Desvendando RingPaxos: Um protocolo para alta taxa de transferência de dados e difusão atômica em sistemas distribuídos.

Cristian Cleder Machado[1], Bruno Augusti Mozzaquatro[2], André Luis Stefanello[1], Maurício Sulzbach[1], Thiago Roberto Sarturi[1]

[1]Universidade Regional Integrada (URI)– Câmpus de Frederico Westphalen
Departamento de Engenharias e Ciência da Computação
Frederico Westphalen – RS – Brasil – 98.400-000

[2]Universidade Federal de Santa Maria (UFSM)
Colégio Politécnico da UFSM
Santa Maria – RS – Brasil – 97.105-900

{cristian,andres,sulzbach,tsarturi}@uri.br,brunomozza@inf.ufsm.br

Abstract. *Distributed Systems are typically complex structures that may present several faults possibilities, which makes it possible to ensure fault tolerance issues nontrivial. Moreover, another important issue is to maintain the performance of the system quickly and efficiently. In this context, this paper proposes to present a set of results, analyzes and experiments with the Ring Paxos Protocol. Ring Paxos is a solution used to solving consensus problems and implementing atomic broadcast in a unreliable processors network at a data-rate close to 900Mbit/s per Learner. As a results, are presented and discussed graphics of the experiments showing good results in the protocol executions on heterogeneous structures, together with the process identifications do not implemented and, problems and shortcomings of the protocol.*

Resumo. *Sistemas distribuídos são, geralmente, estruturas complexas que podem apresentar diversas possibilidades de falhas, o que torna a possibilidade de assegurar questões de tolerância à falhas uma tarefa não trivial. Além disso, outra questão importante é de manter o desempenho do sistema de forma rápida e eficiente. Neste contexto, este trabalho propõe apresentar um conjunto de verificações, análises e experimentos com o protocolo RingPaxos, uma solução usada para resolver problemas de consenso e implementar difusão atômica em uma rede de nós-processadores não-confiáveis numa taxa próxima à 900Mbit/s de dados para cada Learner. Como resultados, são apresentados e discutidos gráficos dos experimentos mostrando bons resultados na execução do protocolo em estruturas heterogêneas, juntamente com a identificação de processos não implementados, problemas e falhas do protocolo.*

1. Introdução

Sistemas distribuídos são, geralmente, estruturas com alto nível de complexidade, atribuída ao conjunto de componentes, aplicações, módulos, escalabilidade e comunicação. A partir dessa complexidade, visto que cada um dos componentes do sistema, em geral, podem ter diversas possibilidades de falhas, assegurar a tolerância à falhas nesses

sistemas é uma tarefa não trivial [Cristian 1991]. Outra questão importante está relacionada com a manutenção do desempenho do sistema, de forma rápida e eficiente. Este objetivo está sujeito a variações devido ao contexto de tarefas e/ou serviços, distribuídos em máquinas distintas, trocando mensagens através de uma rede de comunicação [Moser 1996, Melliar-Smith 1990, Marandi 2011]. Testes realizados manualmente para identificação de todas as possíveis combinações de erros/falhas e suas possíveis soluções tornam-se, na maioria dos casos, exaustivas sobrecargas de trabalho, ostentando ser um procedimento bastante inviável. Para tanto, há uma clara necessidade de implementar um meio onde haja a possibilidade de controle de possíveis tarefas de forma (mais) eficiente e (semi)automática [Neiger 1988].

Neste contexto, este trabalho propõe apresentar um conjunto de verificações, análises e experimentos com o protocolo RingPaxos. As verificações e análises abrangem diferentes aspectos, tais como, quantidade de nós, heterogeneidade de hardware, escalabilidade, entre outros. A realização da adição, remoção e parada proposital de processos executados em nós distintos, também é foco deste trabalho e, tende a explorar à proposta do protocolo de eficiência, escalabilidade e confiabilidade na comunicação frente à problemas de consenso e implementação de difusão atômica. Um *benchmark* e ferramentas para medição de desempenho são utilizadas nos experimentos para validação da proposta de altas taxas de transferência do protocolo. Como resultados, são apresentados e discutidos gráficos dos experimentos realizados e a identificação de processos não implementados, problemas e falhas do protocolo.

