Custo da Comutação Dinâmica de Protocolos de Comunicação

Cristina Fonseca, Liliana Rosa, and Luís Rodrigues

IST/INESC-ID,{cfonseca,lrosa}@gsd.inesc-id.pt,ler@ist.utl.pt

Resumo Este artigo foca-se na adaptação dinâmica dos protocolos de comunicação que suportam as aplicações distribuídas. Em particular, o artigo apresenta duas contribuições: i) dois protocolos de comutação específicos e ii) avaliação quantitativa dos ganhos que se podem obter ao usar comutadores específicos em detrimento de um comutador genérico, usando como caso de estudo o seguinte conjunto de serviços: serviço de ordenação FIFO, serviço de ordenação total de mensagens em difusão e serviço de ordenação causal de mensagens. A análise destes três casos particulares oferece uma melhor compreensão das relações custo-benefício entre estes três tipos de comutadores.

Abstract This paper addresses the dynamic adaptation of communication protocols. Namely, it provides the following contributions: i) it proposes two novel protocol switching algorithms and, ii) it provides a quantitative assessment of the benefits of using failure switching algorithms against the use of generic switching algorithms. The following communication protocols are used as case studies: totally ordered multicast, causally ordered multicast and FIFO reliable multicast. The experimental results offer a better insight on the tradeoffs involved in the execution of protocol switching algorithms.

1 Introdução

Durante a execução de aplicações distribuídas ocorrem frequentemente mudanças de contexto, tais como alterações à capacidade de processamento disponível, na latência da rede, etc. Uma das formas de optimizar o desempenho destes sistemas consiste em adaptar dinamicamente o seu comportamento em resposta às alterações de contexto. Este artigo foca-se na adaptação dinâmica dos protocolos de comunicação que suportam a aplicação. Esta adaptação pode ser conseguida comutando, em tempo de execução, um protocolo por outro que ofereça o mesmo serviço.

A comutação dinâmica é orquestrada por um protocolo designado por *comutador*. Dois tipos de comutadores tem sido propostos na literatura: i) comutadores *genéricos* [1], que podem ser aplicados a um leque vasto de protocolos de comunicação que oferecem diferentes serviços, desde que estes protocolos possuam um conjunto mínimo de propriedades em comum; ii) comutadores *especializados* [2, 3, 4] que apenas podem ser aplicados para trocar concretizações alternativas de um único serviço em particular.

Os comutadores genéricos permitem reduzir o número de comutadores que é necessário suportar no sistema e os comutadores específicos podem optimizar o seu desempenho tirando partido de propriedades específicas dos serviços para os quais se destinam. Este artigo faz uma avaliação quantitativa dos ganhos que se podem obter ao usar comutadores específicos em detrimento de um comutador genérico para três serviços concretos: serviço de ordenação total de mensagens em difusão, serviço de ordenação causal e serviço de ordenação FIFO.

O resto do artigo está organizado do seguinte modo. A Secção 2 descreve o trabalho relacionado. Na Secção 3 descrevem-se os protocolos que são alvo do nosso estudo. Na Secção 4 faz-se uma análise comparativa do desempenho do comutador genérico e dos comutadores específicos. Finalmente, a Secção 5 conclui o artigo.

2 Trabalho Relacionado

A comutação dinâmica entre protocolos de comunicação é uma área onde já existe bastante trabalho realizado, tanto a nível da especificação dos protocolos de comutação como no desenvolvimento de arquitecturas de software que facilitam a composição de protocolos e a reconfiguração destas composições em tempo de execução.

No que se refere ao nosso alvo de estudo, os protocolos de comutação, a aproximação mais comum para comutar entre A e B consiste em utilizar a estratégia que designaremos por *comutação por paragem* [5]. Esta estratégia deixa o serviço indisponível durante um período de tempo que pode ser significativo. Desta forma, têm sido propostas estratégias alternativas que tentam minimizar o período de indisponibilidade do serviço durante a comutação. É neste contexto que surgem as soluções genéricas e especializadas que referimos anteriormente.

Os comutadores genéricos [1] são concebidos para satisfazer um conjunto de propriedades e a capacidade que um protocolo tem de ser aplicável a vários casos advém das hipóteses que este faz acerca dos protocolos alvo assim como das propriedades que o comutador preserva durante a comutação.

