Rate Monotonic Analysis

Introduction

Periodic tasks

Extending basic theory

Synchronization and priority inversion

Aperiodic servers

Case study: BSY-1 Trainer
Purpose of Tutorial

Introduce rate monotonic analysis

Explain how to perform the analysis

Give some examples of usage

Convince you it is useful
Tutorial Format

Lecture

Group exercises

Case study

Questions welcome anytime
RMARTS Project

Originally called Real-Time Scheduling in Ada Project (RTSIA).

- focused on rate monotonic scheduling theory
- recognized strength of theory was in analysis

Rate Monotonic Analysis for Real-Time Systems (RMARTS)

- focused on analysis supported by (RMS) theory
- analysis of designs regardless of language or scheduling approach used

Project focused initially on uniprocessor systems.
Work continues in distributed processing systems.
Real-Time Systems

Timing requirements
  • meeting deadlines

Periodic and aperiodic tasks

Shared resources

Interrupts
What’s Important in Real-Time

Criteria for real-time systems differ from that for time-sharing systems.

<table>
<thead>
<tr>
<th>Capacity</th>
<th>Time-Sharing Systems</th>
<th>Real-Time Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responsiveness</td>
<td>Fast average response</td>
<td>Ensured worst-case latency</td>
</tr>
<tr>
<td>Overload</td>
<td>Fairness</td>
<td>Stability</td>
</tr>
</tbody>
</table>

- *schedulability* is the ability of tasks to meet all hard deadlines
- *latency* is the worst-case system response time to events
- *stability* in overload means the system meets critical deadlines even if all deadlines cannot be met
Scheduling Policies

CPU scheduling policy: a rule to select task to run next
- cyclic executive
- rate monotonic/deadline monotonic
- earliest deadline first
- least laxity first

Assume preemptive, priority scheduling of tasks
- analyze effects of non-preemption later
Rate Monotonic Scheduling (RMS)

Priorities of periodic tasks are based on their rates: highest rate gets highest priority.

Theoretical basis
- optimal fixed scheduling policy (when deadlines are at end of period)
- analytic formulas to check schedulability

Must distinguish between scheduling and analysis
- rate monotonic scheduling forms the basis for rate monotonic analysis
- however, we consider later how to analyze systems in which rate monotonic scheduling is not used
- any scheduling approach may be used, but all real-time systems should be analyzed for timing
Rate Monotonic Analysis (RMA)

Rate monotonic analysis is a method for analyzing sets of real-time tasks.

Basic theory applies only to independent, periodic tasks, but has been extended to address
  • priority inversion
  • task interactions
  • aperiodic tasks

Focus is on RMA, not RMS.
Why Are Deadlines Missed?

For a given task, consider

- *preemption*: time waiting for higher priority tasks
- *execution*: time to do its own work
- *blocking*: time delayed by lower priority tasks

The task is *schedulable* if the sum of its preemption, execution, and blocking is less than its deadline.

Focus: identify the biggest hits among the three and reduce, as needed, to achieve schedulability
Rate Monotonic Theory - Experience

IBM Systems Integration Division delivered a “schedulable” real-time network.

Theory used successfully to improve performance of IBM BSY-1 Trainer.

Incorporated into IEEE FutureBus+ standard

Adopted by NASA Space Station Program

European Space Agency requires as baseline theory.

Supported in part by Ada vendors
Rate Monotonic Analysis - Products

Journal articles (e.g., *IEEE Computer*, Hot Topics)

Videotape from SEI

Courses from Telos and Tri-Pacific


CASE tools from Introspect and Tri-Pacific

Operating systems and runtimes from Alys, DDC-I, Lynx, Sun, Verdix and Wind River

Standards: Futurebus+, POSIX, Ada 9X
Summary

Real-time goals are: fast response, guaranteed deadlines, and stability in overload.

Any scheduling approach may be used, but all real-time systems should be analyzed for timing.

Rate monotonic analysis

- based on rate monotonic scheduling theory
- analytic formulas to determine schedulability
- framework for reasoning about system timing behavior
- separation of timing and functional concerns

Provides an engineering basis for designing real-time systems
Plan for Tutorial

Present basic theory for periodic task sets

Extend basic theory to include
  • context switch overhead
  • preperiod deadlines
  • interrupts

Consider task interactions:
  • priority inversion
  • synchronization protocols (time allowing)

Extend theory to aperiodic tasks:
  • sporadic servers (time allowing)

Present BSY-1 Trainer case study
A Sample Problem

Periodics

τ₁
20 msec
100 msec

τ₂
40 msec
150 msec

τ₃
100 msec
350 msec

Servers

Data Server

2 msec
20 msec

Comm Server

10 msec
10 msec

Aperiodics

Emergency

50 msec

Deadline 6 msec after arrival

Routine

40 msec

2 msec

Desired response 20 msec average

τ₂’s deadline is 20 msec before the end of each period