Real-Time Software

Basic Scheduling and Response-Time Analysis

René Rydhof Hansen

21. september 2010

Last Time

- Time in a real-time programming language
  - Access to a clock
  - Delay
  - Timeouts
- Temporal scopes
  - Deadline, minimum delay, maximum delay, maximum execution time, maximum elapse time

Today's Goals

- To understand the simple process model
- To be able to schedule simple systems using the cyclic executive approach
- To understand process-based scheduling
- To be able to perform utilization-based schedulability tests
- To be able to perform response time analysis for FPS
- To understand the concept of WCET and the role it plays
- To understand the role of scheduling and schedulability in ensuring RTSs meet their deadlines

Scheduling

Definition

A mechanism to restrict non-determinism in a concurrent system

Features generally provided

- An algorithm for ordering the use of system resources
    - CPU (most often)
    - Bus-bandwidth
    - Harddisks
    - ...
- Predictable worst case behaviour under the given scheduling algorithm

Standard Notation

- B Worst-case blocking time for the process
- C Worst-case computation time (WCET)
- D Deadline of the process
- I The interference time of the process
- J Release kitter of the process
- N Number of processes in the system
- P Priority assigned to the process
- R Worst-case response time of the process
- T Minimum time between releases (process period)
- U Utilisation of each process (equal to C/T)
- a-z Process name

The Cyclic Executive Approach

- Common way of implementing a hard RTS
- Concurrent design, but sequential code (collection of procedures)
- Procedures are mapped onto a sequence of minor cycles
- Minor cycles constitute the complete schedule: the major cycle
- Minor cycle determines the minimum period
- Major cycle determines the maximum cycle time

Major Advantage

Fully deterministic

Cyclic Executive

Example

Process	Period	Computation Time
a	25	10
b	25	8
c	50	5
d	50	4
e	100	2

```
loop
  wait_for_minor_cycle;
  proc_a; proc_b; proc_c;
  wait_for_minor_cycle;
  proc_a; proc_b; proc_d; proc_e;
  wait_for_minor_cycle;
  proc_a; proc_b; proc_c;
  wait_for_minor_cycle;
  proc_a; proc_b; proc_d;
end loop;
```

Cyclic Executive: Properties

- No actual processes exist at run-time (only procedures)
- Minor cycles are sequences of procedure calls
- Procedures share a common address space
  - Useful for inter-"process" communication
  - Does not need to be protected: concurrent access not possible
- All “process” periods must be a multiple of minor cycle time

Cyclic Executive: Problems

- Difficult to incorporate processes with long periods
  - Major cycle time determines maximum period
  - Can (sometimes) be (partially) solved with secondary scheduling
- Sporadic processes are difficult to incorporate
- Difficult to construct and maintain (NP-hard)
- Time-consuming "processes" must be split
  - Fixed number of fixed sized procedures
  - May cut across useful and well-established boundaries
  - Potentially very bad for software engineering (error prone)
- More flexible scheduling methods are difficult to support
- Determinism is not required but predictability is

Process-Based Scheduling

Approaches

- Fixed-Priority Scheduling (FPS)
- Earliest Deadline First (EDF)
- Value-Based Scheduling (VBS)

The Simple Process Model

- The application has a fixed set of processes
- All processes are periodic with known periods
- The processes are independent of each other
- All processes have deadline equal to their period
- All processes have a fixed worst-case execution time
- All context-switching costs etc. are ignored
- No internal suspension points (e.g., delay or blocking I/O)
- All processes execute on a single CPU

Fixed-Priority Scheduling (FPS)

Definition (FPS)

- Each process has a fixed, static, priority assigned before run-time
- Priority determines execution order

- Most widely used approach
  - Conceptually simple
  - Well-understood
  - Well-supported
- Main focus of the course

Priority $\neq$ Importance

In RTSs the "priority" of a process is derived from its temporal requirements, not its importance to the correct functioning of the system or its integrity

Earliest Deadline First (EDF)

Definition (EDF)

- Execution order is determined by the absolute deadlines
- The next process to run is the one with the shortest (nearest) deadline

EDF with relative deadlines

- Often only relative deadlines are specified
- Absolute deadlines can be computed at run-time (dynamic scheduling)

Value-Based Scheduling (VBS)

Definition (VBS)

- Assign a value to each process
- Use on-line value-based scheduling algorithm
- Basically: schedule process with highest value

- Adaptive schemes necessary for systems that can be overloaded
  - Static priorities and/or deadlines not sufficient
- Easier to factor in widely differing factors
- Easier (conceptually) to handle unforeseen events

Preemption and Non-Preemption

- With priority-based scheduling, a high-priority process may be released during the execution of a lower priority one
- In a preemptive scheme, there will be an immediate switch to the higher-priority process
- With non-preemption, the lower-priority process will be allowed to complete before the high-priority executes
- Preemptive schemes enable higher-priority processes to be more reactive, and hence they are preferred
- Alternative strategies allow a lower priority process to continue to execute for a bounded time
- These schemes are known as deferred preemption or cooperative dispatching
- Schemes such as EDF and VBS can also take on a preemptive or non-preemptive form

Rate Monotonic Priority Assignment (FPS)

- Each process is assigned a (unique) priority based on its period: the shorter the period, the higher the priority: $T_i < T_j \implies P_i > P_j$
- This assignment is optimal in the sense that if any process set can be scheduled (using pre-emptive priority-based scheduling) with a fixed-priority assignment scheme, then the given process set can also be scheduled with a rate monotonic assignment scheme
- Note: priority 1 (one) is the lowest (least) priority