O artigo está organizado da seguinte forma: Na Seção 2 é exposta uma contextualização dos mecanismos, abordagens e técnicas envoltas ao protocolo apresentado. Na Seção 3 o protocolo RingPaxos é detalhado. As estruturas, arquiteturas e configurações utilizadas para os experimentos são exibidos na Seção 4. Na Seção 5 os resultados dos experimentos são discutidos. Alguns trabalhos que serviram como ponto de partida para este artigo são apresentados e discutidos na Seção 6. Por fim, as considerações finais e a sugestão de trabalhos futuros é descrita na Seção 7.

2. Contextualização

Para fins de entendimento da proposta estabelecida pelo protocolo RingPaxos, esta seção apresenta um resumo dos conceitos que influenciam no contexto de sua execução. Sucintamente, são embasadas as características de Sistemas Distribuídos, Tolerância à Falhas, Problema de Consenso, Difusão Atômica, Multicast, Unicast e Topologias em Anel.

Segundo [Coulouris 2007] um Sistemas Distribuídos é *"Um sistema no qual os componentes de hardware ou software, localizados em computadores interligados em rede, se comunicam e coordenam suas ações apenas trocando mensagens entre si"*. Em [Tanenbaum 2002] , o conceito é apresentado como *"...um conjunto de computadores independentes entre si que se apresenta a seus usuários como um sistema único e coerente"*.

Tolerância à falhas é definida como a característica que assume que softwares ou hardwares permaneçam funcionando "normalmente" após falhas em alguns de seus componentes/serviços. Em inúmeras situações sistemas distribuídos deparam-se com problemas de consenso [Tanenbaum 2002].

Consenso é o processo de chegar ao acordo sobre um resultado entre um grupo de

participantes, para isso, todos os elementos precisam ter as mesmas informações sobre as quais deve-se aplicar um mesmo algoritmo de decisão. Isso torna-se complexo quando o meio de comunicação e/ou os participantes podem confrontar-se com falhas. De modo geral, a finalidade destes problemas é fazer com que os processos alcancem determinada decisão em comum sobre algum valor [Guerraoui 2000]. Um algoritmo de consenso possui as seguintes propriedades: *(i)* Terminação - Todo processo em algum momento decide por um valor; *(ii)* Integridade Uniforme - Todo processo decide no máximo uma vez; e *(iii)* Acordo - Dois processos corretos não decidem por valores diferentes;

Sobre difusão atômica, pode-se dizer que é um mecanismo de comunicação de grande relevância na criação de sistemas distribuídos tolerantes à falhas, pois admite que diferentes processos recebam uma série ordenada de valores. Isto se torna particularmente favorável para concretizar a exclusão mútua distribuída ou para sustentar dados replicados com equivalência [Chandra 1996]. Suas propriedades formais determinam: *(i)* Validade - Se um processo envia por difusão uma mensagem *m*, então ele em algum momento entrega *m*; *(ii)* Acordo - Se um processo correto entrega uma mensagem *m*, então todo processo correto entrega *m* em algum momento; *(iii)* Integridade - Para cada mensagem *m*, todo processo correto entrega *m* no máximo uma vez, e somente se *m* foi enviada anteriormente por algum processo; e *(iv)* Ordem Total - Se dois processos corretos *p0* e *p1* entregam duas mensagens *m0* e *m1*, então *p0* entrega *m0* antes de *m1* se e somente se *p1* entrega *m0* antes de *m1*.

Multicast é o envio de informação para diversos destinatários ao mesmo tempo usando um comutador (por exemplo, um switch) para distribuição dos pacotes, retirando assim, a carga de trabalho de envio do emissor. Essa tática é muito eficiente, pois os pacotes passam por um link somente uma vez e apenas são duplicadas quando o link para os destinatários se divide em duas direções [Deering 1989]. Em contrapartida, numa comunicação Unicast o encaminhamento de um pacote é realizado para um único destino, ou seja, entrega ponto–a–ponto. Isso demanda muito trabalho para o emissor, devido ao fato de consumir link e processamento em cada envio. Em contrapartida, este tipo de comunicação pode trazer benefícios quanto a consistência e garantia de uma comunicação confiável, determinando e conhecendo o destinário que receberá o pacote [Deering 1989].

