Programa de Pós-Graduação em Engenharia de Automação e Sistemas
Universidade Federal de Santa Catarina

Técnicas para Detecção, Tolerância e Recuperação de Faltas Temporais

Projeto de Dissertação de Mestrado
Orientador: Rômulo Silva de Oliveira
Co-Orientadora: Patricia Plentz (INE-UFSC)

1. Introdução

Sistemas computacionais de tempo real são identificados como aqueles sistemas computacionais submetidos a requisitos de natureza temporal. Nestes sistemas, os resultados devem estar corretos não somente do ponto de vista lógico, mas também devem ser gerados no momento correto. As falhas de natureza temporal nestes sistemas são, em alguns casos, consideradas críticas no que diz respeito às suas conseqüências.

Na literatura, os sistemas de tempo real são classificados conforme a criticalidade dos seus requisitos temporais. Nos sistemas tempo real críticos (*hard real-time*) o não atendimento de um requisito temporal pode resultar em consequências catastróficas tanto no sentido econômico quanto em vidas humanas. Quando os requisitos temporais não são críticos (*soft real-time*) eles apenas descrevem o comportamento desejado. O não atendimento de tais requisitos reduz a utilidade da aplicação mas não a elimina completamente nem resulta em consequências catastróficas.

Na medida em que o uso de sistemas computacionais prolifera em nossa sociedade, aplicações com requisitos de tempo real tornam-se cada vez mais comuns. Estas aplicações variam muito com relação ao tamanho, complexidade e criticalidade. Entre os sistemas mais simples estão os controladores embutidos em utilidades domésticas, tais como lavadoras de roupa. Na outra extremidade deste espectro estão os sistemas embutidos em veículos e o controle de tráfego aéreo.

Com aplicações mais complexas, executando em computadores mais poderosos, surge a necessidade de empregar sistemas operacionais com bom comportamento temporal e que facilitem o atendimento dos requisitos temporais. Tais sistemas são normalmente chamados de sistemas operacionais de tempo real. Podemos citar como exemplo o FreeRTOS (www.freertos.org) e o Linux Preempt-RT (wiki.linuxfoundation.org/realtime).

Aplicações de tempo real aparecem muitas vezes em um contexto onde falhas temporais podem ter consequências graves. Temos uma falha temporal quando um requisito temporal deixa de ser atendido. Na terminologia da área, temos uma falta (*fault*) quando um componente do sistema deixa de funcionar de forma apropriada. Uma falta pode levar o sistema a um estado de erro (*error*). Caso o erro seja manifestado para o usuário do sistema, violando requisitos da especificação, temos então uma falha (*failure*).

Faltas temporais podem surgir de várias maneiras, por exemplo:

- O tempo de execução de uma tarefa foi previsto com muito otimismo, a tarefa executa mais do que deveria, gerando maior interferência sobre tarefas de mais baixa prioridade;

- Tempos de bloqueio em mutex e outros mecanismos de sincronização podem durar mais do que o previsto;

- A análise de escalonabilidade feita durante o desenvolvimento não contemplou todos os cenários possíveis ou foi baseada em premissas erradas (otimistas);

- A carga de tarefas durante a execução do sistema excede o que foi previsto no projeto, por exemplo interrupções são geradas com frequência maior do que o suposto no projeto.

A falta associada com uma tarefa pode propagar-se no sistema. Por exemplo, uma tarefa executar além do previsto pode fazer outra tarefa perder o seu deadline. Isto pode acontecer com tarefas de baixa prioridade através do uso do processador e pode acontecer com tarefas de alta prioridade, através de tempos de bloqueio além do previsto.

Como o requisito mais comum e relevante das aplicações de tempo real é o deadline, a detecção de falta corresponde a detectar perdas de deadlines. Existem várias soluções de design para isto. A mais conhecida é o temporizador *watch-dog*, o qual dispara um alarme caso não seja resetado de tempos em tempos. De uma maneira mais genérica, pode-se incluir na aplicação threads responsáveis por monitorar seu comportamento e detectar faltas temporais o mais cedo possível.

