REAL TIME SYSTEM

LECTURER: ER. SAROJ GHIMIRE
QUALIFICATION: MSC.CSIT, BE(COMPUTER)

Lincoln University College

Overview

Priority-driven Scheduling of Periodic Tasks

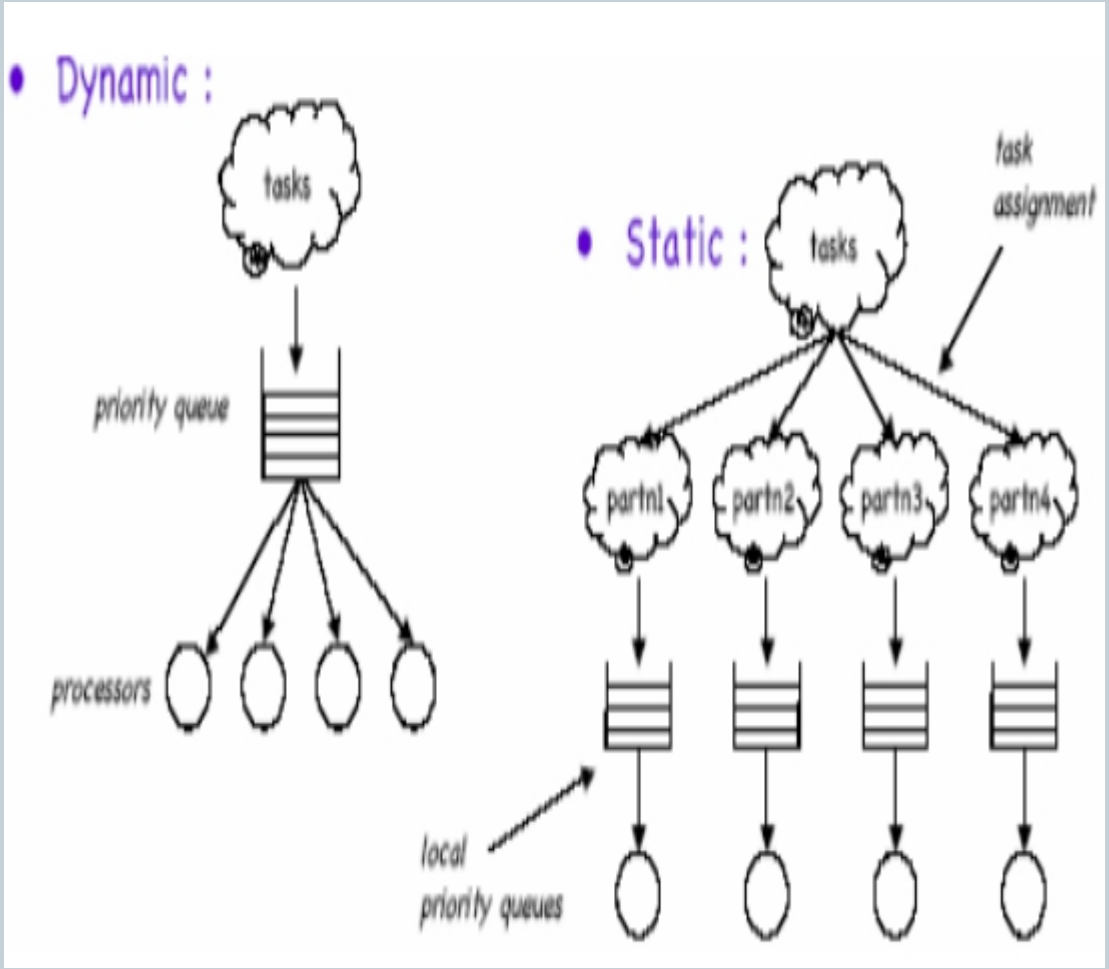
- ✓ Static Assumptions
- ✓ Fixed-Priority versus Dynamic-Priority Algorithms
  - Rate Monotonic and Deadline Monotonic Algorithms
  - Some Well Known Dynamic Algorithms
- ✓ Maximum Schedulable Utilization
- ✓ Sufficient Schedulability Conditions for the RM And DM Algorithms, Practical Factors

Objectives

- ✓ After studying this unit, you will be able to:
  - Describe Static Assumptions
  - Enumerate fixed-Priority Versus Dynamic-Priority Algorithms
  - Explain Maximum Schedulable Utilization

Priority-driven Scheduling of Periodic Tasks

- A multiprocessor priority-driven system is either dynamic or static. In a static system, all the tasks are partitioned into subsystems. Each subsystem is assigned to a processor, and tasks on each processor are scheduled(by themselves).
- In contrast, in a dynamic system, jobs ready for execution are placed in one common priority queue and dispatched to processors for execution as the processors become available.
- In the worst case, the performance of priority-driven algorithms can be unacceptably poor.



