Simple Process Model

- The application is assumed to consist of a fixed set of processes (tasks)
- All processes (tasks) are periodic with known periods
- The processes are completely independent of each other
- All system overheads, context-switch times and so on are ignored
  - Assumed to have zero cost or otherwise negligible
- All processes have a deadline equal to their period
  - Each process must complete before it is next released
- All processes have a fixed WCET

Standard Notation

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

Fixed-Priority Scheduling (FPS)

- This is the most widely used approach and is the main focus of this course
- Each process has a fixed, static, priority which is computed off-line
- The ready processes are executed in the order determined by their priority
- In real-time systems the “priority” of a process is derived from its temporal requirements, not its importance to the correct functioning of the system or its integrity

Preemption and Non-preemption – 1

- 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 other executes
- Preemptive schemes enable higher-priority processes to be more reactive, and hence they are preferred

Preemption and Non-preemption – 2

- 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 (Value Based Scheduling) can also take on a preemptive or non-preemptive form
  - VBS is useful when the system becomes overloaded and some adaptive scheme of scheduling is needed
  - VBS consists in assigning a value to each process and then employing an on-line value-based scheduling algorithm to decide which process to run next

FPS 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 any two processes `i` and `j`

$$T_i < T_j \Rightarrow P_i > P_j$$

- This assignment is optimal in the sense that if any process set can be scheduled (using preemptive 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 is the lowest (least) priority

Utilization-Based Analysis

- A simple sufficient but not necessary schedulability condition exists for rate monotonic scheduling
  - But only for task sets with D=T

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

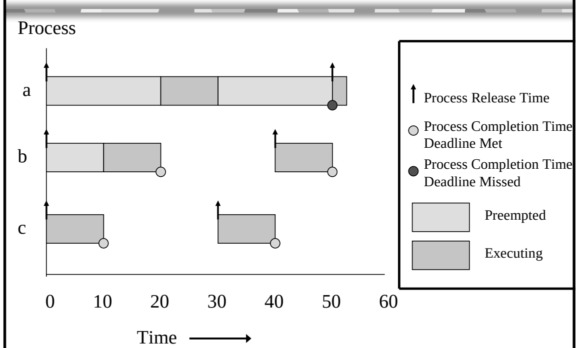
$$U \leq 0.69 \text{ as } N \to \infty$$

Process Set A

Process	Period T	Computation Time C	Priority P	Utilization U
a	50	12	1	0.24
b	40	10	2	0.25
c	30	10	3	0.33

- The combined utilization is 0.82 (or 82%)
- This is above the threshold for three processes (0.78) and, hence, this process set fails the utilization test

Timeline for Process Set A



Process Set B

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

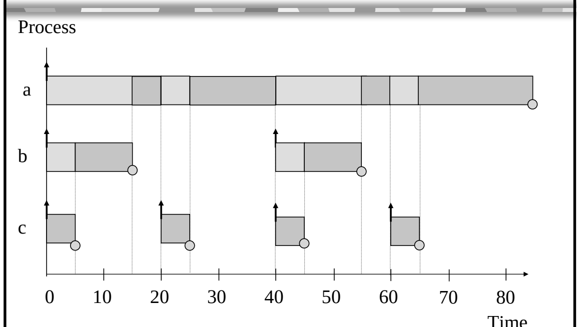
- The combined utilization is 0.775
- This is below the threshold for three processes (0.78) and, hence, this process set will meet all its deadlines

Process Set C

Process	Period T	Computation Time C	Priority P	Utilization U
a	80	40	1	0.50
b	40	10	2	0.25
c	20	5	3	0.25

- 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

Timeline for Process Set C



Criticism of Utilization-based Tests

- Not exact
- Not general
- BUT it is O(N)

The test is said to be sufficient but not necessary

13/29 © Burns and Wellings, 2001

Response Time Analysis

- The worst-case response time R of task i is calculated first and then checked (trivially) with its deadline

$$R_i \le D_i$$

$$R_i = C_i + I_i$$

Where I is the interference from higher priority tasks

14/29 © Burns and Wellings, 2001

Calculating R

During R, each higher priority task j will execute a number of times

$$\text{Number of Releases} = \left\lceil \frac{R_i}{T_j} \right\rceil$$

The ceiling function $\lceil\ \rceil$ gives the smallest integer greater than the fractional number on which it acts. So the ceiling of 1/3 is 1, of 6/5 is 2, and of 6/3 is 2.