Os comutadores especializados [2, 6, 3] usam as propriedades específicas dos protocolos de comunicação para os quais foram desenhados para minimizar a interferência no fluxo de pedidos durante a comutação.

3 Casos de Estudo

Para analisar tanto os comutadores genéricos bem como os comutadores especializados, vamos comparar o desempenho de um comutador genérico (que designaremos por G) com três comutadores especializados, um concebido para comutar entre protocolos de ordem total (que designaremos por comutador T), outro concebido para comutar entre protocolos de ordem causal (que designaremos por comutador C) e um último que permite comutar entre protocolos de difusão melhor-esforço com ordem FIFO (será designado por F). Os testes usarão

composições de protocolos semelhantes à ilustrada na Figura 1. A Figura mostra ainda a interacção do coordenador da adaptação (cujo papel iremos descrever mais à frente) com os nós do grupo de comunicação.

No topo da pilha encontra-se a aplicação, que é responsável por invocar os serviços de comunicação fornecidos pela composição de protocolos subjacente. Mais abaixo, encontra-se a camada *comutador*, seguindo-se A e B que são diferentes protocolos que oferecem o mesmo serviço. O objectivo da arquitectura é permitir comutar da utilização de A para a utilização de B.

Desta composição, as camadas essenciais para a comutação são o comutador e o multiplexer. O comutador, que em cada teste poderá ser instanciado por um comutador G, T, C ou F, é responsável por armazenar estado, nalguns casos mensagens e ainda por enviar mensagens de controlo específicas do protocolo de comutação. É esta camada que efectua todo o processamento relativo à comutação entre os protocolos A e B da camada abaixo. O multiplexer é um componente cujo papel é esconder a existência de duas concretizações diferentes do serviço acima (A e B) das camadas inferiores da composição.

Em termos de canais, como ilustra a Figura 1, existe um canal u que liga a aplicação ao comutador. Este é o canal de comunicação que é utilizado pela a aplicação para trocar mensagens em difusão. O comutador possui dois canais, a e b, para comunicar com a camada multiplexer. Um dos canais utiliza o protocolo A e o outro o protocolo B. Em regime estável, todas as mensagens recebidas pelo canal u são encaminhadas apenas para a (ou b). Durante o processo de comutação, aplicam-se os algoritmos referidos anteriormente. O multiplexer despacha as mensagens recebidas por a e b para um único canal g que concretiza a comunicação em grupo fiável. Mais concretamente, o canal g usa uma composição de protocolos, que faz parte da distribuição do Appia, e que assegura propriedades de *sincronia na vista*[7].

3.1 Coordenação da Comutação

Assumimos que a comutação entre o protocolo A e B (e vice-versa) é activada por uma entidade logicamente centralizada designada por *coordenador*. O coordenador é responsável por comunicar com o comutador existente em cada nó para dar início ao processo de comutação. Neste trabalho, as estruturas de suporte à comutação estão sempre presentes na composição de protocolos pois o protocolo genérico que usámos obriga a que cada comutador informe os restantes do número de mensagens enviadas e consequentemente que as conte.

3.2 O Comutador G

Como comutador genérico escolhemos o comutador descrito em [1]. O protocolo concretizado resume-se no seguinte: quando o coordenador informa o comutador que é necessário comutar entre A e B, G envia uma mensagem a todos os participantes informando-os do número de mensagens que até agora tinham sido enviadas através de A. Esta é a última mensagem enviada através de A e a

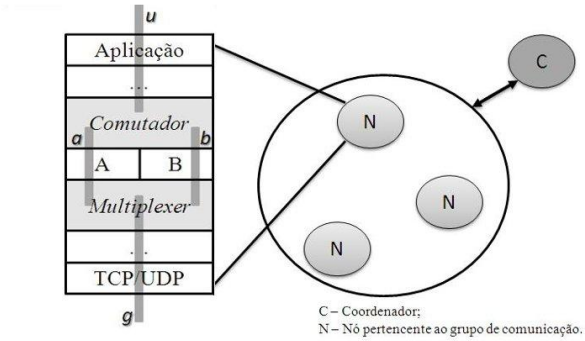


Figura 1. Composição com comutador.

partir daqui todas as mensagens enviadas passam a ser processadas através de B. Ao receber esta mensagem de todos os outros nós, cada comutador sabe quantas mensagens deveria ter recebido de cada um dos outros nós, através do protocolo A. G espera até ter recebido todas as mensagens de todos os nós para poder desactivar A. Finalmente, o comutador informa o coordenador que a comutação foi concluída.