A topologia de rede em anel consiste em computadores interligados através de um circuito fechado, em série. Esse processo pode ser feito fisicamente ou logicamente. Uma topologia em anel pode ter o fluxo de pacotes de forma bidirecional ou direcional [Tanenbaum 2012].

3. RingPaxos

O protocolo Paxos [Prisco 2000] é uma solução usada para resolver problemas de consenso e implementar difusão atômica em uma rede de nós-processadores não-confiáveis. Paxos é normalmente utilizada em situações que requerem a durabilidade, como, por exemplo, para replicar um sistema de arquivos distribuídos ou um banco de dados. Como principal objetivo, Paxos propõe suprir problemas de falhas e perdas de mensagem, entregando valores a taxas de transmissão elevadas. O protocolo Paxos é hoje em dia usado em muitos sistemas tolerantes à falhas. Por exemplo, é a base para serviços como o *Google Analytics* e *Google Earth* [Marandi 2010].

Paxos apresenta diferentes processos e, um processo pode executar um ou mais

papéis. As funções de cada papel são:

- *Client* - Usam Paxos como meio de comunicação confiável. Um *Client* pode ser: *(i)* um *Proposer* – Processo que propõem valores ou *(ii)* um *Acceptor* – Processo que coopera para escolher um valor proposto.
- *Learner* - Aprendem o valor que foi escolhido.

Escrito em linguagem C e utilizando LibEvent [1] e Oracle Berkeley DB [2], RingPaxos[3] é uma implementação da família Paxos. Em testes realizados, RingPaxos chegou a entregar 900Mbit/s de dados para cada *Learner* [Marandi 2010]. Isso se dá inicialmente pela utilização de multicast, o qual pode fornecer altas taxas de transferência de mensagem quando comparada à uma comunicação ponto-a-ponto, pois o trabalho de transferência de mensagens para cada um dos destinos é feito pelo *switch* e também, pelo fato de que, para propagar uma mensagem para todos os destinatários existe apenas uma única chamada do sistema, economizando processamento do sistema. Por exemplo, com 20 *Acceptors* ligados numa rede Multicast, esta fornece quase 20 vezes o rendimento na entrega das mensagens comparado à uma comunicação ponto-a-ponto. No entanto, comunicação Multicast não é confiável, sendo assim, sujeita à perda de mensagens. Para atingir esses resultados, além do uso de multicast para redução de sobrecarga e congestionamento na rede, o protocolo tenta fazer progressos na sua execução, mesmo durante períodos em que um número limitado de réplicas (*Learners*) não responde. A Figura 1 ilustra os processos em uma execução de RingPaxos.

4. Estruturas, Arquiteturas e Configurações

Para realização dos testes foram utilizados 3 cenários de cluster, variando arquiteturas das máquinas e número de nós. Pontos importantes à serem destacados são: *(i)* o número de *cores* existentes (variando com 1 até 4), *(ii)* a memória (entre 1GB até 8GB) e a *(iii)* placa de rede (esta, sempre 1Gb, porém, de fabricantes diferentes). Na infraestrutura da rede, têm-se testes em estruturas de Cabo TP Cat6, conectados através de um switch Gigabit Cisco modelo SR2024. Os números de nós variaram entre 5 até 8, juntamente com o número de *Acceptors*, *Learners* e *Proposers* e o quórum teve sua variação entre 2 até 5 *Acceptors*. Em todas as máquinas o sistema operacional utilizado foi o Ubuntu 13.04, instalado somente com os pacotes necessários. Para coleta de informações das transmissões, os softwares Wireshark, Indicator-multiload e IpTraf foram utilizados.

A Tabela 1 mostra as configurações relevantes para as análises de cada cenário. Como mencionado, foram utilizados 3 cluster para testes. As Conf[1, 2 e 3] são referentes ao cluster 1, a Conf4 ao cluster 2 e a Conf5 o cluster 3. A escolha destas estruturas foi feita com o intuito de dimensionar a escalabilidade do protocolo, o comportamento da rede e a heterogeneidade do hardware.

