Design of Embedded Systems (DES)

Jozef Hooman

http://www.cs.ru.nl/~hooman/DES/

Course 2
10 September 2014

Scheduling (1)
In applications we see many concurrent tasks:
- Printer/copier: control of many motors, paper input, finishing paper, image processing, …
- TV: video & audio processing, get remote control commands, receive teletext, on-screen display, …
- Radio/navigation: process user events, change volume, receive TMC message, database look-up
- X-ray: low-level motor control, supervisory control, collision detection, collision prevention, navigation, planning, user interface, …

Scheduling (2)
Limited hardware; many tasks executed on one processor
Scheduler of OS determines which task is allowed to run
Tasks can be:
- ready to execute
- blocked: cannot continue executing because resource is not available or higher priority task needs to execute
- running: currently executing
- Real-Time OS optimized for short context switches, fast interrupt handling, predictable and precise timing

Scheduling (3)
Consider set of tasks.
Assumptions: [for the moment]
- one processor
- tasks are mutually independent
- tasks are released periodically and worst-case execution time (WCET) is known
- deadline = period
- execution of a task cannot be interrupted (non-preemptive)
[disable 2-clock option in TIMES]

Scheduling (4)

<table>
<thead>
<tr>
<th>Task</th>
<th>Computation time</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>C1 = 1</td>
<td>T1 = 3</td>
</tr>
<tr>
<td>t2</td>
<td>C2 = 2</td>
<td>T2 = 4</td>
</tr>
</tbody>
</table>

- Schedulable? Why?
  - Experiment with TIMES tool
  - Note: WCRT (Worst Case Response Time) is longest time needed to complete
  - Schedulable with C2 = 3?

TIMES: disable 2 clock optimization
Options Configuration Verifier disable: use 2 clocks…

Scheduling (5)

<table>
<thead>
<tr>
<th>Task</th>
<th>Computation time</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>C1 = 1</td>
<td>T1 = 3</td>
</tr>
<tr>
<td>t2</td>
<td>C2 = 2</td>
<td>T2 = 4</td>
</tr>
</tbody>
</table>

Schedulable if there exists schedule in least common multiple of periods
- [For periods 3 and 4, repetition at 12]

Utilization of task ti = Ci/Ti
- [For C2=2: utilization t1: 1/3 ~ 33%, t2: 2/4 ~ 50%]

Total utilization: sum of utilizations
- [For C2=2: 1/3 + 2/4 ~ 83%]
- [For C2=3: 1/3 + 3/4 ~ 108%]
Necessary scheduling condition

Feasible schedule: scheduling strategy for all tasks in which all tasks are guaranteed to meet their deadline

Theorem (necessary condition): For \( n \) tasks,
there exists feasible schedule \( \sum_{j=1}^{n} \frac{C_j}{T_j} \leq 1 \) \((C1)\)

Used in the other direction:
\( \sum_{j=1}^{n} \frac{C_j}{T_j} > 1 \) \( \Rightarrow \) no feasible schedule exists

(C1) necessary but not sufficient

<table>
<thead>
<tr>
<th>Task</th>
<th>Computation time</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>C1 = 3</td>
<td>T1 = 6</td>
</tr>
<tr>
<td>T2</td>
<td>C2 = 1</td>
<td>T2 = 5</td>
</tr>
<tr>
<td>T3</td>
<td>C3 = 5</td>
<td>T3 = 20</td>
</tr>
</tbody>
</table>

Satisfies condition (C1):
\[ \frac{1}{2} + \frac{1}{5} + \frac{5}{20} = 0.5+0.2+0.25 = 0.95 < 1 \]

Schedulable?
\( \Rightarrow \) Try other scheduling policies

Scheduling policies

Round robin: “all are equal”
Rotate between (ready) tasks, usually giving each of them a fixed time slice to execute

Example: scheduling of control tasks at fixed frequency such as motor control at 500 Hz

\[ \quad \]

Round robin scheduling

Very predictable, but also very inflexible
Cannot adapt to changing situations and often inefficient

\( \Rightarrow \) priority based scheduling

Note: priority \( \sim \) number relation differs, e.g.:
- Xenomai & TIMES: larger number is higher prio (in Xenomai lowest is 1, highest 99)
- VxWorks & RTAI: smaller number is higher prio (in VxWorks lowest is 255, highest 0)

Priority-based pre-emptive scheduling

- How to determine priorities?
  E.g. based on deadline, importance, computation time, …?