Neste comutador, a necessidade de esperar por, e entregar, todas as mensagens enviadas pelo protocolo A antes de entregar mensagens enviadas pelo protocolo B é fundamental para que não sejam violadas as propriedades do protocolo subjacente, pelo que este comutador é susceptível a suspender as entregas se uma das mensagens enviadas em A sofrer atrasos.

3.3 O Comutador T

Informalmente, um protocolo de ordem total para comunicação em grupo garante que quaisquer duas mensagens entregues a quaisquer dois processos, são entregues na mesma ordem a cada um destes processos. Como já referimos, o comutador que designamos por T permite comutar entre dois protocolos de ordem total. Neste trabalho usamos o comutador T proposto por Mocito et al.[2] que, de um modo geral, funciona da seguinte forma. Durante a fase de comutação os eventos são processados e enviados simultaneamente pelos dois protocolos, sendo que a primeira mensagem enviada também por B é marcada. Na recepção, todas as mensagens enviadas através de A são entregues à aplicação e as que chegam por B são guardadas temporariamente. Depois de um nó ter recebido a primeira mensagem marcada de todos os outros nós, pode entregar à aplicação as mensagens guardadas que ainda não tinham sido entregues e descartar todas as outras. A partir deste momento o protocolo A deixa de processar mensagens.

Note-se que o comutador T pressupõe que o protocolo subjacente entrega as mensagens de acordo com a ordem total, pelo que poderá não funcionar noutros cenários. Dos comutadores aqui apresentados, apenas este obriga a enviar mensagens simultaneamente em dois protocolos. Isto inclui uma sobrecarga adicional que é discutida em pormenor em [2].

3.4 O Comutador C

Um protocolo de ordenação causal, assegura que as mensagens são entregues de acordo com as relações potenciais de causa-efeito, tal como descrito em [8]. Neste protocolo, duas mensagens que estejam potencialmente relacionadas de forma causal são entregues pela mesma ordem a todos os participantes; pelo contrário, mensagens concorrentes podem ser entregues por ordem distinta a participantes distintos. A função de um comutador para protocolos de ordem causal é assegurar que se preservam as relações de ordem entre as mensagens enviadas pelo protocolo A e as mensagens enviadas pelo protocolo B.

Não tendo encontrado na literatura nenhum comutador especializado para ordem causal, desenvolvemos o comutador C como uma adaptação do protocolo genérico G [1]. Tal como o comutador G, o comutador C também requer a contagem explicita de mensagens. No entanto, a entrega das mensagens enviadas por um nó não fica dependente do momento em que os restantes nós aplicam a comutação. O comutador C proposto funciona da seguinte forma. O comutador concretiza um protocolo de ordenação causal, baseado em relógios vectoriais[9], que apenas ficam activos durante a comutação. Em regime estável, antes da comutação, o comutador limita-se a atribuir um número de sequência local a cada mensagem que envia, e a registar o número de sequência das mensagens que entregou de cada um dos restantes processos.

Um comutador, ao iniciar a comutação, deixa de enviar mensagens pelo protocolo A e passa a enviar as mensagens pelo protocolo B. No entanto, acrescenta também um relógio vectorial às mensagens enviadas por B. Este relógio deixará de ser enviado quando a comutação terminar, mas permite ordenar mensagens enviadas por A em relação a mensagens enviadas por B durante a comutação. Neste algoritmo, um comutador inicia a comutação quando o primeiro dos seguintes eventos ocorrer: i) o comutador recebe uma instrução para comutar, vinda do coordenador ou; ii) o comutador recebe uma mensagem pelo protocolo B (o que significa que é necessário ordenar mensagens recebidas através de A e de B). Durante o processo de comutação, as mensagens recebidas pelo protocolo A são entregues sem restrições e as mensagens recebidas pelo protocolo B são entregues de acordo com o seu relógio vectorial. A comutação termina assim que todos os participantes estiverem a usar o protocolo B. Nesse momento, a utilização do relógio vectorial pelo comutador deixa de ser necessária, bastando garantir as seguintes restrições: i) a ordem FIFO entre as mensagens enviadas com e sem relógio vectorial é respeitada e; ii) as mensagens são entregues pela ordem que são recebidas do protocolo B.

