Capítulo 6

Tendências Atuais em Sistemas de Tempo Real

Sistemas de Tempo Real são reconhecidos por possuírem problemas bem definidos e únicos. Um conjunto de técnicas, métodos, ferramentas e fundamentação teórica formam uma disciplina necessária na compreensão e na concepção de sistemas de tempo real. Apesar da evolução nos últimos anos, em termos de conceitos e métodos, os meios mais convencionais continuam a ser usados na prática. Além do aspecto dessas ferramentas usuais não tratarem com a correção temporal, um outro fato que se pode considerar é o relativo desconhecimento do que seria tempo real. A grande maioria dos sistemas de tempo real é projetada a partir de um entendimento errado que reduz tempo real, a simplesmente uma questão de melhoria no desempenho.

A literatura de tempo real tem sido pródiga no definição de modelos, metodologias e suportes que consideram restrições temporais. Foi no intuito de melhorar o conhecimento desta disciplina que nos propomos neste texto a apresentar duas abordagens para a construção de STRs: a síncrona e a assíncrona.

6.1 Abordagem Síncrona

A abordagem síncrona é baseada na hipótese síncrona que, em seus princípios básicos, considera os processamentos e comunicações instantâneos. O tempo é visto com granularidade suficientemente grossa para que essas premissas sejam aceitas como verdadeiras. A observação dos eventos nessa abordagem é cronológica.

A abordagem síncrona é considerada como orientada ao comportamento de uma aplicação e à sua verificação. Por se situar num nível de abstração maior, os aspectos de implementação são em grande parte desconsiderados, facilitando a especificação e a análise das propriedades de sistemas de tempo real. A abordagem síncrona forma uma base conceitual bastante sólida permitindo a definição de conjuntos de ferramentas integradas próprias para o desenvolvimento de aplicações de tempo real. Entretanto, aplicações complexas que envolvem programas de calculo, grande quantidade de dados a manusear e distribuição são dificilmente tratáveis por completo pela abordagem síncrona. A abordagem síncrona é bem adaptada para sistemas reativos ou partes reativas de sistemas complexos que são dirigidos pelo ambiente e para os quais o determinismo no comportamento é fortemente desejável.

Os sistemas reativos para os quais o uso da abordagem síncrona é uma solução recomendada se encontram nos seguintes campos de aplicação: controle de processo tempo real, sistemas de manufatura, sistemas de transporte, sistemas embutidos, sistemas autônomos, sistemas de supervisão, protocolos de comunicação, "*drivers*" de periféricos, interface homem-máquina, processamento de sinais, entre outros.

Geralmente, as linguagens síncronas não são suficientes para a programação completa dos sistemas complexos. As partes não reativas têm que ser construídas usando a abordagem assíncrona. As linguagens assíncronas devem fornecer mecanismos para a interação entre as partes reativas e não reativas segundo um modo de comunicação assíncrona. Propostas de formalismos ou de metodologias que fazem uso de modos de comunicação síncronos e assíncronos em conjunto podem ser encontradas em [SBS93] e [BCJ97].

A distribuição do código em vários computadores é uma característica indispensável dos sistemas complexos por uma ou mais das razões seguintes: aumento da capacidade de cálculo, melhoria de desempenho, tolerância a falhas, localização geográfica dos sensores e atuadores, entre outras. As dificuldades encontradas para tratar a distribuição na abordagem síncrona provem da necessidade, imposta pela hipótese de sincronismo, da existência de um relógio global e, conseqüentemente, de um algoritmo (síncrono) de sincronização de relógios. A distribuição de código pode ser feita de três formas diferentes [CGP99]:

(i) Compilando separadamente cada parte do programa total, e alocando aos computadores do sistema cada uma destas partes que devem se comunicar. Apesar de ser uma solução fácil, nem sempre a compilação em separado gera programas seqüenciais deterministas [Maf93];

(ii) Compilando o programa todo e gerando programas seqüenciais para cada computador, cada programa podendo se comunicar com os outros, segundo o *método de grafo abstrato* apresentado em [Maf93];

(iii) Compilando o programa num único programa objeto e distribuindo-o posteriormente em tantos programas quanto computadores que realizarão apenas os cálculos que lhes correspondem [CaK88] e [CPG99]. Esta abordagem tem a vantagem de otimizar a compilação e de permitir a verificação e depuração do programa antes de distribui-lo.

