Tópicos Adicionais sobre Sistemas de Tempo Real

Fundamentos dos Sistemas de Tempo Real

Rômulo Silva de Oliveira

eBook Kindle, 2018

www.romulosilvadeoliveira.eng.br/livrotemporeal

Outubro/2018

Sumário

- Servidores de Tarefas Aperiódicas
- Multiprocessadores
- Protocolos de Comunicação
- Protocolos para Sincronização de Relógios
- Computação Imprecisa
- Engenharia de Software

Tópicos Adicionais

- Possível visitar praticamente todas as áreas da computação com um olhar de “tempo real”
  - Banco de dados para tempo real
  - Processamento de imagens em tempo real
  - Etc

- Neste capítulo são apresentados vários tópicos adicionais
  - São brevemente descritos

- Objetivo é mostrar que existem trabalhos importantes sobre eles
- Leitor interessado poderá buscar mais informações na literatura da área em questão

Sumário

- **Servidores de Tarefas Aperiódicas**
- Multiprocessadores
- Protocolos de Comunicação
- Protocolos para Sincronização de Relógios
- Computação Imprecisa
- Engenharia de Software

Servidores de Tarefas Aperiódicas 1/8

- Testes de escalonabilidade podem garantir deadlines
- Mas precisam assumir carga limitada
  - Tarefas periódicas
  - Tarefas esporádicas
- Em muitos sistemas existem tarefas aperiódicas
  - Nada pode ser dito sobre seus padrões de chegada

- Existem tarefas aperiódicas sem restrições de tempo real
- Tentar minimizar o tempo médio de resposta
  - Exemplo: Transferência de arquivos de configuração, histórico
  - Exemplo: Interface humano-máquina

Servidores de Tarefas Aperiódicas 2/8

- Testes de escalonabilidade podem garantir deadlines
- Mas precisam assumir carga limitada
  - Tarefas periódicas
  - Tarefas esporádicas
- Em muitos sistemas existem tarefas aperiódicas
  - Nada pode ser dito sobre seus padrões de chegada

- Como executar tarefas aperiódicas

  sem comprometer a garantia dada para os deadlines

  das tarefas periódicas/esporádicas ?

Servidores de Tarefas Aperiódicas 3/8

- Sempre devem ser cumpridos os deadlines garantidos
- O que sobra de processador é fornecido para uma tarefa especial:
  o Servidor de Aperiódicas
- O servidor de aperiódicas:
  - Executa quando isto não compromete as garantias já dadas
  - Não é uma tarefa de verdade
  - Usa seu tempo para executar os jobs aperiódicos que chegam
  - Jobs aperiódicos formam uma fila que é atendida pelo servidor de aperiódicas
  - Podem existir apenas um ou vários servidores de aperiódicas

- A questão central é:
  quando executar o servidor de aperiódicas ?

Servidores de Tarefas Aperiódicas 4/8

- Quando executar o servidor de aperiódicas ?
- Existem muitos tipos de servidores na literatura
- Alguns para prioridade fixa outros para prioridade variável
- Alguns são capazes de fornecer garantia dinâmica mais facilmente