3.5 O Comutador F

Um protocolo de ordenação FIFO assegura que as mensagens são entregues aos destinatários na ordem pela qual foram enviadas pelo remetente. Note-se que, ao contrário da ordem total, a ordem FIFO só reordena a entrega de mensagens que tenham sido enviadas pelo mesmo remetente. A função de um comutador

para protocolos de ordem FIFO é assegurar que se preserva a ordem FIFO entre a última mensagem enviada por A e a primeira enviada por B.

Não tendo encontrado na literatura nenhum comutador especializado para ordem FIFO, desenvolvemos o comutador F como uma adaptação do protocolo genérico G [1]. A nossa adaptação evita a contagem explicita de mensagens pelo comutador, assim como a sincronização entre os diversos canais FIFO. O comutador F proposto funciona da seguinte forma: após o comutador receber a informação de que deverá iniciar o processo de comutação, deixa de enviar as mensagens através do protocolo A e passa a enviá-las por B. Para assinalar a transição, a última mensagem enviada pelo protocolo A é marcada. No lado do receptor, mensagens recebidas por B vindas de um dado emissor são armazenadas até que a última mensagem enviada por este emissor através do protocolo A seja entregue. Nesse momento, são entregues todas as mensagens recebidas por B, pela ordem em que foram recebidas.

4 Avaliação

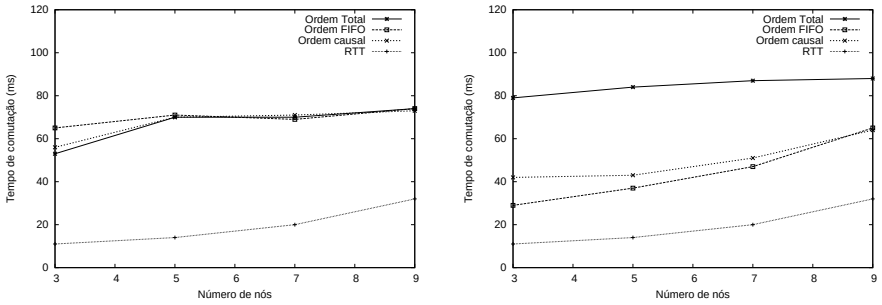
Nesta secção é avaliado o desempenho dos protocolos de comutação genéricos (comutador G) face aos protocolos específicos (comutadores T, C e F).

4.1 O Ambiente de Execução

A comutação dinâmica de protocolos é bastante facilitada se existir uma arquitectura de software de apoio à composição e execução de protocolos de forma eficiente e flexível. Encontram-se na literatura diversos sistemas de suporte à composição de protocolos que satisfazem estes requisitos, tais como o Cactus [10] ou o Appia [11]. Para suportar o nosso trabalho escolhemos o sistema Appia. Os testes foram efectuados em ambiente real numa grid constituída por 17 Intel Pentium IV (1-core) @ 3.2GHz / 1G RAM e 14 Intel Q6600 (4-core) @ 2.4GHz / 8G RAM organizados numa subrede gigabit privada. Desta rede faz também parte 1 AMDOpteron (dual) @ 2.4GHz / 16G RAM numa subrede gigabit pública. A arquitectura é a apresentada anteriormente, o código desenvolvido em Java e executado no nível de utilizador.

4.2 Métricas

A avaliação incide sobre duas métricas de desempenho: o tempo total necessário para realizar a comutação e o impacto que a comutação tem sobre a taxa de entrega das mensagens à aplicação. O tempo para realizar a comutação é o tempo medido entre o coordenador enviar a primeira mensagem com a instrução de iniciar a reconfiguração para um comutador e receber a última confirmação de que a reconfiguração foi concluída. Para perceber o peso computacional do algoritmo mediu-se também o tempo de comunicação entre o coordenador e os comutadores num sistema em que o comutador responde de imediato (RTT), que se indica nos gráficos como forma de comparação. Para avaliar estas métricas, testámos vários cenários cujos resultados se descrevem de seguida.