Maiores detalhes sobre o estado atual das pesquisas sobre distribuição de programas síncronos podem ser encontradas em [CGP99], [BCL99] e [BeC99].

6.2 Abordagem Assíncrona

A abordagem assíncrona visa uma descrição a mais exata possível de um sistema de

tempo real e por isto é considerada como orientada à implementação. Vários aspectos referentes a uma aplicação são tratadas explicitamente a nível de programação e gerenciadas em tempo de execução; o tempo e a concorrência são exemplos deste conhecimento explicito necessário a um programador de aplicações de tempo real nesta abordagem. Como conseqüência, torna-se necessário levar em conta já na especificação e no decorrer do projeto, algumas características dos suportes de software e de hardware. Por outro lado, na abordagem assíncrona, a procura de uma descrição completa do comportamento e a introdução de considerações de implementação torna complexa a análise das propriedades do sistema de tempo real.

Num sentido mais clássico, dentro desta ótica da abordagem assíncrona, uma fundamentação teórica já é bem definida para sistemas onde é possível uma previsibilidade determinista. Uma coleção de algoritmos e técnicas de analise existem para problemas envolvendo escalonamentos com garantias em tempo de projeto, caracterizando o que Stankovic chama de *ciência da garantia de desempenho* [Sta96]. Os sistemas, neste perspectiva de garantia em tempo de projeto, são de dimensões não muito extensas, construídos para realizar um conjunto específico de objetivos e envolvendo, em termos de implementação, soluções ditas proprietárias. Estão dentro desta ciência da garantia do desempenho as abordagens que tratam com ambientes dinâmicos, ainda limitados, usando técnicas para obter garantias dinâmicas. Estas abordagens envolvem testes de aceitação baseados em hipóteses de pior caso dos tempos de computação da carga presente no sistema. O estudo apresentado neste texto de certa maneira cobriu estas técnicas que formam esta ciência da garantia.

Como os sistemas se tornam cada vez maiores e mais complexos, interagindo com ambientes também complexos e dinâmicos – e, portanto, não deterministas – uma expansão desta *ciência da garantia* é necessária para captar as características destes novos sistemas. As técnicas existentes, mesmo as de abordagens assíncronas que tratam com garantias dinâmicas, não são abrangentes o suficiente para que possam ser eficazes diante destes novos sistemas.

A evolução prevista para os próximos anos caracteriza sistemas como sendo distribuídos, de larga escala, oferecendo uma grande variedade de serviços que trata com várias formas de dados e se apresentam constantemente mudando e evoluindo. Em contraste, um desafio crescente colocado a comunidade de tempo real é como construir esses sistemas de propósito geral, abertos, que permitam conviver várias aplicações independentes e de tempo real.

Uma perfeita análise de escalonabilidade a priori é impraticável em um ambiente destes. É necessário uma nova disciplina, com modelos e abordagens criados no sentido de captar as complexidades desses novos ambientes que estão se caracterizando envolvendo novas tecnologias e aplicações. Esta nova disciplina para aplicações de tempo real vai exigir esforços de pesquisa na engenharia de software, em linguagens de programação, em sistemas operacionais, na teoria de escalonamento e na comunicação.

Sistemas Abertos

A meta principal nesta tendência é o desenvolvimento de arquiteturas, suportes de "*middlewares*" e componentes com interfaces públicas e bem definidas, que possam ser desenvolvidos de forma independente, para posteriormente serem combinados e usados na construção de sistemas complexos. Esta estratégia conduz à obtenção, além de grande flexibilidade, de vantagens como melhor portabilidade, interoperabilidade e facilidades na evolução dos sistemas. Estas interfaces favorecem também o uso de componentes/softwares de prateleira ("*off-the-shelf*").