Example (Priority Assignment)

Process	**Period(T)**	**Priority (P)**
a	25	
b	60	
c	42	
d	105	
e	75	

Rate Monotonic Priority Assignment (FPS)

- Each process is assigned a (unique) priority based on its period: the shorter the period, the higher the priority: $T_i < T_j \implies P_i > P_j$
- This assignment is optimal in the sense that if any process set can be scheduled (using pre-emptive priority-based scheduling) with a fixed-priority assignment scheme, then the given process set can also be scheduled with a rate monotonic assignment scheme
- Note: priority 1 (one) is the lowest (least) priority

Example (Priority Assignment)

Process	**Period(T)**	**Priority (P)**
a	25	5
b	60	3
c	42	4
d	105	1
e	75	2

Utilisation-Based Analysis for FPS

- Assume rate monotonic priority assignment
- Sufficient schedulability test for $D = T$ task sets:

$$U \quad \equiv \quad \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N\,(2^{\frac{1}{N}} - 1)$$

- $U \leq 0.69$ as $N \to \infty$

Utilisation bounds

N	Utilisation Bound
1	100.0%
2	82.8%
3	78.0%
4	75.7%
5	74.3%
10	71.8%

Process Set A: Utilisation Based Schedulability Test

Example (Utilisation Test for Process Set A)

Process	Period	Computation Time	Priority	Utilisation
a	50	12	1	0.24
b	40	10	2	0.25
c	30	10	3	0.33

- The combined utilisation is 0.82
- Above threshold for three processes (0.78): process set failed utilisation test

Process Set B: Utilisation Based Schedulability Test

Example (Utilisation Test for Process Set B)

Process	**Period**	**Computation Time**	**Priority**	**Utilisation**
a	80	32	1	0.400
b	40	5	2	0.125
c	16	4	3	0.250

- The combined utilisation is 0.775
- Below threshold for three processes (0.78): utilisation test succeeded (will meet all deadlines)

Process Set C

Example (Utilisation Test for Process Set C)

Process	Period	Computation Time	Priority	Utilisation
a	80	40	1	0.50
b	40	10	2	0.25
c	20	5	3	0.25

- The combined utilisation is 1.0
- Above threshold for three processes (0.78)... but the process set will meet all its deadlines

Utilisation Based Schedulability Test

Sufficient but not necessary

Utilisation-based Tests for FPS: Problems

- Not exact
- Not general (only $T = D$)
- But is $\mathcal{O}(N)$
- The test is sufficient but not necessary

Utilisation-based Test for EDF

A much simpler test

$$\sum_{i=1}^{N} \frac{C_i}{T_i} \leq 1$$

- Superior to FPS; it can support high utilisation
- FPS is easier to implement as priorities are static
- EDF requires more complex run-time system with higher overhead
- Easier to incorporate other factors into a priority than into a deadline
- During overload situations
  - FPS is more predictable; low priority processes miss their deadlines first
  - EDF is unpredictable; domino effect may occur: large number of processes miss deadlines
- Utilisation-based tests: “binary” answer

Response-Time Analysis

Calculating the Slowest Response

- Calculate i's worst-case response time: $R_i = C_i + I$. Where I is the interference from higher priority tasks
- Check (trivially) if deadline is met $R_i \leq D_i$

Calculating I

- During R_i task j (with $P_j > P_i$) is released $\left\lceil \frac{R_i}{T_j} \right\rceil$ number of times.
- Total interference by task j is given by:

$$\left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- The ceiling function, $\lceil x \rceil$: the smallest integer greater than x, e.g., $\lceil 0.25 \rceil = 1$

Response Time Equation

Worst Case Response Time

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

where $hp(i)$ is the set of tasks with priority higher than task i

Solve by forming a recurrence relationship:

$$R_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j$$

The set of values $R_i^0, R_i^1, R_i^2, \ldots, R_i^n, \ldots$ is monotonically non-decreasing. When $R_i^n = R_i^{n+1}$ the solution to the equation has been found, R_i^0, must not be greater than R_i (use e.g., 0 or C_i)

Process Set C: Revisited

Example (Response Time Analysis for Process Set C)

Process	Period	Computation Time	Priority	Response Time
a	80	40	1	80
b	40	10	2	15
c	20	5	3	5

- The combined utilisation is 1.0
- This is above the (utilisation) threshold for three processes (0.78)
- The response time analysis shows that the process set will meet all its deadlines

Response Time Analysis

Necessary and sufficient

Response Time Analysis

- Is sufficient and necessary
- If the process set passes the test, all processes meet all their deadlines
- If the process set fails the test a process will miss its deadline at run-time
  - Modulo wrong estimates, e.g., pessimistic computation time estimate

Worst-Case Execution Time (WCET)

Definition

The maximum amount of execution time a task needs to complete (under all possible circumstances).

- Obtained by either measurement or analysis
- Measurement: hard to guarantee that the worst case has been observed (measured)
  - Never gives too pessimistic results
  - Hard to automate
- Analysis requires effective processor model (including caches, pipelines, memory wait states and other exotic hardware)
  - Bad hardware model may lead to unsound WCET analysis or imprecise (too pessimistic) estimates
  - Can be (partly) automated

Exercises

1. [BW] 11.1
2. [BW] 11.2
3. [BW] 11.3
4. [BW] 11.7
5. [BW] 11.9
6. [BW] 11.10*

Summary

Summary:

- Basic Scheduling: Cyclic executive, FPS, EDF, VBS
- Utilisation analysis for FPS, EDF
- Response time analysis for simple process model