(a) Comutador genérico (b) Comutador específico

Figura 2. Efeito de variação do número de nós

4.3 Variação do Número de Nós

Esta medida pretende avaliar o comportamento dos diferentes comutadores, em função do número de nós envolvidos na comunicação. Esta avaliação foi feita para 3, 5 ,7 e 9 nós, mantendo-se a taxa de envio de mensagens constante e igual a 100msgs/s.

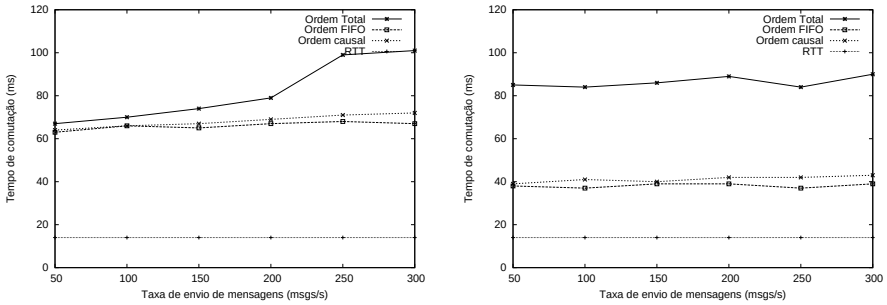
Como se observa na Figura 2, em ambos os casos (comutador genérico e comutador especializado), o tempo total necessário para comutar entre os protocolos varia de forma aproximadamente constante, aumentando ligeiramente com o aumento do número de nós. Sendo este aumento bastante idêntico nos dois casos de comutadores, podemos dizer que o protocolo não tem qualquer influência sobre esta variável. A variação do número de nós interfere assim ligeiramente no tempo de troca pois é necessário tempo adicional para difundir a mensagem com a ordem para comutar (mais nós aumentam o número de mensagens na rede e o processamento efectuado, o que também se traduz no aumento dos valores de RTT).

Podemos ainda concluir através da análise da Figura 2(a), que no caso do comutador genérico todos os protocolos têm tempos de comutação semelhantes. No caso dos comutadores específicos (Figura 2(b)), o tempo necessário para comutar nas diversas situações está relacionado com os requisitos que a ordem que se quer preservar impõe. Em termos de garantias, o protocolo mais exigente é o de ordem total, seguindo-se o de ordem causal e por fim o de ordem FIFO, ou seja, a ordem de complexidade é também a ordem a nível de tempo necessário.

4.4 Variação da Taxa de Envio de Mensagens

Com o objectivo de avaliar o comportamento do comutador entre protocolos para diferentes taxas de envio de mensagens, fez-se variar esta taxa em 50, 100, 150, 200, 250 e 300 msgs/s, mantendo-se o número de nós constante e igual a 5. Mais uma vez a métrica usada foi o tempo total de troca.

Analisando a Figura 3(a) concluímos que, para os comutadores genéricos, o tempo de comutação se mantém aproximadamente constante com excepção do



(a) Comutador genérico (b) Comutador específico

Figura 3. Efeito de variação da taxa

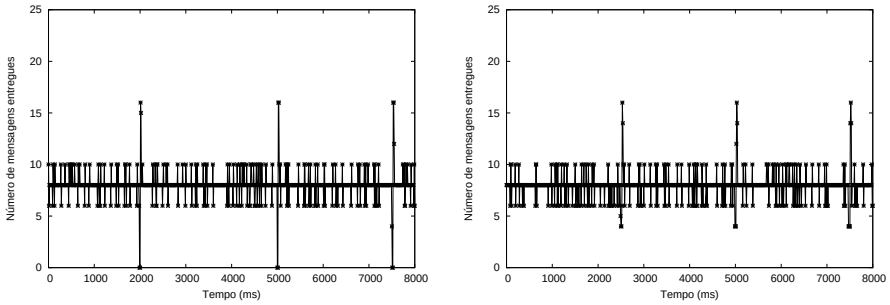
comutador genérico de ordem total em que há um aumento do tempo necessário para comutar, com o aumento do número de mensagens para processar. Este aumento é visível a partir de um certo valor de taxa (entre 200 e 250 msgs/s), após o qual se mantém constante. No que diz respeito aos comutadores específicos (Figura 3(b)), as variações observadas no tempo total de comutação em função do aumento da taxa de envio de mensagens são mínimas (mesmo para taxas superiores a 200 msgs/s). Nos casos dos comutadores de ordem causal e FIFO, os tempos de comutação são menores.