Most often used:
Rate Monotonic Scheduling (RMS):
task with smallest rate (i.e., shortest period) gets highest priority

Rate Monotonic Analysis (RMA)

Liu & Layland published
"Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment" in JACM in 1973:

Theorem (Rate Monotonic Analysis – RMA):
For \( n \) tasks, if
\[ U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(\sqrt{2} - 1) \]

then RMS is a feasible schedule for these tasks.
Rate Monotonic Analysis (RMA)

<table>
<thead>
<tr>
<th>Task</th>
<th>Computation time</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>C1 = 2</td>
<td>T1 = 10</td>
</tr>
<tr>
<td>t2</td>
<td>C2 = 4</td>
<td>T2 = 15</td>
</tr>
<tr>
<td>t3</td>
<td>C3 = 4</td>
<td>T3 = 20</td>
</tr>
<tr>
<td>t4</td>
<td>C4 = 2</td>
<td>T4 = 25</td>
</tr>
</tbody>
</table>

\[ RMA \]

\[
\lim_{n \to \infty} n (\sqrt{2} - 1) = \ln 2 \approx 0.693147 \ldots
\]

When RMA condition does not hold

Suppose: \( n \cdot (2^{\frac{1}{n}} - 1) < \text{utilization} \leq 1 \)

Optimality RMS

- For fixed priorities, RMS is optimal:
  - if there exists a feasible schedule
  - then RMS leads to feasible schedule

- If there is no feasible schedule with RMS and total utilization \( \leq 1 \), then dynamic priorities can be used

Dynamic priority preemptive scheduling

- Earliest Deadline First (EDF):
  - priorities change with time, at any point in time:
    - Task with earliest deadline gets highest priority
  
  **Theorem (EDF):**
  - Utilization \( \leq 1 \) \( \Rightarrow \) EDF leads to feasible schedule

- No priorities, round robin / time-slices
- Priorities
  - Fixed (static) priorities
  - Non-preemptive
  - Preemptive, e.g. RMS
  - Changeable (dynamic) priorities

Scheduling overview
Variations
• Theory above based on strong assumptions
  (no asynchronous tasks, no deadline > period, no shared resources, …)
• There is a lot of work on variations, see e.g.
  “A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis of
  Real-Time Systems”
  Klein, M., Ralya, T., Pollak, B., Obenza, R.
  Harbour, M. G.,
  Kluwer Academic Publishers 1993

Assignment for 17 Sept 2014
• Xenomai exercise #3
  [and exercise #2 if not finished]
• TIMES exercises 1 and 2
  Deadline: 16 September 18:00
  👉 work in groups of 2
Mail me:
• Description of results of TIMES exercises
• Source files of Xenomai exercise #3 (& Makefile)
  [+ #2] + short description
  ➔ Avoid “zip” extension

Instruction #1
• Exercise #1
  – use the error handling approach also in other examples
    (especially if you have problems …)
  retval = rt_task_inquire(curtask,&curtaskinfo);
  if (retval < 0 )
    rt_printf("Sending error %d : %s
    ",&retval,strerror(-retval));
  – see Tips, also remarks on print

Instruction #2
Notes:
• send also Makefile where all programs are build:
  ### List of applications to be build
  APPLICATIONS = eX02a eX02b eX02c eX02d
  (not Makefile.c)
• Important that parameters have type RTIME
  (unsigned long long)!
• Do not forget pause()
  rt_printf("end program by CTRL-C\n");
  pause();

Instruction #3 - Semphores
Semaphore: control access to shared resource
  RT_SEM access
  rt_sem_create(&access, “Access”, 1, S_FIFO)
  rt_sem_p(&access, …)  rt_sem_p(&access, …)
  <change shared data>  // <change shared data>
  rt_sem_v(&access, …)  rt_sem_v(&access, …)
• 3b: use one or more semaphores; see what you need
• 3c: use semaphores to allow tasks to start execution:
  task blook on rt_sem_p, free with rt_sem_v or
  rt_sem_broadcast