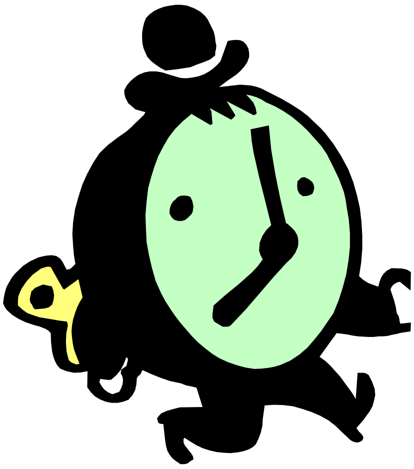
Real-Time Systems Programming

Summer-Semester 2002
Lecture 23
4 July 2002



Fixed Priority Scheduling

The 5 Minute Review Session

1) What are the standard *coordination* mechanisms in POSIX?
2) What are the standard *scheduling* mechanisms in POSIX?
3) What are the assumptions of the simple process model?
4) How do you assess the *cyclic executive* approach for scheduling?
5) What are other approaches to scheduling?

Overview

- Rate-Monotonic priority assignment
- Utilization-based schedulability analysis
- Response time analysis
- Sporadic and aperiodic processes
- Deadline-Monotonic scheduling

Fixed Priority Scheduling and Rate Monotonic Priority Assignment

- Each process is assigned a (unique) priority based on its period
  - The *shorter* the period, the *higher* the priority
  - For two processes *i* and *j*: $T_i < T_j \Leftrightarrow P_j < P_i$
- This assignment is ***optimal:***
  - ***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

Example Priority Assignment

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

- ***Note:*** Priority 1 is the lowest (least) priority
- So far, we assume that the deadline of each process is equal to its period (D = T)
- The schedulability of a process set depends on the ***period*** and the ***maximal computational requirements*** of each process

Utilization-Based Analysis

- For task sets with D=T, a simple ***sufficient but not necessary*** schedulability test exists
- If the following holds, then all *N* processes will meet their deadlines:

$$\sum_{i=1}^{N} \left(\frac{C_i}{T_i} \right) \leq N(2^{1/N} - 1)$$

Asymptotically approaches ln 2 (69.3%)

N	Utilization bound
1	100%
2	82,8%
3	78,0%
4	75,7%
5	74,3%
6	71,8%

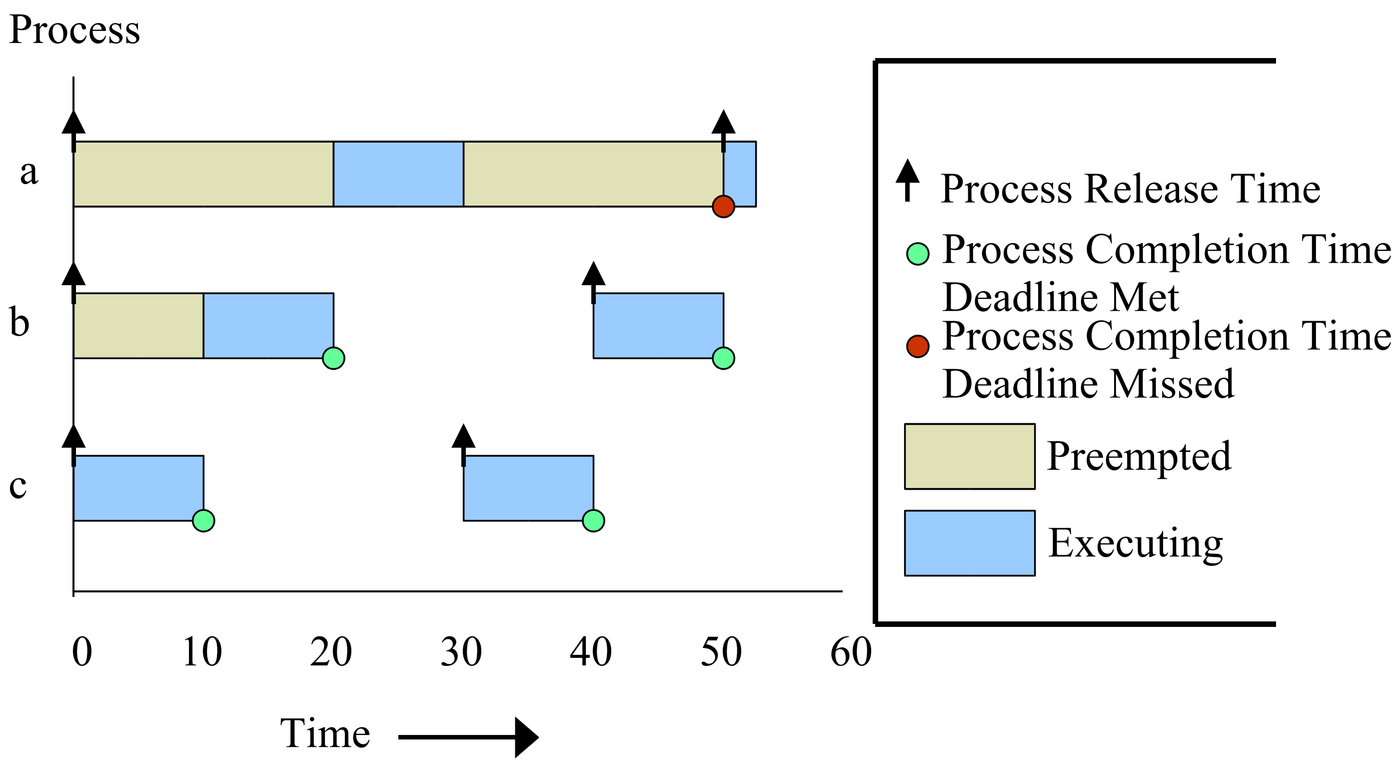
See also C. Liu and J. Layland (1973) – *one of the seminal computer science papers!*