The total interference is given by:

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

15/29 © Burns and Wellings, 2001

Response Time Equation

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

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

Solve by forming a recurrence relationship:

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

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

16/29 © Burns and Wellings, 2001

Response Time Algorithm

```
for i in 1..N loop -- for each process in turn
  n := 0
  w_i^n := C_i
loop
    calculate new w_i^{n+1}
    if w_i^{n+1} = w_i^n then
      R_i = w_i^n
      exit value found
    end if
    if w_i^{n+1} > T_i then
      exit value not found
    end if
    n := n + 1
  end loop
end loop
```

17/29 © Burns and Wellings, 2001

Process Set D – 1

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

$R_a = 3$

$$w_b^0 = 3$$

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

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

$$R_b = 6$$

18/29 © Burns and Wellings, 2001

Process Set D – 2

$$w_c^0 = 5$$

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

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

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

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

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

$$R_c = 20$$

Revisit: Process Set C

Process	Period T	Computation Time 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 utilization threshold for three processes (0.78) therefore it failed the test
- The response time analysis shows that the process set will meet all its deadlines

Response Time Analysis

- RTA is sufficient and necessary
- If the process set passes the test its processes will meet all their deadlines
- If it fails 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
- They also require D<T
- The response time algorithm for fixed-priority scheduling works perfectly for values of D less than T as long as the stopping criteria becomes $W_i^{n+1} > D_i$
- It also works perfectly well with any priority ordering
  - *hp(i)* always gives the set of higher-priority processes

Hard and Soft Processes

- In many situations the WCET for sporadic processes are considerably higher than the average
- Interrupts often arrive in bursts and an abnormal sensor reading may lead to significant additional computation
- Measuring schedulability with WCET 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 therefore be situations in which it is not possible to meet all current deadlines
  - This condition is known as a transient overload
- Rule 2
  All hard real-time processes should be schedulable using WCET and worst-case arrival rates of all processes (including soft)
  - No hard real-time process will therefore miss its deadline
  - If Rule 2 gives rise to unacceptably low utilizations for “normal execution” then action must be taken to reduce the WCET values or the arrival rates

Aperiodic Processes

- These do not have minimum inter-arrival times
- Can run aperiodic processes at a priority below the priorities assigned to hard processes
  - In a preemptive system they therefore cannot steal resources from the hard processes
- This does not provide adequate support to soft processes which will often miss their deadlines
- To improve the situation for soft processes, a server can be employed
- Servers protect the processing resources needed by hard processes but otherwise allow 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

$$D_i < D_j \Rightarrow P_i > P_j$$

DMPO is Optimal – 1

- Deadline monotonic priority ordering (DMPO) is optimal

  if any process set `Q` that is schedulable by priority-driven scheme `W` is also schedulable by DMPO

- The proof of optimality of DMPO involves transforming the priorities of `Q` (as assigned by `W`) until the ordering is DMPO
- Each step of the transformation will preserve schedulability

DMPO is Optimal – 2

- Let `i` and `j` be two processes (with adjacent priorities) in `Q` such that under `W`
  $$P_i > P_j \wedge D_i > D_j$$
- Define scheme `W'` to be identical to `W` except that processes `i` and `j` are swapped
- Now consider the schedulability of `Q` under `W'`
- All processes with priorities greater than will be unaffected by this change to lower-priority processes
- All processes with priorities lower than will be unaffected; they will all experience the same interference from `i` and `j`
- Process `j`, which was schedulable under `W`, now has a higher priority, suffers less interference, and hence must be schedulable under `W'`

DMPO is Optimal – 3

- All that is left is the need to show that process i, which has had its priority lowered, is still schedulable
- Under `W`
  $$R_j < D_j, D_j < D_i \text{ and } D_i \le T_i$$
- Hence process j only interferes once during the execution of i
- It follows that:
  $$R'_i = R_j \le D_j < D_i$$
- It can be concluded that process `i` is schedulable after the switch
- Priority scheme `W'` can now be transformed to `W"` by choosing two more processes that are in the wrong order for DMP and switching them