Static Assumptions

Priority-driven scheduling of periodic tasks on a single processor assumption are as follows:

- ✓ A fixed number of independent periodic tasks exist
  - Jobs comprising those tasks:
    - Are ready for execution as soon as they are released
    - Can be pre-empted at any time
  - Never suspend themselves
  - New tasks only admitted after an acceptance test; may be rejected
  - The period of a task defined as minimum inter-release time of jobs in task
- ✓ There are no aperiodic or sporadic tasks
- ✓ Scheduling decisions made immediately upon job release and completion
  - Algorithms are event driven, not clock driven
  - Never intentionally leave a resource idle
- ✓ Context switch overhead negligibly small; unlimited priority levels

Fixed Priority Versus Dynamic Priority Algorithms

- Priority-driven algorithms differ from each other in how priorities are assigned to jobs. We classify algorithms for scheduling periodic tasks into two types: fixed priority and dynamic priority.
- A **fixed-priority algorithm** assigns the same priority to all the jobs in each task. other words, the priority of each periodic task is fixed relative to other tasks.
- In contrast, a **dynamic-priority algorithm** assigns different priorities to the individual jobs in each task. Hence the priority of the task with respect to that of the other tasks changes as jobs are released and completed. This is why this type of algorithm is said to be "dynamic."

Rate Monotonic Algorithm

- A well-known fixed-priority algorithm is the rate-monotonic algorithm.
- This algorithm assigns priorities to tasks based on their periods:
  - the shorter the period, the higher the priority.
- The rate (of job releases) of a task is the inverse of its period. Hence, the higher its rate, the higher its priority
- The rate (of job releases) of a task = 1/period
- A given task is scheduled under rate monotonic scheduling Algorithm, if its satisfies the following equation:

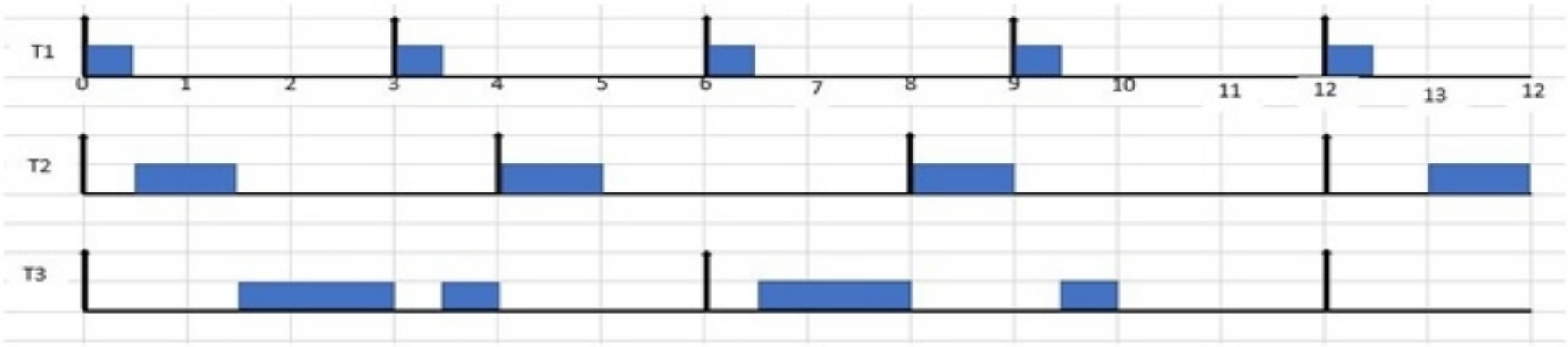
$$\sum_{k=1}^{n} \frac{C_k}{T_k} \leq U_{RM} = n(2^{1/n} - 1)$$

where n is the number of tasks in a task set, C_k is execution and T_k period.