Example: Process Set A

Process	Period (T)	ComputationTime (C)	Priority (P)	Utilization (U)
a	50	12	1	0.240
b	40	10	2	0.250
c	30	10	3	0.333

- The combined utilization is 0.823 (or 82.3%)
- This is above the threshold for three processes (0.78) and, hence, this process set fails the utilization test

Time-line for Process Set A



Note that at time 50, process *a* has consumed only 10 ticks of execution, whereas it needed 12, and hence missed its first deadline

Process Set B

Process	Period (T)	ComputationTime (C)	Priority (P)	Utilization (U)
a	80	32	1	0.400
b	40	5	2	0.125
c	16	4	3	0.250

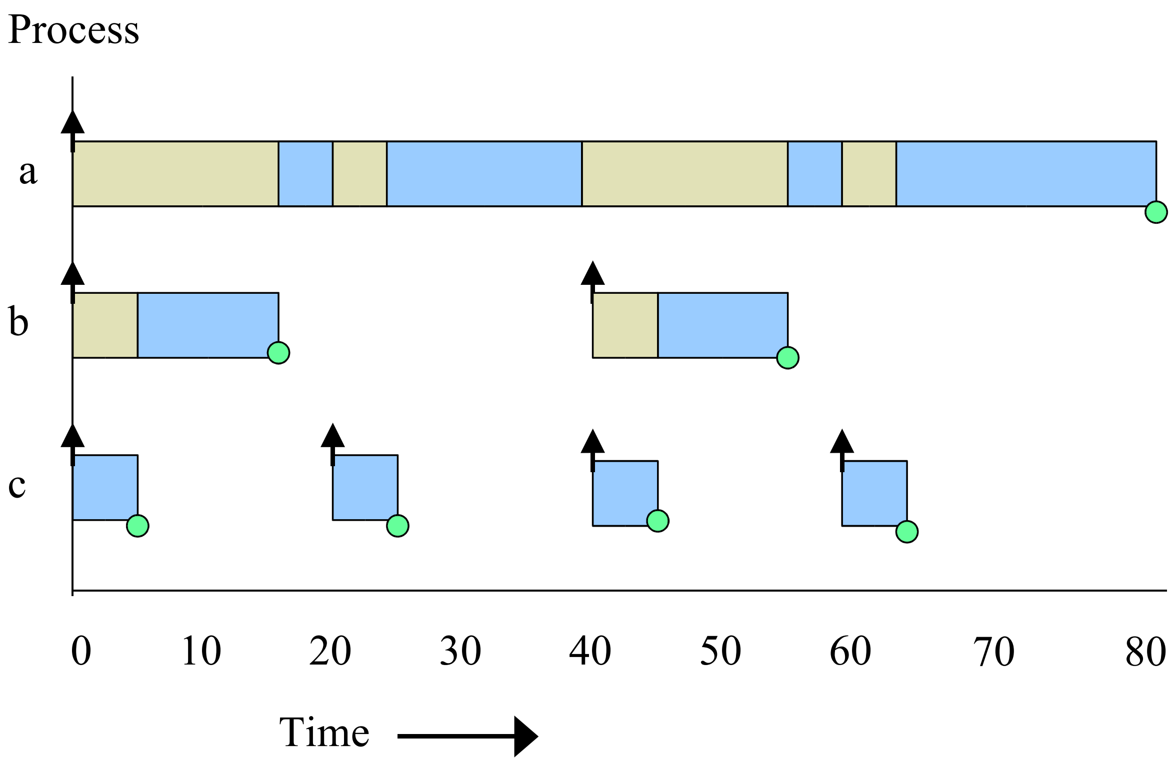
- The combined utilization is 0.775
- This is below the threshold for three processes (0.78) and, hence, this process set will meet all its deadlines

