Checkpointing e Recuperação de Falhas em Sistemas Distribuídos Particionáveis

Tiemi C. Sakata Islene C. Garcia Luiz E. Buzato

Universidade Estadual de Campinas
Caixa Postal 6176
13083-970 Campinas, SP, Brasil
Tel: +55 19 3788 5876 Fax: +55 19 3788 5847
{tiemi, islene, buzato}@ic.unicamp.br

Resumo

Os protocolos existentes para *checkpointing* e recuperação por retrocesso normalmente consideram que não haverá partições na rede de comunicação e que a recuperação por retrocesso só será iniciada quando todos os processos que falharam voltarem à operação normal. Estas restrições podem tornar inviável a utilização de *checkpointing* e recuperação por retrocesso em aplicações que operam sobre ambientes móveis. Neste resumo comentaremos como essas restrições podem ser contornadas para permitir o avanço seguro da aplicação mesmo na presença de falhas e/ou partições da rede.

1 Introdução

Um sistema distribuído assíncrono é formado por um conjunto de componentes que se comunicam exclusivamente através da troca de mensagens. O meio pelo qual estas mensagens serão transmitidas pode variar de um sistema para outro. Em redes fixas, a ligação entre os componentes é feita através de cabos de comunicação. Em redes ad hoc os componentes móveis se comunicam diretamente através de conexão sem fio.

Um sistema distribuído está dividido em partições se pelo menos um par de componentes não consegue se comunicar. Cada partição é formada pelo subconjunto de componentes que ainda conseguem estabelecer comunicação. Nas redes fixas, partições ocorrem apenas quando há falhas de componentes ou de canais de comunicação. Nas redes ad hoc, as partições podem ocorrer por falhas nos componentes ou por incapacidade de comunicação devido ao distanciamento. Desta forma, podemos considerar que partições são relativamente raras e transientes em redes fixas, mas podem ser freqüentes na presença de componentes móveis.

Após a ocorrência de uma falha, é necessário que o sistema (r)estabeleça um estado global consistente, ou seja, um estado global que poderia ter sido analisado por um observador onisciente externo [3]. A recuperação pode ser feita por avanço (dependente da aplicação) ou por retrocesso de estado (independe da aplicação) [5, cap. 7].

Um *checkpoint* é um estado de interesse de um componente de uma computação distribuída escolhido para ser armazenado em memória estável e a atividade de *checkpointing* corresponde

à seleção de *checkpoints* feita por todos os componentes. Vários protocolos foram propostos na literatura para *checkpointing* e recuperação por retrocesso em redes fixas [3, 4, 6]. Mais recentemente surgiram trabalhos para redes móveis estruturadas [1, 2], mas não conhecemos nenhum trabalho sobre recuperação por retrocesso em redes ad hoc.

Em um sistema distribuído assíncrono, devido ao atraso arbitrário das mensagens, é impossível distinguir uma partição real de uma virtual [7]. Além disso, permitir que o sistema progrida na presença de partições pode levar a inconsistências. Provavelmente esta complexidade e a baixa ocorrência de falhas que resultam em partição em redes fixas induziram os protocolos existentes para recuperação por retrocesso a ignorar a possibilidade de partições.

Propomos a extensão dos protocolos existentes para *checkpointing* e recuperação de falhas de maneira a incorporar a possibilidade de (i) divisão do sistema em uma ou mais partições, (ii) avanço independente de cada partição, (iii) recuperação parcial em caso de falha em um dos componentes de uma partição e (iv) agrupamento ou reconfiguração das partições. No momento da reunificação das partições, pode ser detectada uma inconsistência que deverá ser corrigida através de recuperação por retrocesso ou por avanço.

2 Redes Ad Hoc

Uma aplicação distribuída em uma rede ad hoc é composta por um conjunto de nós móveis ($MH_1, \ldots, MH_n$), onde os nós móveis podem trocar informações entre si (desde que cada nó esteja dentro da área de alcance do outro) como ilustra a Figura 1 (a). Os nós das redes ad hoc podem se mover de forma arbitrária de modo que a topologia da rede muda freqüentemente, dificultando o roteamento e a localização dos nós móveis. O distanciamento pode inclusive ocasionar a partição da rede, como ilustram as Figuras 1 (b, c, d, e, f).

Graças ao avanço tecnológico, esses computadores possuem memória cada vez mais rápida e com maior capacidade. Utilizaremos esta memória para o armazenamento de *checkpoints* e replicação destes *checkpoints* em nós vizinhos com o objetivo de tolerar falhas de perda, roubo ou danos físicos.