- For example: (a): T1 = (4,1); T2 = (5,2); T3 = (20,5) .
  - Rate(T1)=1/4= .25
  - Rate(T2)= 1/5=.2
  - Rate(T3)=1/20=.05 , Therefore T1 has highest priority.

Rate Monotonic Algorithm

- Example t1, t2,t3 with execution time 0.5, 1 ,2 and period 3,4,6
  - Task set **U**= 0.5/3 +1/4 +2/6 = 0.167+ 0.25 + 0.333 = 0.75
  - As processor utilization is less than 1 or 100% so task set is schedulable and it also satisfies the above equation of rate monotonic scheduling algorithm.



Deadline Monotonic Algorithm

- ✓ Deadline-monotonic priority assignment is a priority assignment policy used with fixed-priority pre-emptive scheduling.
- ✓ With deadline monotonic priority assignment, tasks are assigned priorities according to their deadlines.
- ✓ The task with the shortest deadline is assigned the highest priority.
- ✓ A given task is scheduled under rate monotonic scheduling Algorithm, if its satisfies the following equation:

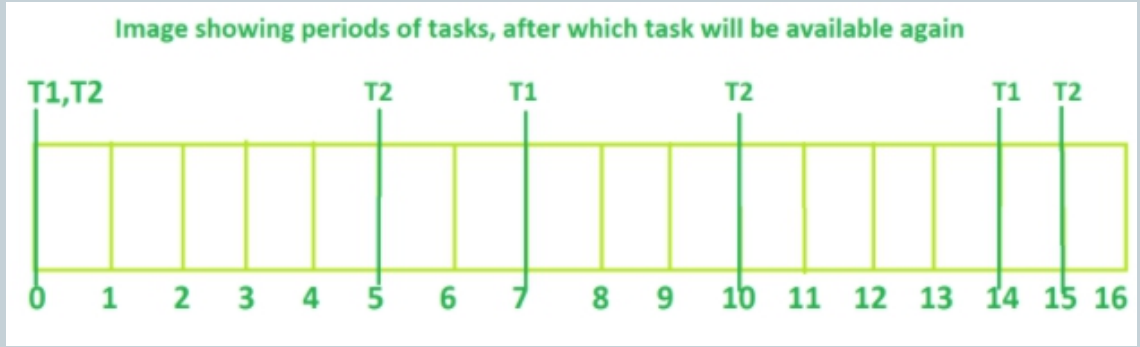
$$\sum_{k=1}^{n} \frac{C_k}{T_k} \leq U_{RM} = n(2^{1/n} - 1)$$

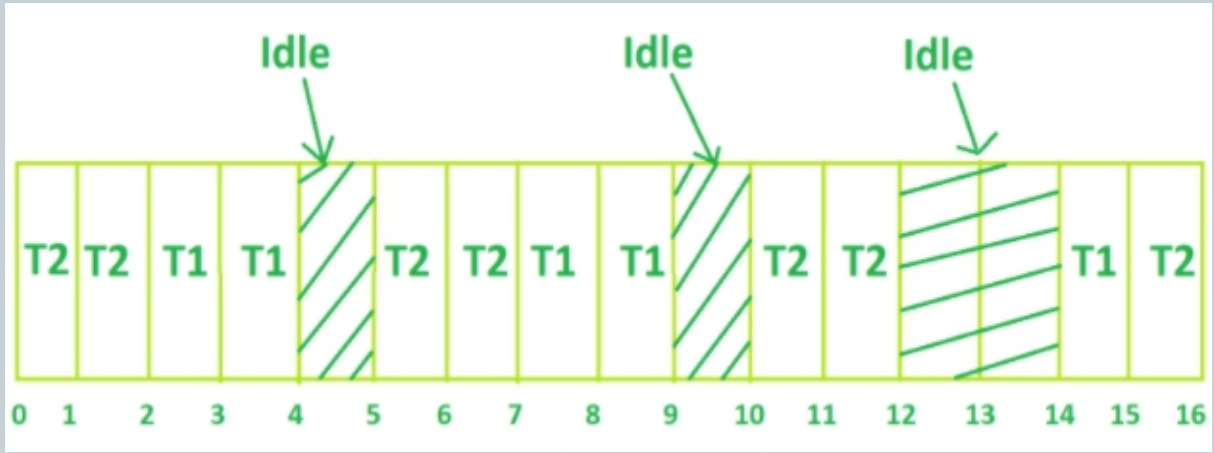
where n is the number of tasks in a task set, C_k is execution and T_k period.