Algumas características do sistema operacional, como tamanho do buffer do kernel, MTU, entre outras, podem influenciar na execução do protocolo, porém não foram alterados. Os parâmetros para execução de cada processo seja, *Learner*, *Acceptor* ou *Proposer*, são configuradas num arquivo que posteriormente é passado como parâmetro na

[1]Uma biblioteca assíncrona para notificação de eventos. Maiores informações em http://libevent.org.

[2]Biblioteca que proporciona o gerenciamento e alta performance de dados em banco de dados complexos. Maiores informações em http://www.oracle.com/technetwork/products/berkeleydb/overview/index-085366.html.

[3]Disponível no endereço http://libpaxos.sourceforge.net/paxos_projects.php#ringpaxos

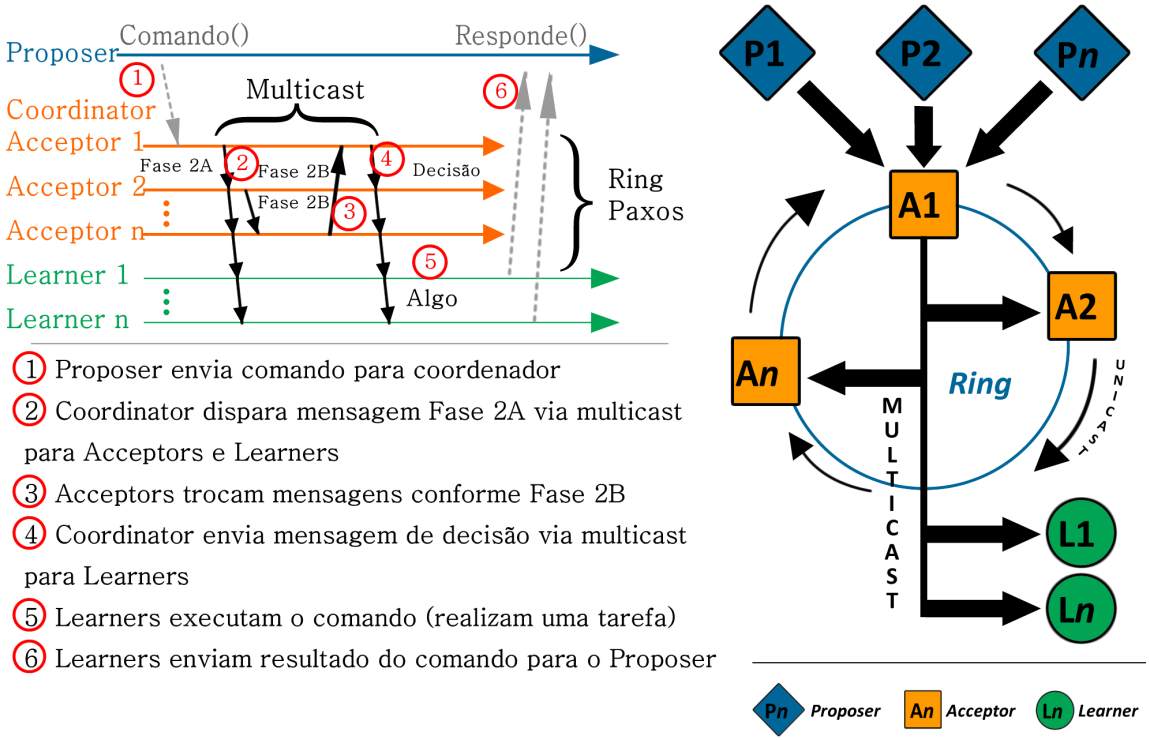

Figura 1. Processos na execução da estrutura da RingPaxos [Marandi 2010, Marandi 2011].

Tabela 1. Configuração dos cenários de testes.