Uma vez detectada, a falta temporal pode ser tratada de alguma forma, para evitar que ela evolua para uma falha temporal. Quando isto acontece, o sistema é dito tolerante a faltas. Existem inúmeros métodos de tolerância a faltas na literatura, variando em cobertura e custos de implementação. E nem todos eles são viáveis para aplicação em sistemas de tempo real.

No caso da falta se propagar e gerar uma falha temporal, existem na literatura técnicas para realizar o controle de danos (*damage confinement*) no sistema, o que pode ser muito importante em sistemas críticos. O papel do controle de danos relacionados com falhas temporais é prevenir a propagação dos erros para outros componentes do sistema. Idealmente, haveria um isolamento temporal entre componentes de maneira que eles possam ser considerados separadamente no que diz respeito a falhas temporais.

2. Objetivo

Estudar aspectos da detecção e tolerância a faltas temporais em sistemas de tempo real. Avaliar a aplicabilidade dos mecanismos clássicos de tolerância a faltas no contexto dos sistemas de tempo real. Recomendar as técnicas mais apropriadas para este tipo de sistema.

A dissertação deverá incluir um estudo da bibliográfia relativa à análise de tempo de resposta e também aos princípios da tolerância a faltas. Será escolhida uma plataforma para testes composta por computador e sistema operacional e implementada uma aplicação para estudo de caso.

Espera-se ao final que o mestrando conheça a teoria de escalonamento tempo real em geral, os principais mecanismos de tolerância a faltas, e a problemática da aplicação de tais mecanismos em sistemas de tempo real. O estudo deverá mostrar técnicas de detecção e tolerância a faltas para sistemas de tempo real, e sua validação experimental no estudo de caso.

3. Lista de Atividades

(a) Estudar os conceitos básicos dos sistemas de tempo real.

(b) Estudar a análise de tempo de resposta para sistemas de tempo real.

(c) Estudar a terminologia clássica da área de tolerância a faltas.

(d) Estudar mecanismos para detectar faltas temporais.

(e) Estudar mecanismos clássicos de tolerância a faltas.

(f) Selecionar mecanismos de tolerência a faltas apropriados para faltas temporais.

(g) Escolher a classe de aplicações de tempo real alvo deste estudo e as plataformas típicas de arquitetura de computador e de sistema operacional que serão consideradas.

(h) Implementar uma aplicação de tempo real que servirá de estudo de caso.

(i) Implementar e avaliar mecanismos de detecção e de tolerância a faltas temporais.

(j) Estudar mecanismos clássicos de recuperação de falhas.

(k) Implementar e avaliar mecanismos de recuperação de falhas temporais.

(l) Preparar recomendações para desenvolvedores de aplicações de tempo real no que concerne a detecção e tolerância a faltas temporais.

(m) Redigir artigos para publicação em congressos e/ou revistas.

(n) Redigir a dissertação.

4. Bibliografia Parcial

[1] R. I. Davis and A. Burns, A survey of hard real-time scheduling algorithms and schedulability analysis techniques for multiprocessor systems, Systems Research, 2009.

[2] G. Buttazzo, Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. Springer, 2 ed., 2005.

[3] J. W. S. Liu, Real-Time Systems. Prentice Hall, 1 ed., 2000.

[4] D. B. de Oliveira, R. S. de Oliveira, Timing Analysis of the PREEMPT RT Linux Kernel, Software Practice and Experience, Volume 46, Issue 6, pages 789–819, June 2016.

[5] N. C. Audsley, A. Burns, R. I. Davis, K. W. Tindell, A. J. Wellings, Fixed priority pre-emptive scheduling: An historical perspective, Real-Time Systems, March 1995, Volume 8, Issue 2–3, pp 173–198.

[6] R. I. Davis, A. Zabos, A. Burns, IEEE Transactions on Computers, Efficient Exact Schedulability Tests for Fixed Priority Real-Time Systems Volume 57, Issue 9, Sept. 2008.

[7] I. Koren, C. M. Krishna, Fault-Tolerant Systems, Elsevier, 2010.