3 Checkpointing na Presença de Partições

Propomos a extensão dos protocolos existentes para *checkpointing* e recuperação de falhas de maneira a incorporar a possibilidade de:

(i) divisão do sistema em uma ou mais partições—devida à falha em componentes, canais de comunicação ou ao distanciamento dos computadores em redes móveis.

(ii) avanço independente de cada partição— pode não ser desejável esperar pela reunificação completa dos componentes para continuar a computação.

(iii) recuperação parcial restrita a uma partição—a recuperação deve ser feita com a colaboração de todos os componentes desta partição.

(iv) agrupamento ou reconfiguração das partições— pode ser detectada uma inconsistência que deverá ser corrigida através de recuperação por retrocesso ou por avanço.

Ilustraremos estes quatro passos com o auxílio do diagrama espaço-tempo apresentado na Figura 2, cujas partições são mostradas através do cenário da Figura 1.

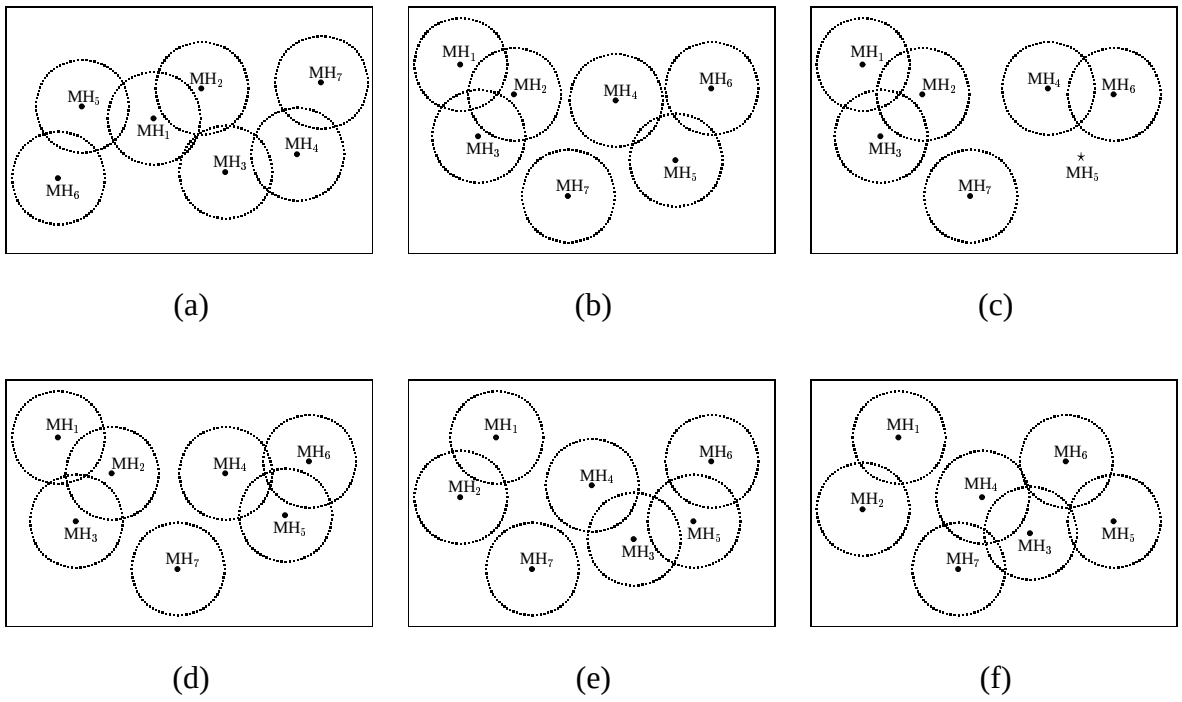

Figura 1: Rede ad hoc

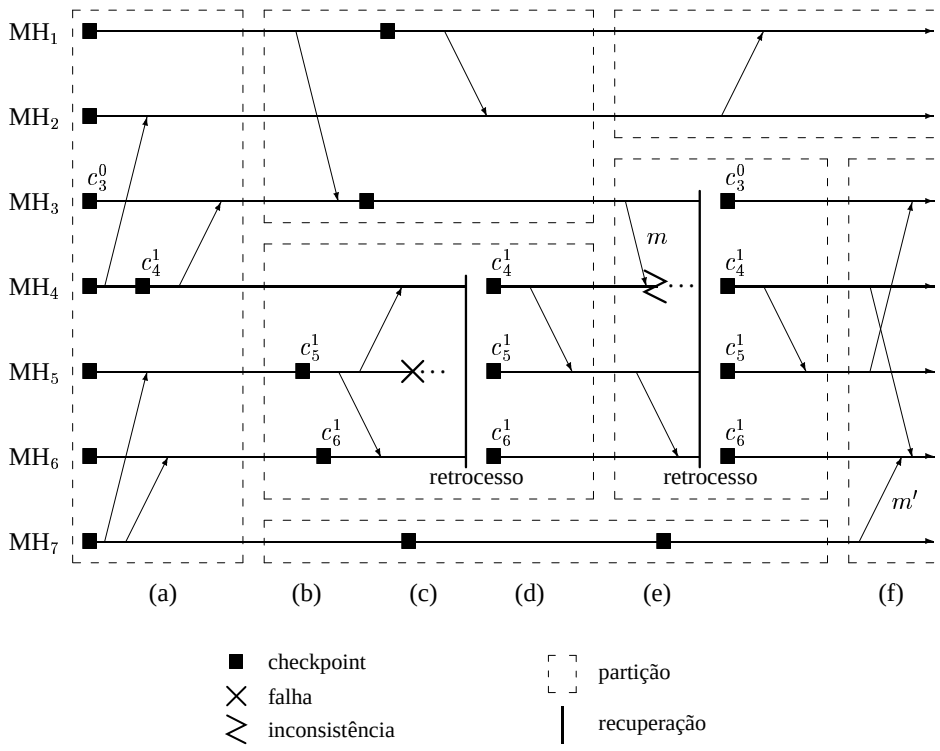

Figura 2: Reunificação de partições com recuperação por retrocesso

Imediatamente antes de começar a computação, todos os componentes (MH_1,...,MH_7) tiram um *checkpoint* inicial. A execução será descrita a seguir:

(a) todos conseguem se comunicar.

(b) divisão do sistema em três partições: (MH_1, MH_2, MH_3), (MH_4, MH_5, MH_6) e (MH_7).

(c) ocorre uma falha no componente MH_5.

(d) MH_5 volta a processar mas terá de retroceder a um *checkpoint* armazenado previamente. Todos os componentes da partição devem retroceder para um estado consistente e portanto MH_4, MH_5 e MH_6 retornam respectivamente para os *checkpoints*: c_4^1, c_5^1 e c_6^1.

(e) o sistema sofre uma reconfiguração formando três novas partições: (MH_1, MH_2), (MH_3, MH_4, MH_5, MH_6) e (MH_7). Na recepção da mensagem m, uma inconsistência é detectada. Na Figura 2, MH_3, MH_4, MH_5 e MH_6 devem retroceder para um estado consistente, ou seja, retornam respectivamente para os *checkpoints*: c_3^0, c_4^1, c_5^1 e c_6^1.