As tecnologias abertas para sistemas distribuídos de tempo real são recentes. O RT CORBA [OMG98] e o ANSAware/RT [Li95] são exemplos destes esforços de organizações padronizadoras na definição de padrões para componentes de "*middleware*" que suportem aplicações distribuídas de tempo real. E, mesmo para a automação industrial e sistemas embutidos em geral, estão previstos certos requisitos para as próximas gerações que envolvem padrões abertos [Sta96]: os controladores (computadores especializados) devem apresentar arquitetura modular que suporte mudanças em tempos mínimos sem comprometer os requisitos de desempenho de uma planta industrial; devem suportar facilmente a adição de funcionalidade ou a alteração no comportamento da aplicação e, também, não depender de fabricante ou tecnologias proprietárias.

A adequação de padrões abertos para o desenvolvimento de aplicações de tempo real sempre foi vista com uma certa desconfiança por projetistas da área. A necessidade de previsibilidade no atendimento das restrições temporais, historicamente tem contribuído para soluções proprietárias, específicas para suas aplicações alvo. Algumas das dificuldades de uma arquitetura aberta para aplicações de tempo real em sistemas distribuídos de larga escala incluem:

- Características de hardware desconhecidas até o tempo de execução (velocidades de processadores, "*caches*", barramentos , memória variam de máquina para máquina).

- Ambientes onde convivem diferentes aplicações onde suas necessidades de recursos e seus requisitos temporais são desconhecidos até o tempo de execução.

Linguagens

Até recentemente, muito pouco tinha sido feito em termos de linguagens de programação para sistemas de tempo real. Na prática corrente, no uso de abordagens assíncronas em muitas aplicações, as partes críticas das aplicações eram (ou ainda são) implementadas em códigos de máquina ou *"Assembly"*. Isto forçava um conhecimento

prévio sobre a arquitetura dos processadores usados no sistema. As implicações são muitas no uso desta prática: os códigos produzidos não são portáveis, e a própria modificação ou evolução do sistema fica comprometida. Todos esses fatores apontam para a necessidade do uso de linguagens de programação de alto nível com construções explícitas que facilitem a descrição do comportamento temporal de uma aplicação de tempo real.

Em quase todos os modelos de escalonamento citados no capítulo 2, é assumido o pior caso nos tempos de computação dos códigos das tarefas. Os códigos gerados na compilação destas linguagens de alto nível devem ter, portanto, bem determinados os valores de seus piores tempos de computação. Para a obtenção destes tempos é necessário evitar a utilização de certas construções de linguagens convencionais e certas práticas correntes de programação. O uso de estruturas dinâmicas deve também ser limitado.

São necessárias ferramentas para preverem o pior caso e o tempo de computação médio destes códigos. A estimação do pior caso pode ser obtida por análises ou por medidas. Algumas ferramentas existentes, baseada em análises do código, são protótipos acadêmicos [Gus94], [Har94]. A dificuldade de métodos baseados em análise é a necessidade de um modelo que capte todas as características da arquitetura do computador ("*cache*", "*pipeline*", etc). O problema de técnicas baseadas em medidas está na dificuldade de se conseguir o pior caso de tempo de computação a partir de um certo número de ativações sob o qual é submetido o código de uma tarefa.

As linguagens de programação em geral são fracas no sentido de suportarem a implementação das abordagens de escalonamento de tempo real. A falta de expressividade fica evidente quando o objetivo é a generalização de requisitos temporais e de estratégias de implementação. É improvável que uma única linguagem seja abrangente o suficiente para acomodar todos os requisitos temporais das abordagens existentes e mais os novos que certamente deverão surgir nos próximos anos. São necessárias novas estruturas de linguagens mais flexíveis que permitam a especialização diante dos diferentes tipos de aplicações de tempo real. A Reflexão Computacional, através dos protocolos Meta-Objeto ([Mae87], [TaT92]), e a Programação Orientada a Aspectos [Kic97] são exemplos de novas técnicas e mecanismos de programação que são incentivadas porque trazem esta flexibilidade necessária na programação em tempo real. Ambas técnicas enfatizam a separação das funcionalidades de uma aplicação de seu gerenciamento.