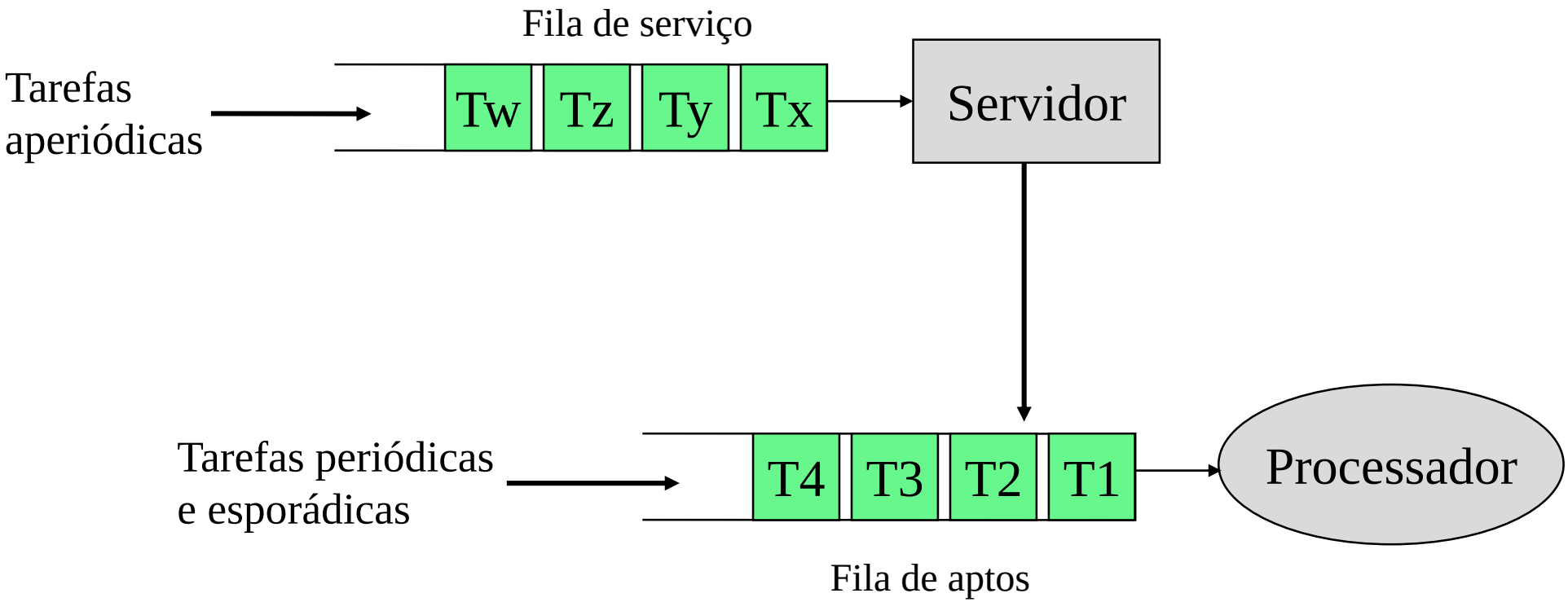
- Principal diferença está em como as sobras de tempo de processamento são detectadas

  ou seja

  Como a capacidade (*budget*) do servidor é reabastecida

Servidores de Tarefas Aperiódicas 5/8

- Duas filas são normalmente usadas



Servidores de Tarefas Aperiódicas 6/8

- **Servidor de Background**

- Executa quando o processador está idle
- Pode ser usado com prioridade fixa ou variável facilmente
- Simples de implementar
- Não afeta a escalonabilidade do sistema

- Problema: tempos de resposta elevados para os jobs aperiódicos
  - É possível melhorar isto

Servidores de Tarefas Aperiódicas 7/8

- **Servidor Periódico (polling server)**
- Uma tarefa periódica é criada para atender a carga aperiódica
- A tarefa servidora possui um período P_{PS} e um tempo máximo de execução C_{PS} a cada período (capacidade nominal)
- Ela é escalonada como uma tarefa periódica normal
- Análise de escalonabilidade para tarefas periódicas pode ser usada
- Em cada ativação
  - A tarefa servidora executa as requisições aperiódicas pendentes dentro do limite de sua capacidade, com sua prioridade natural (fixa ou variável)
- Quando não houver requisições aperiódicas pendentes
  - A tarefa servidora suspende-se até o início do próximo período
  - Neste caso, a sua capacidade é zerada até o próximo período
  - Sua capacidade é reabastecida com C_{PS} no início do próximo período

Servidores de Tarefas Aperiódicas 8/8

- Existem muitos outros algoritmos para servidores

- Prioridade Fixa
  - Priority Exchange Server
  - Sporadic Server
  - Etc

- Prioridade Variável (EDF)
  - Total Bandwidth Server
  - Constant Bandwidth Server
  - Etc

