Last version (with booklet format and LaTeX sources)
https://bitbucket.org/OPiMedia/memo-info-f404-real-time-operating-systems

Memo
about INFO-F404

Real-Time Operating Systems

course by Joël Goossens, 2016

Master in Computer science 1
Université Libre de Bruxelles
Computer Science Department

ULB

Notes by
Olivier Pirson
olivier_pirson_opi@yahoo.fr
http://www.opimedia.be/

June 21, 2017

Cover picture by David˜O.

1 Introduction

	$T = D$	C	$load$
τ_1	20	4	20%
τ_2	10	2	20%
τ_3	5	3	60%
			100%

Round Robin failed
RM success

2 Uniprocessor real-time scheduling

Correctness: logical result of computations *and* time of the result
- Hard-real time: catastrophic consequences
- Firm real-time: some deadline may be missed
- Soft real-time: deadline missed not important

Job $J_i = (a, e, d)$
- a: relase time
- e: computer requirement (#CPU unit must be received in $[a, d)$)
- d: absolute deadline

Periodic task $\tau_i = (O_i, T_i, D_i, C_i)$
- O_i offset: release time of the 1st job
- T_i: period between 2 consecutive task releases
- D_i relative deadline: time delay between time release and absolute deadline
- C_i computation time: worst-case execution requirement

Induce infinite sequence of jobs: $J_{i,1}, J_{i,2}, J_{i,3}, \ldots, J_{i,k}, \ldots$
$a = O_i + (k-1)T_i \qquad e = C_i \qquad d = a + D_i$

Sporadic task: T is the *minimum* between 2 consecutive task release

Periodic/sporadic system: finite set of periodic/sporadic tasks $\tau = \{\tau_1, \ldots, \tau_n\}$

Task utilization $U(\tau_i) := \frac{C_i}{T_i}$ System utilization $U(\tau) := \sum_{\tau_i \in \tau} U(\tau_i)$

- Implicit deadline: $\forall i : D_i = T_i$
- Constrained deadline: $\forall i : D_i \leq T_i$
- Abritrary deadline
- Synchronous: $\forall i : O_i = 0$
- Asynchronous: not synchronous

- Scheduling discipline: algo. distribute ressources
- Preemptive scheduler: capable interrupts process to allocate another

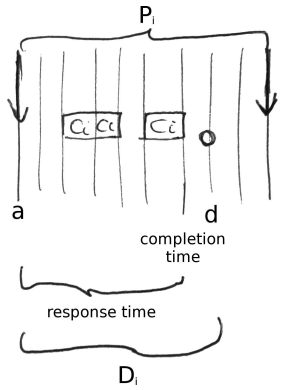
- FTP Fixed Task Priority: offline, one priority by task
- FJP Fixed Job Priority: online, one priority by job
- DP Unrestricted Dynamic Priority: priority may be change during lifetime of job

Assumptions on this chapter:
- uniprocessor

Contents

List of Theorems

- hard real-time
- preemtible tasks/jobs
- work-conserving: CPU cannot be idle if exist active jobs
- tasks are independent (only CPU is shared)



Job response time = completion time - release time

Lemma 6. $U(\tau) \leq 1 \quad \Longleftarrow \quad \tau$ schedulable

2.1 FTP

WLOG: $\tau_1 \succ \tau_2 \succ \tau_3 \succ \cdots \succ \tau_n$ ($\simeq$sort, $O(n \log n)$)

2.1.1 RM Rate Monotonic: lower period, higher priority

For synchronous, *implicit* deadline

- System FTP-feasible: if exists schedulable FTP-priority assignment
- Assignment FTP-optimal: if schedules *all* FTP-feasible system

Critical instant of τ_i: instant s.t. job release at this instant has maximum response time of all jobs in τ_i

Lemma 10 (Liu, 2000)**.**
$J_{i,k}$ release at same time t job of every higher priority task
$\Longleftarrow$ t is critical instant of τ_i

Lemma 11. τ periodic, S_1, S_2 schedules of work-conserving schedulers for τ
$\forall t:$ S_1 is idle at $t \iff S_2$ is idle at t

Thm 12. synchronous, *implicit* deadline
RM is an FTP-optimal priority assignment

r_i^1: response time of the 1st request of τ_i
Thm 13. synchronous, *constrained* deadline
FTP-schedulable $\iff$ $\forall i : r_i^1 \leq D_i$