Recentemente, o grande sucesso da linguagem Java no contexto da Internet e mesmo fora dela, atraiu a atenção das pessoas que desenvolvem software de tempo real. Na sua forma original Java não é apropriada para tempo real devido às diversas fontes de não determinismo presentes. O exemplo clássico é o seu coletor de lixo, o qual pode executar a qualquer momento e ocupar o processador por um tempo não desprezível, sempre que o suporte ficar sem memória para instanciar novos objetos. Existem dois fortes consórcios definindo uma versão de Java para tempo real (*Real-Time Java*, [Ort99]). Embora neste momento não esteja claro qual será o consórcio vencedor, o fato das maiores empresas de computação do mundo participarem deles é um sinal de que

Java poderá ser usada no futuro também para sistemas de tempo real.

Sistemas Operacionais

O mercado de sistemas operacionais de tempo real está passando por uma fase de rápidas mudanças. A teoria de escalonamento de tempo real, ignorada até alguns anos atrás, começa a ser incorporada aos sistemas operacionais. Desta forma, pode-se esperar para os próximos anos suportes de execução cada vez mais previsíveis. Ao mesmo tempo, Posix firmou-se como a interface padrão nesta área. Embora o próprio Posix seja grande e por vezes ambíguo, ele estabelece um padrão para as chamadas de sistema a serem oferecidas para aplicações de tempo real.

Variações do Linux para tempo real estão surgindo com força neste mercado. Como o capítulo 3 mostrou, muitos grupos de pesquisa em todo o mundo estão trabalhando no sentido de criar versões do Linux apropriadas para aplicações de tempo real. Enquanto alguns trabalhos nesta área buscam uma previsibilidade rigorosa para atender sistemas críticos, outros procuram melhorar o desempenho do escalonamento para melhor suportar aplicações com requisitos temporais brandos. Não é possível ignorar que a disponibilidade de um sistema operacional Unix completo, com fonte aberto e adaptado para tempo real, vai causar um grande impacto no mercado. Por exemplo, no momento em que este livro é escrito, é anunciado que o QNX será fornecido sem custo para aplicações não comerciais (http://get.qnx.com).

Com respeito aos sistemas distribuídos, ainda existe um longo caminho a ser percorrido. Embora RT-CORBA padronize interfaces e crie uma estrutura conceitual baseada em objetos, sua implementação depende dos serviços de escalonamento e comunicação tempo real fornecidos pelo sistema operacional e protocolos de comunicação subjacentes. A medida que a qualidade destes serviços aumentar, implementações do RT-CORBA também apresentarão melhor qualidade com respeito a previsibilidade temporal.

Apesar de todos estes esforços, a verdade é que o mercado de sistemas operacionais de tempo real continuará segmentado ainda por muitos anos. Com aplicações tão diversas quanto uma teleconferência e o controle de um motor elétrico apresentando restrições temporais, obviamente existe espaço para um grande conjunto de soluções. Estas incluem desde sistemas operacionais completos, adaptados para fornecer escalonamento do tipo melhor esforço, até núcleos totalmente previsíveis, capazes de garantir *"deadlines"* em aplicações críticas.

Escalonamento de Tempo Real

A evolução prevista com novos ambientes de larga escala e novas aplicações preconiza o surgimento de novas abordagens com algoritmos e testes de escalonamento

que possam tratar com estruturas complexas de tarefas e de recursos, definindo escalas que atendam granularidades variáveis de restrições temporais [Sta96]. As técnicas de escalonamento devem ser robustas ou ainda, flexíveis e adaptativas quando necessário.