Sumário

- Servidores de Tarefas Aperiódicas
- **Multiprocessadores**
- Protocolos de Comunicação
- Protocolos para Sincronização de Relógios
- Computação Imprecisa
- Engenharia de Software

Multiprocessadores 1/10

- A cada dia produtos com computadores embutidos neles ficam mais complexos, usando mais software e mais hardware
- No passado o poder computacional dos processadores aumentava através do aumento da frequência do clock
- Questões relacionadas com o consumo de energia e a dissipação de calor interromperam esta tendência
- Hoje o poder computacional dos processadores aumenta através do aumento no número de núcleos (multicore)
- Infelizmente, o escalonamento de sistemas de tempo real em multiprocessadores é um problema muito mais difícil do que em monoprocessadores

Multiprocessadores 2/10

- Escalonamento de multiprocessador pode ser visto como um problema duplo:

- O problema de alocação
  - Em qual processador uma tarefa deve executar ?

- O problema de prioridade
  - Em qual ordem as tarefas devem executar em um dado processador ?

Multiprocessadores 3/10

- Sobre alocação:

- *Sem migração*
  - Cada tarefa é alocada em um processador e jamais executa em outro

- *Migração da tarefa como um todo*
  - Os jobs de uma tarefa podem executar em processadores diferentes
  - Entretanto cada job precisa executar inteiro no mesmo processador

- *Migração a qualquer momento*
  - Um mesmo job pode migrar e executar em diferentes processadores
  - Porém sua execução em paralelo não é permitida

Multiprocessadores 4/10

- Algoritmos de escalonamento onde a migração não é permitida são chamados de Particionados
- Algoritmos onde a migração é permitida são chamados de Globais
- A maior parte da pesquisa em escalonamento global foca em modelos onde a migração a qualquer momento é permitida

Multiprocessadores 5/10

- Um algoritmo de escalonamento é dito clarividente
  - Se ele usa informação sobre eventos futuros
  - Tais como a chegada futura de tarefas esporádicas
- Não existe algoritmo dinâmico ótimo para o caso de uma coleção arbitrária de jobs com deadlines diferentes dos períodos
- Tal algoritmo ótimo precisaria conhecer as chegadas futuras e os tempos de execução exatos futuros
- Optimalidade requer clarividência

Multiprocessadores 6/10

- Sistemas particionados são mais simples de implementar e analisar
  - Porém são menos eficientes na ocupação dos processadores
  - Caso uma tarefa seja alocada a um dado processador, ela não poderá executar em nenhum outro processador, mesmo que este outro processador esteja livre
- A análise do tempo de resposta para monoprocessadores está baseada no conceito de instante crítico
- Existem vários tipos de anomalias que tornam a determinação da pior sequência de chegadas um problema combinatório
- Não existe um cenário de chegadas conhecido para gerar o tempo de resposta no pior caso de uma tarefa em sistemas com tarefas esporádicas e escalonamento global

Multiprocessadores 7/10

- É possível comparar dois algoritmos de escalonamento para multiprocessador através da identificação da relação entre eles
- **Dominância** (***dominance***): O algoritmo A domina o algoritmo B se todos os conjuntos de tarefas que são escalonáveis por B são também escalonáveis por A, porém existem conjuntos de tarefas escalonáveis com A que não são escalonáveis com B
- **Equivalência** (***equivalence***): O algoritmo A é equivalente ao algoritmo B se todos os conjuntos de tarefas escalonáveis por B são também escalonáveis por A, e vice-versa
- **Incomparável** (***incomparable***): O algoritmo A é incomparável com o algoritmo B se existem conjuntos de tarefas escalonáveis com A que não são escalonáveis com B e vice-versa

Multiprocessadores 8/10

- É sabido que escalonamento global usando prioridades dinâmicas domina todas as demais classes
- Também é sabido que o emprego de prioridades fixas torna os três tipos de migração incomparáveis entre eles
- E para qualquer tipo de política de prioridades, escalonamento particionado é incomparável com escalonamento onde apenas migração de tarefa é permitida