(f) o sistema sofre uma reconfiguração formando duas partições: (MH_1, MH_2) e (MH_3, MH_4, MH_5, MH_6, MH_7). Neste momento, MH_6 recebe uma mensagem m' de MH_7 e neste caso não ocorre inconsistência e a computação segue normalmente.

4 Conclusão

Um sistema distribuído pode ser particionado pela ocorrência de falhas ou por desconexão de um de seus componentes. Em uma rede ad hoc, o particionamento e a reunificação do sistema ocorre com freqüência pela mobilidade dos componentes. Este trabalho analisa as características desta rede e os pontos importantes que devemos considerar na extensão dos protocolos para *checkpointing* de rede fixa para as rede ad hoc.

Referências

[1] A. Acharya and B. R. Badrinath. Checkpointing Distributed Applications on Mobile Computers. In *International Conference on Parallel and Distributed Information Systems*, Sept. 1994.

[2] G. Cao and M. Singhal. Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing Systems. *IEEE Trans. on Parallel and Distributed Systems*, 12(2):157–172, Feb. 2001.

[3] M. Chandy and L. Lamport. Distributed Snapshots: Determining Global States of Distributed Systems. *ACM Trans. on Computing Systems*, 3(1):63–75, Feb. 1985.

[4] E. N. Elnozahy, D. Johnson, and Y.M.Yang. A Survey of Rollback-Recovery Protocols in Message-Passing Systems. Technical Report CMU-CS-96-181, Carnegie Mellon University, 1996.

[5] P. A. Lee and T. Anderson. *Fault Tolerance: Principles and Practice.* Second, Revised Edition, Springer-Verlag, New York, 1990.

[6] D. Manivannan and M. Singhal. Quasi-Synchronous Checkpointing: Models, Characterization, and Classification. Technical Report OH 43210, Department of Computer and Information Science, The Ohio State University, 1997.

[7] A. Ricciardi, A. Schiper, and K. Birman. Understanding Partitions and the "No Partition" Assumption. In *Proceedings of the 4th IEEE Computer Society Workshop on Future Trends in Distributed Computing Systems*, pages 354–360, Lisboa, Portugal, 1993.