, N., and Pedone, F. (2014). Ring paxos: High-throughput atomic broadcast. *CoRR*, abs/1401.6015.

Preparata, Metze, and Chen (1967). On the connection assignment problem of diagnosable systems. In *IEEE Transactions on Electronic Computers*, volume 16.