Multiprocessadores 9/10

- Um importante problema adicional encontrado com multiprocessadores é a interferência indireta sofrida de tarefas que executam em outros processadores

- Em um sistema com múltiplos núcleos de processamento
  o WCET de uma tarefa é fortemente afetado por interferências
  que acontecem entre diferentes núcleos
  através da disputa por recursos de hardware compartilhados
  - memória principal
  - memórias cache
  - barramentos comuns

- Ao ponto de ser impossível determinar o WCET de uma tarefa em uma arquitetura multiprocessadora complexa

- Existem várias propostas na literatura
  - Isolamento temporal
  - Inclusão da interferência indireta na análise da escalonabilidade

Multiprocessadores 10/10

- O escalonamento tempo real em multiprocessadores é um tema complexo
- Existe uma literatura científica gigantesca
- E muitas questões ainda em aberto
- Trata-se ainda de um problema longe de estar resolvido
- Provavelmente necessitaria de um livro inteiro para ser tratado de forma abrangente

Sumário

- Servidores de Tarefas Aperiódicas
- Multiprocessadores
- **Protocolos de Comunicação**
- Protocolos para Sincronização de Relógios
- Computação Imprecisa
- Engenharia de Software

Protocolos de Comunicação 1/11

- Preocupação central é com a divisão entre as tarefas do tempo do processador

- Atualmente um grande número de aplicações de tempo real são na verdade distribuídas
- Colaboração de tarefas que executam em diferentes processadores
- Comunicam-se através de redes de computadores

Protocolos de Comunicação 2/11

- Toda comunicação entre computadores requer algum protocolo
- Um **protocolo de comunicação** (***communication protocol***) define as regras que devem ser seguidas para que a comunicação entre dois computadores ocorra de sucesso
- As pilhas de protocolos mais conhecidas são usadas na Internet
  - protocolo HTTP usado nos browsers
  - protocolo de transporte de dados TCP
  - protocolo IP para conectar computadores de diferentes redes
  - protocolo Ethernet para conexão em rede local

Protocolos de Comunicação 3/11

- Muitas aplicações de tempo real com requisitos temporais rigorosos e críticos
  - aviões, carros, geração de energia elétrica, etc.
- Neste tipo de sistema são usados protocolos de comunicação específicos
  - Apresentam propriedades mais interessantes para o contexto da aplicação
- Protocolos de comunicação usados em um contexto industrial
  - ou dentro de veículos e máquinas
- são genericamente chamados de **barramento de campo** (***fieldbus***)
- O nome vem do fato dos computadores conectados não estarem em um escritório ou residência, mas sim no “campo”
  - o que é “campo” varia de aplicação para aplicação

Protocolos de Comunicação 4/11

- Barramentos de campo diferem dos protocolos da Internet
- Apresentam menor variância nos tempos de comunicação
- Apresentam em geral maior confiabilidade
  - mensagens são perdidas ou descartadas com menor frequência

- Maior robustez eletro-mecânica contra
  - Ruídos eletromagnéticos
  - Vibrações
  - Geração de faíscas em ambientes explosivos
  - etc

Protocolos de Comunicação 5/11

- **CAN (Controller Area Network)**
- Muito usada em veículos automotores e máquinas industriais

- Cada mensagem possui uma prioridade (*CAN network identifier*)
  - Define a ordem de transmissão no barramento
  - De forma equivalente a um escalonamento com prioridades fixas não preemptivas

Protocolos de Comunicação 6/11

- **FlexRay**
- Foi criado especificamente para redes automotivas
  - Determinismo
  - Sincronização
  - Confiabilidade

- Inclui sincronização de relógios
  tolerância a faltas
  mecanismo para controle de acesso ao barramento sem colisões

- Mecanismo baseado no conceito de ciclos de comunicação (*FlexRay Cycle*)
  - com duração fixa

Protocolos de Comunicação 7/11

- **AFDX** (**Avionics Full Duplex Switched Ethernet**)