Nome	Conf1	Conf2	Conf3	Conf4	Conf5
Equipamento	Pentium 4	Pentium 4	Pentium 4	Xeon	Xeon & Pentium 4
Processadores	1	1	1	1	1 & 1
Cores	1	1	1	4	4 & 4
Clock	1.7Ghz	1.7Ghz	1.7Ghz	2.4Ghz	2.4Ghz & 1.7Ghz
Memória	1GB	1GB	1GB	8GB	8GB & 1GB
Acceptors	4	3	2	3	3 + 2
Learners	2	4	6	1	1 + 7
Proposers	1	1	1	1	1 & 0
Quórum	4	3	2	2	2 + 2

inicialização de cada processo. A Tabela 2 mostra os parâmetros obrigatórios e detalhes de cada item.

Tabela 2. Parâmetros obrigatórios para execução do RingPaxos.

Item	Descrição	Exemplo
Multicast	IP e Porta do endereço Multicast.	Multicast 224.0.0.1 6667
Acceptor	Contém o nome, que por padrão deve ser Acceptor; seu id no anel; porta no anel e porta de conexão do Learner.	Acceptor 1 10.1.255.1 7771 5551
p1_interval	Intervalo de tempo antes do líder executar algum processo na fase 1. São dois valores, sendo o primeiro em segundos e o segundo em microssegundos.	1 0 p1_interval
p2_interval	Intervalo de tempo antes do líder executar algum processo na fase 2. São dois valores, sendo o primeiro em segundos e o segundo em microssegundos.	1 0 p1_interval

5. Experimentos e Resultados

Nesta seção são apresentadas situações para verificação e análise de possíveis falhas na inicialização e durante a execução do protocolo. Posteriormente, os experimentos e discussão dos resultados obtidos são descritos.

5.1. Metodologia

O *benchmark* para realização dos testes está incluso no *sourcecode* da biblioteca RingPaxos. O *benchmark* é simples, porém muito útil para a proposta estabelecida. Na sua execução, o mesmo submete diversos valores randômicos à uma taxa fixa para o algoritmo de consenso executado nos *Acceptors*. Os testes foram realizados 30 vezes para cada estrutura/situação estabelecida. Cada processo tinha o tempo de 30 segundos de execução, afim de verificar a estabilidade do processo durante um período de tempo. Para fins de casualidade as execuções eram alternadas entre os números de nós *Acceptors*, *Learners* e *Proposers*, conforme cada "Conf" da Tabela 1.

5.2. Experimentos

Inicialmente, percebesse que RingPaxos possui uma sequência lógica para inicialização. Primeiro, iniciam-se os *Acceptors* em ordem decrescente, ou seja, do último para o primeiro, pois estes tem uma comunicação Unicast dependente do *"Acceptor 1"* para *"Acceptor 2"* até, *"Acceptor n"* para *"Acceptor 1"* novamente. Essa comunicação é predefinida pela topologia em anel, a qual é utilizada no envio das mensagens de *"Acceptor"* para *"Acceptor"* seguinte, no processo de consenso sobre o dado proposto. Posteriormente, os *Learners* são inicializados sem restrição de ordem. Por fim, um *Proposer* deve iniciar a execução do protocolo.

Algumas situações foram exploradas durante a execução dos testes e são apresentadas na seguinte forma:

- Não existe uma algoritmo de eleição de líder na execução de RingPaxos. O *Acceptor 1* é considerado o líder e coordenador de cada execução.
- A adição de um *Learner* no futuro (Tempo em que o processo ficou parado) implica que este solicite todos os valores propostos. Nos testes realizados, a adição deste não aparentou nenhum problema. Talvez, se o futuro fosse num tempo muito longo, – sendo que nada foi estimado como tempo longo – algum problema poderia surgir devido ao limite de armazenamento dos valores decididos.
- A adição de um segundo *Proposer* simultâneo a execução do primeiro, resulta em queda do primeiro e parada do sistema. Porém, ao fim da execução de um *Proposer*, a execução de um *Proposer* novo pode ser realizada sem problema algum.
- Não há um mecanismo de detecção de falhas e consequentemente, reconfiguração e recuperação de processos. *Proposer* e *Acceptors* não podem falhar durante a execução e, *Learners*, após falhas, podem ser reestabelecidos manualmente.