Se compararmos os dois comutadores de ordem total, concluímos que a variação que se observa no caso genérico se deve ao armazenamento temporário de mensagens. Considerando o seu funcionamento, para o caso do comutador específico (em que são sempre armazenadas mensagens durante o período de comutação), esta variação da taxa não se verifica e o tempo de comutação apresenta pequenas variações com a variação da taxa de envio de mensagens. No caso do comutador genérico este protocolo armazena mensagens a partir de uma certa taxa, sendo necessário tempo adicional para se efectuar a comutação. O tempo adicional é o necessário para reservar, manipular e libertar os recursos associados a esse armazenamento.

4.5 Taxa de Entrega das Mensagens à Aplicação

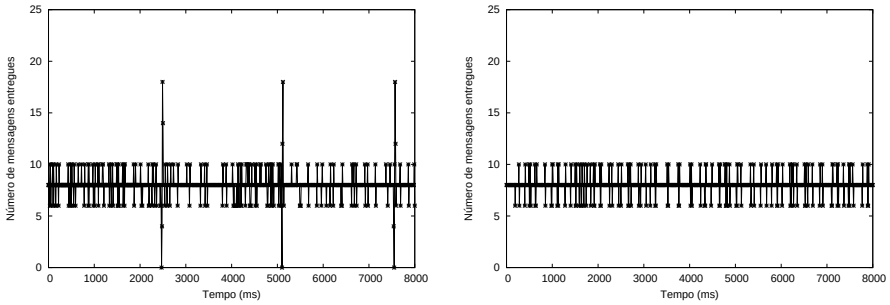
Outra forma de avaliar e comparar os diferentes protocolos de comutação, no que diz respeito ao impacto que estas operações têm para a aplicação, é estudar o número de mensagens que lhe são entregues e analisar a variação durante o período de comutação. Neste caso usou-se como métrica o número de mensagens entregues à aplicação, a cada 10 ms, durante 3 execuções do protocolo consecutivas,para um nó pertencente a um grupo de comunicação constituído por 5 elementos.

Por análise da Figura 4, concluiu-se que, para os protocolos de ordem total, enquanto o genérico inibe a entrega de mensagens à aplicação, o específico apenas diminui a taxa de entrega durante o período de comutação. Esta inibição ou não



(a) Genérico (b) Específico

Figura 4. Entrega das mensagens à aplicação - ordem total.



(a) Genérico (b) Específico

Figura 5. Entrega das mensagens à aplicação - ordem causal.

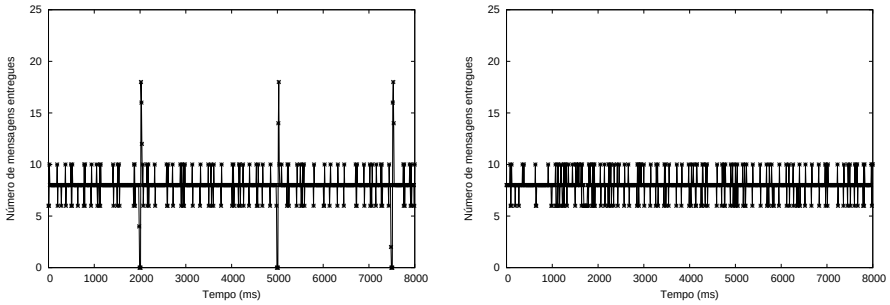
de mensagens não está relacionada com a especificidade do protocolo mas sim com a abordagem escolhida para o concretizar.

Analisando as Figuras 5 e 6 relativas aos protocolos de ordem causal e FIFO, para o caso genérico há interrupção na entrega de mensagens (de forma semelhante ao que acontece no comutador genérico para ordem total), também devido ao armazenamento temporário de mensagens, enquanto no caso específico essa interrupção não ocorre.

Conclui-se que, dependendo dos casos, é possível realizar alterações na composição de protocolos que suportam a comunicação, em tempo de execução, minimizando a interferência para as camadas superiores.