Process Set C

Process	Period (T)	ComputationTime (C)	Priority (P)	Utilization (U)
a	80	40	1	0.50
b	40	10	2	0.25
c	20	5	3	0.250

- The combined utilization is 1.0
- This is above the threshold for three processes (0.78) ***but the process set will meet all its deadlines***

Time-line for Process Set C



Utilization-based Test for EDF

- Similar to the utilization-based schedulability test for FPS, there is also a – even simpler – test for EDF:

$$\sum_{i=1}^{N} \left(\frac{C_i}{T_i} \right) \leq 1$$

- EDF is superior to FPS in that it can support high utilizations
- However, there are also weaknesses of EDF compared to FPS ...

Weaknesses of EDF vs. FPS

- FPS is ***easier to implement*** as priorities are static
  - EDF is dynamic and requires a more complex run-time system which will have higher overhead
- It is easier to incorporate processes without deadlines into FPS
  - Giving a process an arbitrary deadline is more artificial
- It is easier to incorporate other factors into the notion of priority than it is into the notion of deadline
- During overload situations
  - FPS is ***more predictable***; Low priority process miss their deadlines first
  - EDF is ***unpredictable***; a domino effect can occur in which a large number of processes miss deadlines

Response-Time Analysis for FPS

- Two drawbacks of the utilization-based tests for FPS:
  - Not exact
  - Not applicable to a more general process model
- An alternative to utilization analysis is ***response time analysis*** – for each process i:
  - 1. Compute the ***worst-case response time***, $\boldsymbol{R_i}$
  - 2. Check whether the process meets its deadline, $R_i \leq D_i$
- For the highest priority process, it is $R = C$
- Other processes may suffer ***interference*** from higher-priority tasks:

(1)
$$\boxed{R_i = C_i + \boldsymbol{I_i}}$$

Bounding the Interference

- I_i is the maximum interference that process i can experience from higher-priority processes within the time interval $[t, t + R_i)$
- To determine a bound on I_i, assume that all processes are released at once; wlog, assume this is at time 0
- Consider process j with a higher priority than process i
- The number of times process j is released is bounded by

(2) $$NumberOfReleases_{i,j} = \left\lceil \frac{R_i}{T_j} \right\rceil$$

- ***Example:*** if $R_i = 15$, $T_j = 6$, process j is released at times 0, 6, 12

Bounding the Interference

- Each release of process j will impose an interference of C_j
- Hence the maximum interference is imposed by process j on process i is given by

(3) $$MaxInterference_{i,j} = \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- ***Example:*** if $R_i = 15$, $T_j = 6$, and $C_j = 2$, then process j imposes an interference of 6 time units

Bounding the Interference

- Each process of higher priority than process i interferes with i; hence:

(4)
$$I_i = \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- Substituting (4) back into (1) yields

(5)
$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Response Time Equation

- Equation (5) can be solved by forming a recurrence relationship

(6)
$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$

- Can initialize $w_i^0 = C_i$
- The set of values w_i^0, w_i^1, ... is monotonically non-decreasing
- When $w_i^n = w_i^{n+1}$, then the solution to the equation has been found
- If w_i^n exceeds T_i, process will not meet its deadline

Response Time Algorithm

```
// For each process in turn
for i in 1..N loop
  n := 0
  w[i,0] = C[i]

  loop
    // calculate new w[i,n+1] from Equation (5)
    ...
    if w[i,n+1] = w[i,n] then
      value_found = true
      exit
    else if w[i,n+1] > T[i] then
        value_found = false
        exit
    end if
    n := n + 1
  end loop
end loop
```

Process Set D

Process	Period (T)	Computation Time (C)	Priority (P)
a	7	3	3
b	12	3	2
c	20	5	1

- Process *a*, which has the highest priority, has a response time equal to its computation time: $R_a = 3$
- Applying the recurrence relation (6) to process *b* results in the following series:

$$w_b^0 = 3, w_b^1 = 3 + \left\lceil \frac{3}{7} \right\rceil 3 = 6, w_b^2 = 3 + \left\lceil \frac{6}{7} \right\rceil 3 = 6 = w_b^1$$