Além disso, na Figura 2 são exibidos gráficos da taxa de transferência em Mbits/s em 30 execuções de processo RingPaxos em cada configuração estabelecida. Os resultados dos experimentos têm como objetivo verificar a alta taxa de transmissão de dados no processo RingPaxos.

Observando a Figura 2 nas conf1, conf2 e conf3 (2 (a), (b) e (c) em sequência) é apresentada uma oscilação evidente na taxa de transferência. A provável causa desta

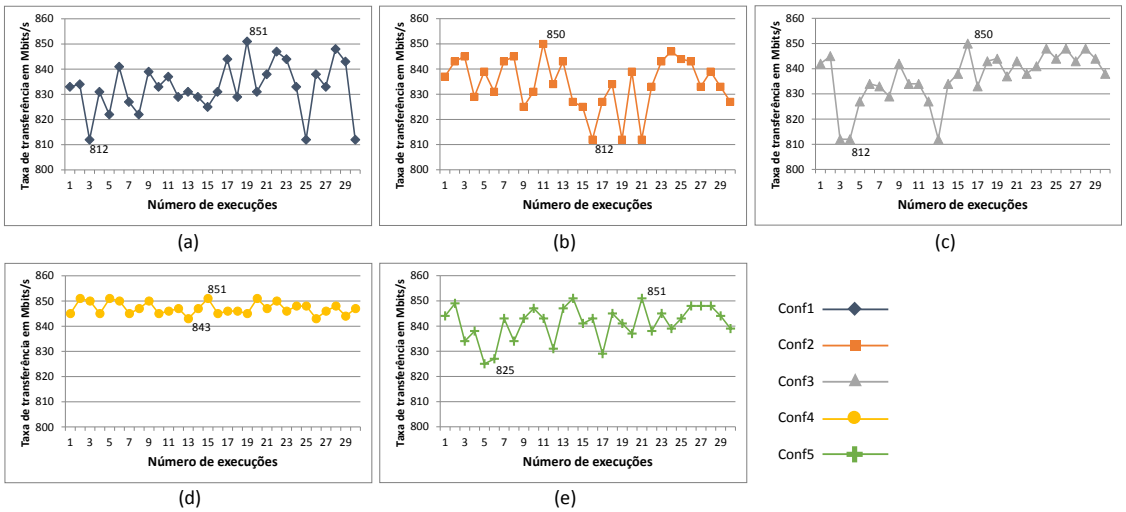

Figura 2. Execuções de RingPaxos em cada Configuração.

oscilação está relacionada ao hardware (placa de rede, memória, processamento e disco) de cada nó, o qual não foi projetado para fins de prestação de serviços desse porte. Mesmo com essa oscilação, o protocolo transmitiu dados a uma taxa com um valor mínimo de 812 Mbits/s nos casos de Conf1, Conf2 e Conf3; e como máxima 851 Mbits/s na Conf1, e uma máxima de 850 Mbits/s nas Conf2 e Conf3.

Na Figura 2 (d), diferente dos outros gráficos (Figura 2 (a), (b), (c) e (d)) tem-se uma pequena oscilação na taxa de transferência. Como o hardware desta configuração é projetado para prestação de serviços, atribui-se esta melhor estabilidade a esse fato. O valor mínimo de transferência nessa configuração (Conf4) atingiu 843 Mbits/s e o valor máximo 851 Mbits/s.

Visualizando a Figura 2 (e), a mescla de hardwares da Conf5 demonstra e reafirma a hipótese de que o hardware das configurações é implicante na variação da taxa de transferência. O gráfico apresenta a configuração num estado acima das Conf1, Conf2 e Conf3 (as que possuem um hardware de "menor qualidade") e abaixo da Conf4 (hardware de "melhor qualidade").

A partir da coleta das informações das execuções, um gráfico da média e um gráfico do desvio padrão foi elaborado para validação da proposta de alta taxa de transferência. A Figura 3 apresenta estes gráficos.