Existe uma necessidade de testes e algoritmos mais robustos. Na chamada *ciência da garantia* testes e algoritmos, definidos segundo certas premissas, são válidos para determinados modelos de tarefas. Quando aplicados em um sistema, a atuação de um algoritmo é tida como correta se todas as premissas assumidas no modelo são válidas. Se algumas dessas premissas não são verificadas e mesmo assim as propriedades do algoritmo se mantém corretas, as restrições de tempo real da aplicação são atendidas e o algoritmo é dito robusto. O uso de um algoritmo robusto reduz, significativamente, a necessidade da caracterização da aplicação e de seu ambiente de execução nos esforços de análises e medidas para a validação do conjunto de tarefas que formam a sua carga [Sta96]. A eficiência e a robustez podem ser concretizadas facilmente se o grau de sucesso da validação não é o objetivo maior. Testes de aceitação (ou testes de escalonabilidade), por serem excessivamente pessimistas, apresentam um baixo grau de sucesso nestes novos e complexos ambientes.

As abordagens de *escalonamento adaptativo* ("*adaptive scheduling*") têm sido propostas no sentido de se adotar modelos mais flexíveis. Essas abordagens assumem que as condições do sistema são monitoradas, e as decisões do escalonamento são baseadas nas condições observadas [LSL94]. Modelos e abordagens de escalonamento adaptativos são empregados com objetivo de lidar com a incerteza na carga do sistema e a obtenção de uma *degradação suave* [CLL90], [GNM97], [MFO99], [TaT93]. Degradação suave em sistemas de tempo real, significa a manutenção das propriedades de previsibilidade na sobrecarga.

De forma diversa às abordagens mais clássicas, algumas abordagens adaptativas não necessitam o conhecimento *a priori* dos piores casos de tempo de computação das tarefas. A determinação destes tempos das tarefas, necessários em abordagens mais clássicas, sempre é um trabalho árduo.

Diversas técnicas adaptativas têm sido propostas recentemente. Por exemplo, uma técnica de adaptação pode ser baseada no ajuste dos períodos de tarefas periódicas em tempo de execução. A adaptação dos períodos de tarefas pode ser usada em alguns sistemas de controle realimentados ou também, através da mudança da freqüência de apresentação de quadros, em uma aplicação de vídeo sob demanda [KuM97]. A computação imprecisa [LSL94] — uma outra técnica que permite a combinação de garantia determinista com degradação suave — é usada para o tratamento de sobrecargas. Nessa técnica, cada tarefa é decomposta em duas partes: uma parte obrigatória e uma parte opcional. A parte obrigatória de uma tarefa deve ser completada antes do deadline da tarefa para produzir um resultado aproximado com uma qualidade aceitável. A parte opcional refina o resultado produzido pela parte obrigatória. Durante uma sobrecarga, um nível "mínimo" de operação do sistema pode ser garantido de forma determinista, através da execução apenas das partes obrigatórias.

Em [HaR95], foi proposto o conceito de "*deadline (m,k)-firm*", definindo que uma

tarefa periódica deve ter pelo menos *m "deadlines"* atendidos em cada janela de *k* ativações. O limite superior tolerado de perdas de *"deadlines"* é dado por *k-m*. Uma falha dinâmica é assumida no sistema quando esse limite é excedido. O DBP ("*Distance Based Priority Assignment*") é a heurística de escalonamento usada com essa técnica no sentido de minimizar o número de falhas dinâmicas. Essa heurística de escalonamento atribui as mais altas prioridades para as tarefas que estão próximas de exceder o limite superior especificado para as perdas de *"deadlines"*. Outras técnicas semelhantes "*deadline (m,k)-firm*" foram introduzidas na literatura [KoS95], [CaB97] e [BeB97]. Todas se utilizam do descarte de certas ativações de tarefas periódicas para aumentar a escalonabilidade do sistema.

Comunicação

Se considerarmos os sistemas de tempo real distribuídos, a comunicação desempenha um papel importantíssimo nos tempos de resposta que envolve uma aplicação nestes sistemas. Os objetivos de protocolos de comunicação em sistemas de tempo real são diferentes dos sistemas que não são de tempo real. Em sistemas mais convencionais, não tempo real, o aumento do desempenho na forma de taxas de transferências é o ponto chave. Por outro lado, na comunicação em sistemas de tempo real o que se procura é a obtenção de altas probabilidades que uma mensagem será entregue dentro de um deadline específico ou atendendo uma latência máxima.