- Também conhecido como o padrão de comunicação ARINC 664
- Criado especificamente para sistemas aviônicos modernos
- Baseia-se na tecnologia de **ethernet chaveada** (***switched ethernet***)
- Mas impõe rigoroso controle de fluxo

  Para permitir que os tempos de comunicação
  entre os computadores no pior caso
  possam ser determinados em tempo de projeto

Protocolos de Comunicação 8/11

- **Foundation Fieldbus**

- Criado visando aplicação em plantas industriais
  - principalmente na indústria de petróleo e gás

- Preocupação com a geração de faíscas em atmosferas explosivas

- Foundation Fieldbus é usado para conectar
  - sensores (temperatura, pressão, vazão, etc)
  - atuadores (válvulas, bombas, etc)
  - controladores

Protocolos de Comunicação 9/11

- **Profibus**

- Criado para sistemas de automação industrial em geral
  - Produtos químicos
  - Farmacêuticos
  - Papel e celulose
  - Alimentos
  - Energia
  - Água
  - Etc
- Permite a integração de diferentes equipamentos de campo
  - mesmo usando diferentes protocolos a nível físico
- Desde que os mesmos implementem a interface Profibus

Protocolos de Comunicação 10/11

- **TTP (Time-Triggered Protocol)**

- Desenvolvido visando aplicações em veículos e controle industrial
- Controle do acesso ao barramento está fortemente conectado com instantes de tempo para transmissão de cada computador
  - Pré-definidos durante o projeto

- Possui recursos importantes de tolerância a faltas e sincronização de relógios
- Excelente determinismo temporal

Protocolos de Comunicação 11/11

- **Real-Time Ethernet**

- Existem muitos protocolos baseados em ethernet para aplicações de tempo real
  - Inclusive alguns dos listados antes

- Redes ethernet são baratas para instalar e manter
  - Robustas
  - Escaláveis
  - Relativamente simples

Sumário

- Servidores de Tarefas Aperiódicas
- Multiprocessadores
- Protocolos de Comunicação
- **Protocolos para Sincronização de Relógios**
- Computação Imprecisa
- Engenharia de Software

Protocolos para Sincronização de Relógios 1/6

- Relógios em computadores são baseados em cristais de quartzo
- Inevitavelmente apresentam alguma **taxa de deriva** (***drift rate***)
  - Os faz divergirem com o passar do tempo.
- Para que a hora de dois relógios diferentes possa ser comparada, deve existir uma sincronização entre estes relógios
- A sincronização estabelece um limite máximo de erro entre eles
- O que é um limite de erro aceitável
  depende dos requisitos temporais da aplicação em questão

Protocolos para Sincronização de Relógios 2/6

- **Sincronização externa**
- Acontece quando todos os relógios do sistema são sincronizados com uma fonte externa de referência, tipicamente a UTC

- **Sincronização interna**
- Quando os relógios são sincronizados com um grau de precisão conhecido em relação a outro relógio dentro do próprio sistema

- Na ausência de uma conexão direta a um receptor de tempo **GPS** (***Global Positioning System***) ou equivalente:
  é usado algum protocolo de sincronização via rede de comunicação

Protocolos para Sincronização de Relógios 3/6

- No ambiente geral da Internet o protocolo mais usado é o **NTP** (***Network Time Protocol***)

- Precisão na ordem de dezenas de milissegundos na Internet
- Precisão de um milissegundo em redes locais
  - Depende das condições da rede

Protocolos para Sincronização de Relógios 4/6

- Em ambientes industriais:
  - Requisitos temporais mais rigorosos com respeito à sincronização
  - Disponibilidade de redes de comunicação com atrasos mais previsíveis

- Uma solução cada vez mais frequente é utilizar o protocolo **PTP** (***Precision Time Protocol***)

- PTP implementado em software pode atingir precisão da ordem de microssegundos

Protocolos para Sincronização de Relógios 5/6

- Além dos protocolos de propósito geral, como NTP e PTP
- Existem protocolos de sincronização de relógios definidos para um tipo específico de sistema
  - Atrelado a um protocolo de rede industrial (*fieldbus*) específico