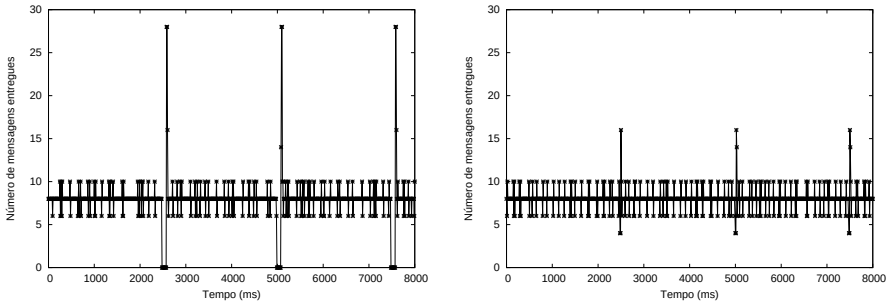
4.6 Comportamento do Sistema Perante um Nó Lento

Até aqui tirámos apenas conclusões usando cenários em que cada nó processa de imediato a ordem de comutação que recebe do coordenador. No entanto, um nó pode sofrer atrasos no processamento das mensagens que recebe. Analisámos a taxa de entrega de mensagens à aplicação, no caso em que um dos nós demora



(a) Genérico (b) Específico

Figura 6. Entrega das mensagens à aplicação - ordem FIFO.



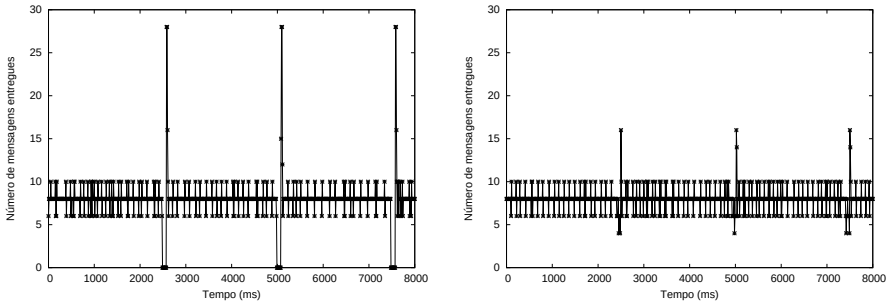
(a) Genérico (b) Específico

Figura 7. Entrega das mensagens à aplicação (nó lento) - ordem total.

1000 ms a processar a ordem para comutar, vinda do *coordenador*. Os resultados são apresentados nas Figuras 7, 8 e 9.

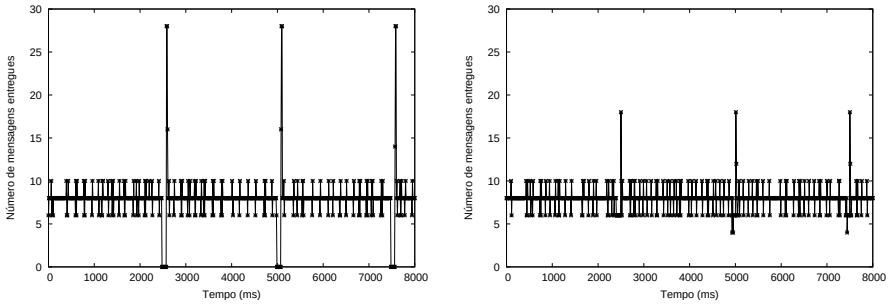
Se compararmos estes resultados com os resultados apresentados anteriormente, observamos que, quando se usa o comutador genérico, a existência de um nó lento faz com que exista um período significativo em que não é entregue nenhuma mensagem à aplicação (enquanto o nó lento não responde). Isto acontece porque o protocolo em causa obriga a que a troca só se efectue quando todos os nós respondem ao grupo dizendo qual o número de mensagens já enviadas utilizando o protocolo A, ou seja, o processo de comutação de um nó está dependente dos outros nós envolvidos na comunicação. Pelo contrário, com os comutadores específicos a interferência para a aplicação é minimizada, uma vez que as únicas mensagens que deixam de ser entregues à aplicação são as mensagens enviadas pelo próprio nó lento, que se acumulam no emissor e depois são enviadas de rajada.

Concluímos então que quando um dos nós é mais lento relativamente aos outros, os protocolos específicos permitem minimizar o impacto que a existência deste nó causa na aplicação distribuída.



(a) Genérico (b) Específico

Figura 8. Entrega das mensagens à aplicação (nó lento) - ordem causal.



(a) Genérico (b) Específico

Figura 9. Entrega das mensagens à aplicação (nó lento) - ordem FIFO.