Concomitante a discussão da Figura 2, analisando a Figura 3, observa-se que o desvio padrão das Conf1, Conf2 e Conf3 são muito próximos, o que valida a justificativa da influência do hardware na variação da taxa de transferência. Além disso, o pequeno aumento na média da taxa de transferência apresentado em sequência de Conf1, Conf2 e Conf3, deve-se ao fato da utilização de multicast para envio de mensagens aos *Learners*, e da diminuição do quórum no anel, agilizando o processo de decisão e envio da resposta para os Learners. O valor baixo de desvio padrão para a Conf4, resulta numa boa média de transferência das execuções. Enquanto que, devido ao hardware da Conf5, esta tem um aumento no valor de desvio padrão em comparação a Conf4 e, consequentemente, uma queda na média das execuções.

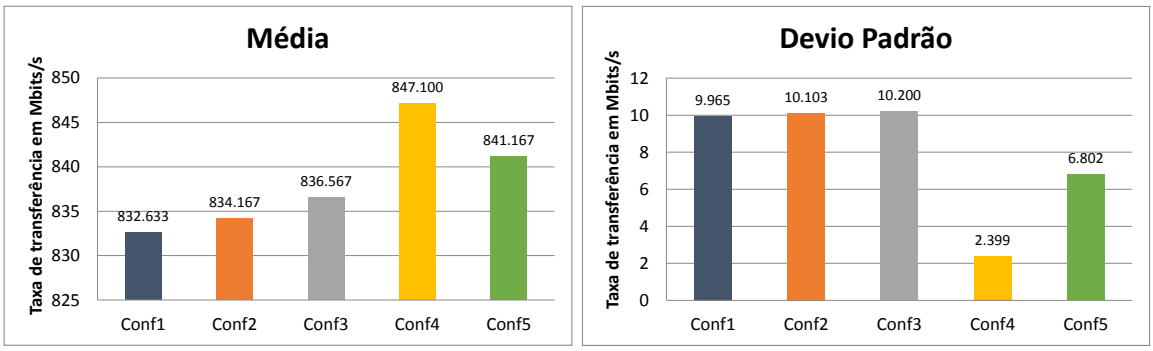

Figura 3. Média e desvio padrão das 30 execuções de RingPaxos em cada Conf.

6. Trabalhos Relacionados

O trabalho de [Marandi 2010] apresenta testes de comparação entre Ring Paxos e outros algoritmos para difusão atômica. Como resultados, são apresentados gráficos iniciais que demonstram a eficiência na utilização de *multicast versus unicast* para a propagação de mensagens entre receptores. Posteriormente, os processos de Ring Paxos são descritos, afim de apresentar detalhes e diferenciais do protocolo. Por fim, os resultados de comparação com outros algoritmos são apresentados demonstrando que Ring Paxos possui um throughput mais eficiente que diversos algoritmos, quando há aumento no número de receptores.

No trabalho de [Marandi 2011] são apresentadas técnicas para otimização de performance em replicação de estado de máquina utilizando execução especulativa para reduzir o tempo de resposta e particionamento de estado para aumentar a taxa de transferência de replicação de estado de máquina. Os resultados apresentados nos gráficos demonstram diversos testes com variação no throughput e tempo de resposta quando alterado o número de clientes e o número de réplicas no sistema.

Em contrapartida aos bons resultados e testes apresentados nos dois trabalhos, questões sobre falhas no sistema, análise de configurações de hardware e estruturas heterogêneas não são abordadas. Estas questões são abordadas e discutidas neste artigo.

7. Considerações Finais

Para implementar RingPaxos é necessário resolver uma série de questões práticas que são externas ao protocolo. Muitas opções de design não se tornam evidentes antes de perceber para que tipo de aplicação Paxos vai ser usado. Por exemplo, a escolha dos equipamentos, configurações e até mesmo modificações de parte do sistema podem influenciar nos resultados. Por esta razão deve-se considerar essas questões com antecedência e em tempo de design. Além disso, outro fator importante são os parâmetros do protocolo. Por exemplo, dois parâmetros importantes são os intervalos de tempo utilizados para os *timeouts* da fase 1 e fase 2. Um intervalo muito pequeno de tempo limite pode fazer com que os *Acceptors* não tenham a oportunidade de enviar a resposta e, um intervalo muito grande pode fazer o *Proposer* reenviar valores. Qualquer tipo de atraso nesses casos vai gerar mais tráfego na rede, devido as retransmissões das mensagens. Não há nenhum número "mágico" para identificar esses valores, uma vez que eles dependem quase inteiramente da latência do disco dos *Acceptors* e da latência da rede.