- Hence the response time for process *b* is $R_b = 6$

Process Set D

- The response time for the final process, c, is calculated as follows:

$$w_c^0 = 5$$

$$w_c^1 = 5 + \left\lceil \frac{5}{7} \right\rceil 3 + \left\lceil \frac{5}{12} \right\rceil 3 = 11$$

$$w_c^2 = 5 + \left\lceil \frac{11}{7} \right\rceil 3 + \left\lceil \frac{11}{12} \right\rceil 3 = 14$$

$$w_c^3 = 5 + \left\lceil \frac{14}{7} \right\rceil 3 + \left\lceil \frac{14}{12} \right\rceil 3 = 17$$

$$w_c^4 = 5 + \left\lceil \frac{17}{7} \right\rceil 3 + \left\lceil \frac{17}{12} \right\rceil 3 = 20$$

$$w_c^5 = 5 + \left\lceil \frac{20}{7} \right\rceil 3 + \left\lceil \frac{20}{12} \right\rceil 3 = 20$$

$$w_c^5 = w_c^4 = R_c = 20$$

Revisit: Process Set C

Process	Period (T)	ComputationTime (C)	Priority (P)	Response Time (R)
a	80	40	1	80
b	40	10	2	15
c	20	5	3	5

- The combined utilization is 1.0
- This was above the ulilization threshold for three processes (0.78), therefore it failed the test
- ***However,*** the response time analysis shows that the process set will meet all its deadlines

Assessment Response Time Analysis

- RTA is ***necessary*** and ***sufficient***
- If the process set passes the test they will meet all their deadlines
- If they fail the test then, at run-time, a process will miss its deadline – unless the computation time estimations themselves turn out to be pessimistic

Sporadic Processes

- Sporadic processes have a minimum inter-arrival time T
- For periodic processes, we assumed $\boldsymbol{D = T}$
- For sporadic processes, deadlines are typically shorter: $\boldsymbol{D < T}$
  - For example, if the sporadic process is some error handling routine, which is rarely triggered, but must be executed quickly
- The response time analysis algorithm for fixed priority scheduling based on equation (5) works applies here as well

Hard and Soft Processes

- In many situations the worst-case figures for sporadic processes are considerably higher than the averages
- Interrupts often arrive in bursts and an abnormal sensor reading may lead to significant additional computation
- Measuring schedulability with worst-case figures may lead to ***very low processor utilizations*** being observed in the actual running system

General Guidelines

Rule 1 — all processes should be schedulable using average execution times and average arrival rates

- There may be ***transient overloads***, in which it is not possible to meet all current deadlines

Rule 2 — all ***hard real-time*** processes should be schedulable using worst-case execution times and worst-case arrival rates of all processes (including soft)

- No hard real-time process will miss its deadline
- If this gives rise to unacceptably low utilizations for "normal execution" then action must be taken to reduce the worst-case execution times (or arrival rates)

Aperiodic Processes

- Do ***not*** have minimum inter-arrival times
- Can run aperiodic processes at a priority ***below*** the priorities assigned to hard processes
  - Therefore, they cannot steal, in a pre-emptive system, resources from the hard processes
- ***Problem:*** soft processes will often miss deadlines
- ***Improvement: Server***
  - Protects processing resources needed by hard processes
  - But allows soft processes to run as soon as possible
- ***POSIX*** supports Sporadic Servers

Process Sets with D < T

- For D = T, ***Rate Monotonic*** priority ordering is optimal
- For D < T, ***Deadline Monotonic*** priority ordering is optimal
  - Here the priority of a process is inversely proportional to its deadline

Process	**Period (T)**	**Deadline (D)**	**Computation Time (C)**	**Priority (P)**	**Response Time (R)**
a	20	5	3	4	3
b	15	7	3	3	6
c	10	10	4	2	10
d	20	20	3	1	20

Optimality of DMPO

- Deadline monotonic priority ordering (DMPO) is optimal:
  - ***If a process set `Q` is schedulable by some priority scheme `W`, then `Q` is also schedulable by DMPO***
- Can prove this by transforming the priorities of `Q` (as assigned by `W`) until the ordering is DMPO
  - Each step of the transformation will preserve schedulability
  - *Proof: Homework*

Summary

- ***Rate-monotonic priority assignment*** is an optimal fixed priority assignment for periodic processes with $D = T$
- Simple ***utilization-based*** schedulability tests are not exact – they are sufficient but not necessary properties
- ***Response time analysis*** is more powerful and provides exact results (ignoring variations in execution times of the processes themselves, as always)
- Deadline-Monotonic scheduling is optimal for aperiodic processes with $D < T$

To Go Further

- Chapter 13 of [Burns and Wellings 2001]
- C. Liu and J. Layland, *Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment*, Journal of the ACM **20**(1): 46-61, 1973