Audsley & Tindell, 1991: $r_i^1 = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_i^1}{T_j} \right\rceil C_j$

Fixed point iteration (until $w_{k+1} = w_k$ or $w_k > D_i$): $w_0 := C_i$

$$w_{k+1} := C_i + \sum_{j=1}^{i-1} \left\lceil \frac{w_k}{T_j} \right\rceil C_j$$

Thm 14 (Leung, 2004)**.** periodic, synchronous, implicit deadline
$U(\tau) \le n(\sqrt[n]{2}-1) \implies \tau$ feasible with RM

Corollary 17. periodic, synchronous, implicit deadline
$U(\tau) \le \ln 2 = 0.693\ldots \implies \tau$ feasible with RM

	$T_i = D_i$	C_i
τ_1	10	4
τ_2	15	3
τ_3	20	7

not schedulable with RM

2.1.2 DM Deadline Monotonic: lower relative deadline, higher priority

For synchronous, *constrained* deadline (RM $\subset$ DM)

Thm 16. synchronous, *constrained* deadline
DM is an FTP-optimal priority assignment

	T_i	D_i	C_i
τ_1	100	110	52
τ_2	140	154	52

RM: $\tau_1 \succ \tau_2$ failed
DM: $\tau_1 \succ \tau_2$ failed
but $\tau_2 \succ \tau_1$ success

$\implies$ RM, DM are not optimal for arbitrary deadline

Feasibility interval:
finite interval s.t. if no deadline missing during this interval $\Rightarrow$ system feasible

Thm 18. synchronous, *constrained* deadline
$[0, \max_i D_i)$ is a feasibility interval

$x \in \mathbb{N}$ is an idle point if all requests occuring $< x$ are completed $\le x$
CPU is idle at $x \implies x$ is an idle point

$[a, b)$ is an elementary busy period if a, b are idle points; idle point $\notin (a, b)$

Level-i busy period: elementary busy period for $\{\tau_1, \tau_2, \ldots, \tau_i\}$; τ_i executes at least 1 job

Thm 22. synchronous:
largest response time for a request of $\tau_i \in$ *first* level-i busy period $[0, \lambda_i)$
λ_i is the first idle point > 0 for $\{\tau_1, \tau_2, \ldots, \tau_i\}$
$[0, \lambda_i)$ is the largest level-i busy period

Index

Thm 23. synchronous, *arbitrary* deadline:
$[0, \lambda_n)$ is a feasability interval

λ_n is the smallest solution of $\lambda_n = \sum_{i=1}^{n} \left\lceil \frac{\lambda_n}{T_i} \right\rceil C_i$

Fixed point iteration: $w_0 := \sum_{i=1}^{n} C_i$

$$w_{k+1} := \sum_{i=1}^{n} \left\lceil \frac{w_k}{T_i} \right\rceil C_i$$

2.1.3 Asynchronous

Hyper-period $P := \text{lcm}_i\, T_i$
$O_{max} := \max_i O_i$

Thm 24 (Leung & Whitehead, 1982)**.** asynchronous, *constrained* deadline:
$[O_{max}, O_{max} + 2P)$ is a feasibility interval

Thm 25. periodic task:
feasibility schedules are periodic

Thm 26. asynchronous, *constrained* deadline, $\tau_1 \succ \tau_2 \succ \cdots \succ \tau_n$
$S_1 := O_1$
$S_i := O_i + \left\lceil \frac{(S_{i-1} - O_i)^+}{T_i} \right\rceil T_i$ $\qquad x^+ := \max\{0, x\}$
Schedule is feasible up to $S_n + P$ $\implies$ | is feasible
| periodic from S_n
| period P

$\implies$ $[0, S_n + P)$ is a feasibility interval

	O_i	$T_i = D_i$	C_i
τ_1	10	12	1
τ_2	0	12	6
τ_3	0	8	3

RM, DM: $\tau_3 \succ \tau_1 \succ \tau_2$ failed
but $\tau_3 \succ \tau_2 \succ \tau_1$ success