A implementação de RingPaxos disponível não está completa. Não há um mecanismo de detecção de falhas e consequentemente, reconfiguração e recuperação de processos. Por exemplo, se um *Acceptor* falhar, a transmissão será interrompida. Um mecanismo de reconfiguração dinâmica de terceiros, ou seja, implementado fora da estrutura Paxos, pode ser utilizado para "liberar" uma réplica que falhou permanentemente, ou para adicionar réplicas para substituição das que falharam e/ou adicionar novas réplicas para o grupo conforme demanda. A adição de novas réplicas só pode ser aplicada para novos *Learners* ou *Proposers*, pois a adição de *Acceptors* inicialmente é determinada e possui uma quantidade mínima para execução do protocolo e estática nos arquivos de configuração.

Referente a proposta de altas taxas de transmissão para a entrega, o protocolo apresentou oscilações quando executado numa estrutura heterogênea e/ou com nodos de hardware de baixa qualidade. Porém, bons resultados foram obtidos quando executado o protocolo sob hardwares indicados para servidores de rede, chegando próximo aos 900 Mbits/s de taxa de transmissão proposta.

Perante a utilização de multicast do protocolo para difusão de mensagens, como proposta de trabalhos futuros, pretende-se realizar experimentos aumentando o número de *switches* da rede, afim de verificar as alterações de *delay* e da taxa de transferência na entrega de informações. Além disso, pretende-se realizar testes com sistemas de dados distribuídos, com o intuito verificar a consistência dos mesmos, quando expostos a situações de falhas na execução do protocolo.

Referências

Chandra, Tushar Deepak e Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, 43(2):225–267.

Coulouris, George e Dollimore, J. e. K. T. (2007). *Sistemas distribuídos: conceitos e projeto*. Editora Bookman.

Cristian, F. (1991). Understanding fault-tolerant distributed systems. *Commun. ACM*, 34(2):56–78.

Deering, S. (1989). Rfc-1112: Host extension for ip multicasting. Technical report.

Guerraoui, Rachid e Hurfinn, M. e. M. A. e. O. R. e. R. M. e. S. A. (2000). Consensus in asynchronous distributed systems: A concise guided tour. In *Advances in Distributed Systems*, pages 33–47. Springer.

Marandi, P.J. e Primi, M. e. P. F. (2011). High performance state-machine replication. In *Dependable Systems Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, pages 454–465.

Marandi, Parisa Jalili e Primi, M. e. S. N. e. P. F. (2010). Ring paxos: A high-throughput atomic broadcast protocol. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, pages 527–536. IEEE.

Melliar-Smith, P.M. e Moser, L. e. A. V. (1990). Broadcast protocols for distributed systems. *Parallel and Distributed Systems, IEEE Transactions on*, 1(1):17–25.

Moser, L. E. e Melliar-Smith, P. M. e. A. D. A. e. B. R. K. e. L.-P. C. A. (1996). Totem: a fault-tolerant multicast group communication system. *Commun. ACM*, 39(4):54–63.

Neiger, Gil e Toueg, S. (1988). Automatically increasing the fault-tolerance of distributed systems. In *Proceedings of the seventh annual ACM Symposium on Principles of distributed computing*, PODC '88, pages 248–262, New York, NY, USA. ACM.

Prisco, Roberto De e Lampson, B. e. L. N. (2000). Revisiting the paxos algorithm. *Theoretical Computer Science*, 243(1):35–91.

Tanenbaum, Andrew S e Wetherall, D. J. (2012). *Computer networks*. Pearson Higher Ed.

Tanenbaum, Andrew S e Van Steen, M. (2002). *Distributed systems*, volume 2. Prentice Hall.