Em sistemas críticos ("*hard real-time systems*") e, portanto na ótica da *ciência da garantia*, os protocolos de comunicação para tempo real devem apresentar um limite máximo na latência de uma mensagem durante uma comunicação. Em sistemas mais brandos, tais como multimídia e videoconferência, onde dados de diferentes naturezas estão sendo transmitidos, é evidente a necessidade da entrega de mensagens segundo certas restrições temporais. Ocasionais falhas em atender estas restrições temporais é perfeitamente admissível; porém, atrasos ou perdas de mensagens excessivos degradam a qualidade do serviço fornecido a nível da aplicação.

São duas as abordagens usadas para tratar comunicações em sistemas de tempo real. A primeira é baseada sobre a utilização do conhecimento sobre a máxima latência nas transferências de mensagens no sistema [ARS91]. As análises e escalas produzidas para o sistema consideram então este pior tempo de comunicação. Na segunda abordagem as mensagens estão sujeitas elas próprias a restrições temporais (*"deadlines"*, períodos, etc) e escalonadores atuam, segundo políticas de tempo real, em filas de emissão levando em conta estas restrições [ZhB94].

Sistemas de larga escala com diferentes tipos de aplicações, implicam na necessidade do suporte a vários tipos de comunicação, sujeitas a diferentes necessidades de garantias e de restrições de tempo. Novas tecnologias de comunicação estão surgindo para suprir essas necessidades. Exemplo destas tecnologias são as redes baseadas no protocolo IP, que inicialmente foram concebidas no sentido das técnicas de melhor

esforço e que estão sendo estendidas para implementar diferentes Qualidades de Serviço (QoS). Redes ATM também suportam diferentes classes de serviço. Além de serviços de melhor esforço estas tecnologias oferecem classes de s*erviços garantidos* que, com reservas de recursos, fornecem serviços garantidos fim-a-fim com o controle rígido da largura de banda e do retardo [SPG97]. Com base nestas tecnologias novos modelos de comunicação de tempo real deverão surgir, possibilitando a associação de parâmetros de QoS relacionados com tempo real a um canal de comunicação.

Metodologias

Atualmente, a complexidade dos sistemas computacionais exige cada vez mais o uso de metodologias para especificar, analisar, projetar e implementar. Os sistemas de tempo real em particular quando distribuídos, apresentam características que se encaixam nesta categoria. As metodologias de propósito geral, normalmente utilizadas na comunidade acadêmica e no setor industrial e de serviços, tem apresentado inconvenientes. As razões estão na inadequação da linguagem de modelagem que não inclui o conjunto especifico de requisitos necessários para representar os sistemas tempo real e, as dificuldades de implementação que não facilitam a ligação entre os conceitos usados numa modelagem de alto nível de abstração e os conceitos usados na implementação. Metodologias para sistemas de tempo real e que ao mesmo tempo acrescentam as potencialidades da orientação objeto têm sido propostas para remediar este problema e diminuir as dificuldades na construção destes sistemas. Destacam-se entre estas linguagens de modelagem e metodologias para tempo real a ROOM ("*Real-Time Object-Oriented Modeling*") [SGW94] e a UML-RT ("*Unified Modeling Language for Real-Time*") [SeR98]. Essas metodologias apresentam as seguintes características: seguir o paradigma de orientação-objeto; permitir a modelagem e a construção de software tempo real; fornecer modelos executáveis em todos os níveis de abstração; permitir um processo de desenvolvimento incremental e iterativo; e facilitar a documentação.

Neste mesmo contexto, técnicas de descrição formal tem sido adotadas num número crescente de aplicações, devido aos benefícios significativos que o rigor de seus formalismos traz para a produção de software de qualidade, sobretudo quando esta qualidade se expressa em termos de consistência, segurança e correção temporal. Muitas pesquisas e desenvolvimentos de ferramentas vem sendo realizadas neste campo nos últimos anos. Esta área é muito ampla e promissora e sai do escopo deste livro; os interessados podem procurar, na ampla bibliografia disponível da área, as referências [HeM96], [BCN95] e [Rus93] que destacamos por apresentarem uma visão geral das técnicas de descrição formal próprias para sistemas tempo real.