$\implies$ RM / DM are not optimal for asynchronous

2.1.4 Audsley algorithm

τ_i is lowest-priority viable $\iff$ all jobs meet their deadline when
| τ_i has the lowest priority
| other tasks have any higher priority
| when scheduling $\tau_j \neq \tau_i$ consider deadline softs

Thm 28. τ_i lowest-priority viable:
$\exists$ feasible FTP-assignment for τ $\iff$ $\exists$ feasible FTP-assignment for $\tau \setminus \{\tau_i\}$

Audsley algorithm

```
procedure Audsley(τ) {
  if (there is no lowest-priority viable task) {
    return infeasible
  }
  else {
    τ_i := a lowest-priority viable task in τ
    assign lowest priority to τ_i
    Audsley(τ \ {τ_i})
  }
}
```

Checking if a task is lower-priority viable on $[0, S_n + P_n)$
$O(n^2)$ distinct priority assignments $\ll O(n!)$

2.2 FJP

Earliest Deadline First EDF: lower *absolute* deadline, higher priority

Job set J is feasible if $\exists$ schedule which meets all the job deadlines

Thm 30. Job set J is feasible $\implies$ EDF feasibly schedule J
EDF is | FJP-optimal
FTP-optimal for | asynchronous
abritrary deadline

Thm 31. Periodic, implicit deadline:
$U(\tau) \leq 1 \iff \tau$ schedulable

Corollary 32. Periodic, implicit deadline:
$U(\tau) \leq 1 \iff \tau$ is EDF-schedulable

Thm 33.
τ | synchronous
periodic
arbitrary deadline
EDF-feasible
$\implies$ | $\forall$ corresponding | asynchronous
periodic | : EDF-feasible
$\forall$ corresponding sporadic: EDF-feasible

	$T_i = D_i$	C_i
τ_1	4	2
τ_2	7	3

response time: $5, 5, 5, \mathbf{6}$ (the largest is not the first)

Thm 34 (Liu, Leyland, 1973)**.** periodic, synchronous, constrained deadline:
deadline miss in $t \implies \nexists$ idle $< t$

References

[INFO-F404 2016] Joël Goossens.
Course INFO-F404 *Real-Time Operating Systems*.
Université Libre de Bruxelles, Computer Science Department, 2016

5 Parallel algorithms

t_s: execution time of uniprocesor (sequential) version
t_t: execution time of multiprocesor (parallel) version with n processors
Speedup factor $S(n) := \frac{t_s}{t_p}$ (relative performance difference)

Linear speedup principle:
maximum speedup factor that can be gained using n processors is n

Law 1 (Amdahl's law)**.**
Let f the fraction of sequential computations of a parallel program.
$(1-f)$ is the fraction of parallelisable computations.
$S(n) = \frac{t_s}{f t_s + (1-f)\frac{t_s}{n}} = \frac{n}{1+(n-1)f}$

Upper bound of speedup factor (even with unlimitied number of processors) :
$\lim_{n\to\infty} S(n) = \frac{1}{f}$
(If only 5% of the computations are sequential, the *potential* maximum speedup factor is 20, regardless of the number of processors.)

...

Thm 35 (Goosens, 1999)**.** periodic, synchronous, constrained deadline:
deadline miss in $t \implies \nexists$ (except 0) idle point $< t$
$\implies [0, L)$ is a feasibility interval L is the first idle point (except 0)

$L\ (= \lambda_n)$ = smallest solution of $L = \sum_{i=1}^{n} \left\lceil \frac{L}{T_i} \right\rceil C_i$

Periodic system R, schedule S:
Configuration $C_S(R,t) := \Big((\gamma_1(t), \alpha_1(t), \beta_1(t)), \ldots, (\gamma_n(t), \alpha_n(t), \beta_n(t)) \Big)$

where
$\gamma_i(t) := \begin{cases} (t - O_i) \% T_i & \text{if } t \geq O_i \\ t - O_i & \text{if } t < O_i \end{cases}$
$\alpha_i(t) :=$ # active jobs of τ_i
$\beta_i(t) :=$ cumulative CPU time used by oldest active job of τ_i

Thm 37. R periodic, asynchronous, arbitrary deadline, S = EDF:
R is feasible $\iff$ all deadline $\in [0, O_{max} + 2P)$ met
$C_S(R, O_{max} + P) = C_S(R, O_{max} + 2P)$