- Exemplos desta classe os protocolos encontrados
  no EtherCAT para automação industrial
  no FlexRay para sistemas automotivos

Protocolos para Sincronização de Relógios 6/6

- Todos os protocolos de sincronização fazem basicamente o mesmo

- Obter o tempo de alguma fonte confiável
- Avaliar os atrasos da rede no momento de trocar informações
- Compensar erro absolutos (*offset compensation*)
- Compensar taxas de derivas (*rate compensation*)

- Cada protocolo faz isto de forma diferente
- Redes apresentam diferentes distribuições de atrasos
- A escolha do protocolo de sincronização a ser usado depende
  - Da rede de comunicação usada
  - Do rigor dos requisitos temporais da aplicação em questão

Sumário

- Servidores de Tarefas Aperiódicas
- Multiprocessadores
- Protocolos de Comunicação
- Protocolos para Sincronização de Relógios
- **Computação Imprecisa**
- Engenharia de Software

Cmputação Imprecisa 1/7

- Sistemas em geral:
  "fazer o trabalho usando o tempo que for necessário"
  - Compilador C
  - Folha de pagamentos

- Sistemas de tempo real:
  "garantir que o trabalho será concluído no tempo disponível"
  - O tempo é limitado
  - É preciso garantir que será possível atender aos deadlines

Cmputação Imprecisa 2/7

- Dificuldade encontrada no escalonamento de tempo real: "fazer o trabalho possível dentro do tempo disponível"

- Sacrificar a qualidade dos resultados para cumprir os prazos
- **Computação Imprecisa** (***Imprecise Computation***)
  - Flexibiliza o escalonamento tempo real
  - Pode ser usada para obter certos níveis de **tolerância a faltas** (***fault tolerance***) no sistema
  - Como um mecanismo para tratar de sobrecargas temporárias em sistemas cuja carga é dinâmica

Cmputação Imprecisa 3/7

- Computação Imprecisa fundamentada na idéia de que cada tarefa do sistema possui:

- **Parte obrigatória** (***mandatory part***)
- É capaz de gerar um resultado com a qualidade mínima, necessária para manter o sistema operando de maneira segura
- **Parte opcional** (***optional part***)
- Refina este resultado, até que ele alcance a qualidade desejada

- O resultado da parte obrigatória é dito **impreciso** (***imprecise result***)
- O resultado das partes obrigatória+opcional é dito **preciso** (***precise result***)
- **Tarefa imprecisa** (***imprecise task***): é possível decompô-la em parte obrigatória e parte opcional

Cmputação Imprecisa 4/7

- Situações normais: executa parte obrigatória e parte opcional
- Se, por algum motivo, não for possível executar todas as tarefas do sistema, algumas partes opcionais serão deixadas de lado
  - Permite uma degradação controlada do sistema

- Computação Imprecisa generaliza algoritmos do tipo "**a-qualquer-tempo**" (***anytime algorithms***)
  - Formado por refinamentos iterativos que, a qualquer momento, podem ser interrompidos e a melhor resposta até o momento é fornecida

- Existem 3 formas básicas de programar usando Computação Imprecisa:
  - funções monotônicas
  - funções de melhoramento
  - múltiplas versões

Cmputação Imprecisa 5/7

- **Funções monotônicas** (***monotone functions***)
- São aquelas cuja qualidade do resultado aumenta (ou pelo menos não diminui) na medida em que o tempo de execução de uma função iterativa aumenta

- As computações necessárias para obter-se um nível mínimo de qualidade correspondem à parte obrigatória

- Qualquer computação além desta será incluída como parte opcional

- Algoritmos deste tipo podem ser encontrados nas áreas de cálculo numérico, estimativa probabilista, pesquisa heurística, ordenação e consulta a banco de dados

Cmputação Imprecisa 6/7

- **Funções de melhoramento** (***sieve functions***)
- São aquelas cuja finalidade é produzir saídas no mínimo tão precisas quanto as correspondentes entradas
- O valor de entrada é melhorado de alguma forma