5 Conclusões

Este artigo abordou o problema de alterar, em tempo de execução, os protocolos de comunicação que suportam a aplicação. Foram propostos dois novos protocolos de comutação específicos, um para suportar a comutação de protocolos de ordem causal e outro para suportar a comutação de ordem FIFO, e foi feita uma análise comparativa do desempenho de protocolos de comutação genéricos e de protocolos de comutação específicos. Os resultados experimentais confirmam e reforçam as expectativas intuitivas: o protocolo genérico interfere significativamente com o fluxo de mensagens. Por outro lado, os protocolos específicos para ordem causal e FIFO, aqui propostos, conseguem suportar a comutação sem afectar o fluxo de mensagens. O protocolo específico para ordem total não evita totalmente a interferência, devido à necessidade de enviar mensagens por dois canais, mas é substancialmente menos intrusivo que o protocolo genérico. As principais razões que suportam este facto são a sincronização entre os nós (que de modo muito particular tem consequência negativas quando um dos nós está atrasado no processamento de mensagens relativamente aos outros), e a necessidade de se armazenarem mensagens mesmo quando as características do

protocolo não obrigavam a tal (como é o caso da ordem FIFO). Há ainda o facto de o protocolo genérico ter de verificar um conjunto de propriedades para poder ser aplicável nos diferentes casos, o que gera processamento e manutenção de estado adicional. Desta forma, pode concluir-se que a utilização de protocolos específicos traz vantagens de desempenho significativas que, do ponto de vista dos autores, justifica o acréscimo de complexidade introduzido pela necessidade de suportar a existência de múltiplos comutadores no sistema.

Agradecimentos Este trabalho foi parcialmente apoiado pela FCT através do projecto "Redico" (PTDC/ EIA/ 71752/ 2006). Os autores agradecem ainda ao José Mocito pelos comentários feitos a versões preliminares deste artigo.

Referências

1. Liu, X., Renesse, R.V., Bickford, M., Kreitz, C., Constable, R.: Protocol switching: Exploiting meta-properties. In: In Proc. 21st IEEE Intl. Conf. on Distributed Computing Systems Workshops (ICDCSW´01), Intl. Workshop on Applied Reliable Group Communication (WARGC). (2001) 37–42
2. Mocito, J., Rodrigues, L.: Run-time switching between total order algorithms. In: In: Proceedings of the Euro-Par 2006. LNCS, Springer-Verlag (2006) 582–591
3. Chen, W.: Constructing adaptive software in distributed systems. In: Proceedings of the 21st International Conference on Distributed Computing Systems, IEEE Computer Society (2001) 635–643
4. Liu, X., van Renesse, R.: Fast protocol transition in a distributed environment (brief announcement). In: PODC '00: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing, New York, NY, USA, ACM (2000) 341
5. van Renesse, R., Birman, K., Hayden, M., Vaysburd, A., Karr, D.: Building adaptive systems using ensemble. Softw. Pract. Exper. **28**(9) (1998) 963–979
6. Rütti, O., Wojciechowski, P.T., Schiper, A.: Structural and algorithmic issues of dynamic protocol update. In: Proceedings of IPDPS '06: the 20th IEEE International Parallel and Distributed Processing Symposium (Rhodes Island, Greece), IEEE Computer Society (2006)
7. Birman, K., van Renesse, R., eds.: Reliable Distributed Computing With the ISIS Toolkit. Number ISBN 0-8186-5342-6. IEEE CS Press (March 1994)
8. Lamport, L.: Time, clocks and the ordering of events in a distributed system. Communications of the ACM **21**(7) (July 1978) 558–565
9. Raynal, M., Schiper, A., Toueg, S.: The causal ordering abstraction and a simple way to implement it. Information processing letters **39**(6) (September 1991) 343–350
10. Goodale, T., Allen, G., Lanfermann, G., Massó, J., Seidel, E., Shalf, J.: The cactus framework and toolkit: Design and applications. In: 5th International Conference In Vector and Parallel Processing (VECPAR), Springer (2003) 182–192
11. Miranda, H., Pinto, A., Rodrigues, L.: Appia: A flexible protocol kernel supporting multiple coordinated channels. In: in Proc. 21st International conference on Distributed Computing Systems (ICDCS-21. (2001) 707–710