	O_i	T_i	D_i	C_i	U
τ_1	0	4	4	2	1/2
τ_2	2	4	7	3	3/4
					5/4

all deadline $[0, 10)$ met
but $\beta_2(6) = 2 \neq \beta_2(10) = 1$ and τ_2 misses at $t = 21$

Corollary 38. periodic, asynchronous, arbitrary deadline:
$U(\tau) \leq 1 \implies [0, O_{max} + 2P)$ is a feasibility interval

2.3 DP

Job $J = (a, e, d)$
$\epsilon_J(t) :=$ cumulative CPU time used by J since a.
laxity $l_J(t) := d - t - \big(e - \epsilon_J(t)\big)$
> 0: J can wait $l_J(t)$ time-units
$= 0$: J must run immediately and until d
< 0: J will miss

J running $\implies l_J(t+1) = l_J(t)$

LLF Least Laxity First

Thm 41 (Mok, 1983)**.** LLF is DM-optimal

	T_i	D_i	C_i
τ_1	10	8	4
τ_2	10	9	5

EDF: preemptive free
LLF: 6 peemptions by 10 time-units

Trashing: large amount of resources are used to do a minimal work

$l_{J_1}(t) = l_{J_2}(t)$
run J_1
$\Longrightarrow \quad l_{J_1}(t) = l_{J_2}(t) + 1$

$\exists$ FJP-optimal
LLF cause lot of preemptions | so LLF useless

4 Concurrency

...

3.2.2 EDF$^{(k)}$

Assume $U(\tau_1) \geq U(\tau_2) \geq \cdots \geq U(\tau_n)$
$\tau^{(i)} := \{\tau_i, \tau_{i+1}, \ldots, \tau_n\}$
EDF$^{(k)}$: $\forall i < k$: set maximal priority to τ_i ($d := -\infty$)
$\forall i \geq k$: set EDF priority to τ_i

EDF = EDF$^{(1)}$

Thm 20 (Goossens, Funk, & Baruah, 2003)**.** Sporadic, implicit deadline:
$m = (k-1) + \left\lceil \frac{U(\tau^{(k+1)})}{1-U(\tau_k)} \right\rceil \implies$ EDF$^{(k)}$ schedulable

Corollary 21. Sporadic, implicit deadline
is schedulable with $m_{min}(\tau) = \min_{k=1}^{n} \left\{ (k-1) + \left\lceil \frac{U(\tau^{(k+1)})}{1-U(\tau_k)} \right\rceil \right\}$ processors.
$k_{min} := k$ that minimize

3 Multiprocessor real-time scheduling

Consider only limited parallel architecture. At each instant:
- one processor executes one task at most
- one task is execute on one processor at most (no task or job parallelism)
- strongly coupled systems (common time reference and shared memory)
- identical parallel machines: $\pi_1, \pi_2, \ldots, \pi_m$

- Identical parallel machines: all processors are identical (same computing power)
- Uniform parallel machines: computing capacity s_i for each processor π_i
- Unrelated parallel machines: execution rate $s_{i,j}$ for each job-processor (J_i, π_j)

Identical $\subset$ Uniform $\subset$ Unrelated

- Partitioned scheduling: divide system in subsystems associate to 1 processor
- Global scheduling: jobs can to migrate during their lifetime

Lemma 1 (Baruah, 2007)**.** Exists systems schedulable with global FJP algorithms but not partitioned.

	T_i	D_i	C_i
τ_1	3	2	2
τ_2	4	3	3
τ_3	12	12	5

With 2 processors:
all partitioned FJP failed
global FJP with $\succ \tau_3$ success

Lemma 2 (Baruah, 2007)**.** Exists systems schedulable with partitioned FJP algorithms but not global.