- Se o resultado recebido como entrada por uma função de melhoramento é aceitável como saída, então a função pode ser completamente omitida

- As funções de melhoramento normalmente formam partes opcionais logo após algum cálculo obrigatório
- Por exemplo, etapas no processamento de sinais de radar ou de imagens em geral podem ser puladas, reduzindo a qualidade do resultado e o tempo total de processamento

Cmputação Imprecisa 7/7

- **Múltiplas versões** (***multiple versions***)
- Normalmente são empregadas duas versões
- A versão primária gera um resultado preciso, porém possui um tempo de execução no pior caso desconhecido ou muito grande
- A versão secundária gera um resultado impreciso, porém seguro para o sistema, em um tempo de execução menor e bem conhecido

- A cada ativação da tarefa, cabe ao escalonador escolher qual versão será executada
- Por exemplo, o controle realimentado de um processo pode ser implementado com
  - versão primária baseada em algoritmos mais demorados e complexos (por exemplo controle preditivo)
  - versão secundária baseada em controlador proporcional extremamente rápido, porém com menor qualidade no resultado

Sumário

- Servidores de Tarefas Aperiódicas
- Multiprocessadores
- Protocolos de Comunicação
- Protocolos para Sincronização de Relógios
- Computação Imprecisa
- **Engenharia de Software**

Engenharia de Software 1/5

- Desenvolvimento de sistemas em software apresenta uma enorme lista de outros aspectos que também precisam ser considerados
  - Além de requisitos temporais

- **Engenharia de software** (***software engineering***)
  é normalmente associada com o estudo de conceitos e mecanismos para o planejamento e a gerência do processo de desenvolvimento de software

- Inclui as etapas de
  - Especificação
  - Desenvolvimento
  - Testes
  - Manutenção do software

Engenharia de Software 2/5

- Existe engenharia de software em sistemas de tempo real
- Neste contexto destacam-se diversas
  linguagens de modelagem
  processos de desenvolvimento
  - concebidos para serem aplicados em sistemas de tempo real

- Podemos destacar, entre outras, UML, SYSML e MARTE

Engenharia de Software 3/5

- A linguagem de modelagem **UML** (***Unified Modeling Language***) é amplamente usada para a modelagem visual de software a nível conceitual, de especificação e design
- UML é uma linguagem gráfica
  - Orientada a objetos
  - Propósito geral

- Para sua utilização em sistemas de tempo real, UML provê formas de representar
  - requisitos temporais
  - não determinismo
  - concorrência
- UML fornece maneiras padronizadas para representar questões relacionadas com requisitos temporais e escalonabilidade

Engenharia de Software 4/5

- Linguagem de modelagem **SYSML** (***Systems Modeling Language***)
  projetada para prover construções simples mas poderosas
  para a modelagem de uma grande variedade de problemas na engenharia de sistemas
  - Incluem tipicamente aplicações com requisitos de tempo real

- Reutiliza um subconjunto dos modelos da UML
  e provê construções adicionais

- Particularmente efetiva na especificação de propriedades do sistema para apoiar a parte de análise, tais como
  - Estrutura
  - Comportamento
  - Requisitos
  - Restrições

Engenharia de Software 5/5

- **MARTE**
  (*Modeling and Analysis of Real-time and Embedded Systems*)
- Define como sistemas de tempo real podem ser descritos em termos de modelos
- Os modelos procuram dar o suporte necessário ao desenvolvimento desde a especificação até o design detalhado
- Inclui facilidades para a análise temporal do sistema a partir de seus modelos
- MARTE provê uma forma padronizada de modelar
  - tanto software como hardware
- Visa sistemas embutidos de tempo real
  (*real-time embedded systems*)

Sumário

- Servidores de Tarefas Aperiódicas
- Multiprocessadores
- Protocolos de Comunicação
- Protocolos para Sincronização de Relógios
- Computação Imprecisa
- Engenharia de Software