	T_i	D_i	C_i
τ_1	3	2	2
τ_2	4	3	3
τ_3	12	12	4
τ_4	12	12	3

With 2 processors:
all global FJP failed
partitioned $\{\tau_1, \tau_3\}, \{\tau_2, \tau_4\}$ FJP success

	T_i	D_i	C_i
τ_1	4	2	1
τ_2	5	3	3
τ_3	2	3	7

Density $\lambda_i := \frac{C_i}{\min(T_i, D_i)}$
Total density $\lambda_{sum} := \sum_{\tau_i \in \tau} \lambda(\tau_i)$ Maximal density $\lambda_{max} := \max_{\tau_i \in \tau} \lambda(\tau_i)$

sporadic tasks, m identical processors:
$\left.\begin{array}{l} \lambda_{sum}(\tau) \leq m \\ \lambda_{max}(\tau) \leq 1 \end{array}\right| \impliedby \tau$ schedulable

3.1 Partitioned scheduling

For *implicit* deadline

Bin-Packing problem: k objects of different sizes in a minimum of bins.
NP-complete, so heuristics.
Greedy algorithm, with optional pre-processing (sort by DU decreasing utilization or IU increasing utilization):
- FF First-fit
- Best-fit
- Worst-fit
- Next-fit

RM: $U(\tau) \leq n_\pi (\sqrt[n_\pi]{2} - 1)$ $\qquad$ n_π := # tasks on processor π

Thm 5. Sporadic, implicit deadline:
$\left.\begin{array}{l} U(\tau) \leq \frac{m+1}{2} \\ U_{max} \leq 1 \end{array}\right| \Longrightarrow$ τ schedulable with FFDU (with EDF local scheduler)

3.2 Global scheduling

Online scheduler: take scheduling decisions at *runtime* with jobs already realeased and *without* knowledge of future

Thm 7 (Hong & Leung, 1988).
$m > 1 \implies \nexists$ online and optimal scheduling

	$T_i = D_i$	C_i
τ_1	2	4
τ_2	2	4
τ_3	4	8

Intuitively positive change: decrease U (increase T_i, decrease $D_i \dots$)
Anomaly if schedulable system become unschedulable with intuitively positive change

Sporadic: no sufficient to check synchronous periodic sub-case (critical instant is an open question)

A schedulability test is sustainable if schedulable system remains schedulable with theses possible changes:
- decreased C_i (Sustainable with respect to the execution requirements)
- later arrival times

- larger D_i

Thm 12 (Cucu-Grosjean & Goossens, 2010). Unrelated platforms:
FJP schedulers are sustainable with respect to the execution requirements

Thm 13. Periodic, implicit deadline:
$\left.\begin{array}{l} U(\tau) \leq m \\ U_{max}(\tau) \leq 1 \end{array}\right| \Longleftrightarrow$ τ feasible

3.2.1 PFAIR scheduling

Time slice or quantum: period of time during a process is allowed to run

$$\text{Schedule } S: \begin{array}{ccc} \tau \times \mathbb{N} & \to & \{0,1\} \\ (\tau_i, t) & \mapsto & \begin{cases} 1 & \text{if } \tau_i \text{ scheduled in slice } [t, t+1) \\ 0 & \text{otherwise} \end{cases} \end{array}$$

For ideal fair schedule each τ_i received $U(\tau_i).t$ slices during $[0, t)$ (implies all deadline are met)

$$lag(\tau_i, t) := U(\tau_i).t - \sum_{l=0}^{t-1} S(\tau_i, l)$$

Schedule has PFAIR property $\overset{\triangle}{\Longleftrightarrow} -1 < lag(\tau_i, t) < 1, \forall \tau_i \in \tau, \forall t \in \mathbb{N}$
$\Longleftrightarrow \forall \tau_i \in \tau : \tau_i$ receive $\lfloor U(\tau_i).t \rfloor$ or $\lceil U(\tau_i).t \rceil$ in $[0, t)$

Thm 15. PFAIRness $\implies$ all deadline are met

Thm 16 (Baruah, Cohen, Plaxton & Varvel, 1996).
Periodic, synchronous, implicit deadline:
$\left.\begin{array}{l} U(\tau) \leq m \\ U_{max}(\tau) \leq 1 \end{array}\right| \Longleftrightarrow \exists$ PFAIR schedule
$\Rightarrow$ PFAIR is optimal

Thm 17. Sporadic, arbitrary deadline:
$\left.\begin{array}{l} \lambda_{sum}(\tau) \leq m \\ \lambda_{max}(\tau) \leq 1 \end{array}\right| \implies$ feasible

Thm 18 (Srinivasan & Baruah, 2002). Sporadic, implicit deadline:
$U(\tau) \leq m - (m-1)U_{max}(\tau) \implies$ schedulable with global EDF