Deadline Monotonic Algorithm

Example –Suppose there are two tasks that need to be executed.

- ✓ Task1 has release time 0, period 7 units, execution time 2 units, and deadline of 6 units (T1 (0, 7, 2, 6)).
- ✓ Task2 has release time 0, period 5 units, execution time 2 units, and deadline of 4 units (T2 (0, 5, 2, 4))





Some Well Known Dynamic Algorithms

- The EDF assigns priorities to individual jobs in the tasks according to their absolute deadlines; it is a dynamic-priority algorithm.

Example: Consider 3 periodic processes scheduled using **EDF**, the following acceptance test shows that all deadlines will be met.

Process	Execution Time = C	Period = T
P1	1	8
P2	2	5
P3	4	10

The utilization will be:

$$\frac{1}{8}+\frac{2}{5}+\frac{4}{10}=0.925=92.5\%$$

The theoretical limit for any number of processes is 100% and so the system is schedulable.

Maximum Schedulable Utilization

- we say that a system is schedulable by an algorithm if the algorithm always produces a feasible schedule of the system.
- A system is schedulable (and feasible) if it is schedulable by some algorithm, that is, feasible schedules of the system exist
- The schedulable utilization of a scheduling algorithm is defined as follows:

A scheduling algorithm can feasibly schedule any set of periodic tasks on a processor if the total utilization of the tasks is equal to or less than the schedulable utilization of that algorithm.

A periodic task is defined as a tuple (ϕ_i, p_i, e_i, D_i) where $u_i = e_i / p_i$

- Total utilization of the system $U = \Sigma_{i=1}^{n} u_i$ where $0 \leq U \leq 1$
- The ratio of the execution time e_k of a task T_k to the minimum of its relative deadline D_k and period p_k is called the density of the task
- Density of $T_k = e_k / \min(D_i, p_i)$
- The sum of the densities of all tasks in a system is the density of the system and is denoted by Δ
- If the density of a system is larger than 1, the system may not be feasible

Sufficient Schedulability Conditions for the RM and DM Algorithms

- Time-demand analysis method requires the periods and execution times of all the tasks in an application system to determine whether the system is schedulable. Before we have completed the design of the application system, some of these parameters may not be known.
- Sufficient Schedulability Conditions for the RM and DM Algorithms are:
  1. Schedulable Utilization of the RM Algorithm for Tasks with Schedulable Utilization of the RM ($D_i = p_i$).
  2. Schedulable Utilization of RM Algorithms as Functions of Task Parameters
  3. Schedulable Utilization of Fixed Priority Tasks with Arbitrary Relative Deadlines
  4. Schedulable Utilization of the RM Algorithm for Multi-frame Tasks

Sufficient Schedulability Conditions for the RM and DM Algorithms

Schedulable Utilization of the RM ($D_i = p_i$)

- A system of n independent, pre-emptible periodic tasks with relative deadlines equal to their respective periods can be feasibly scheduled on a processor according to the RM algorithm if its total utilization U is less than or equal to $U_{RM}(n) = n(2^{1/n} - 1)$.

Schedulable Utilization of the RM

- Since is not a necessary condition, a system of tasks may nevertheless be schedulable even when its total utilization exceeds the schedulable bound
- Example: Total utilization of T= {(3, 1), (5, 1.5), (7, 1.25), (9, 0.5)} is 0.85, which is larger than URM(4) = 0.756, but this system is schedulable according to the RM algorithm!

Enhanced Schedulable Utilization

- When some of the task parameters are known, this information allows us to improve the schedulable utilization of the RM algorithm:
  - The Utilization of Individual Tasks
  - The Number n_h of Disjoint Subsets Each Containing Simply Periodic Tasks
  - Some Functions of the Periods of the Tasks

Schedulable Utilization of the RM Algorithm for Multi-frame Tasks

- Consider a task that models the transmission of an MPEG compressed video over a network link.
  - Jobs in this task, modeling the transmissions of individual video frames, are released periodically.
  - The size of I-frames can be very large compared with that of Band P- frames, the execution times of the jobs can vary widely.
  - When modeled as a periodic task, the execution time of the task is equal to the transmission time of an I-frame

Practical Factors

- We have assumed that:
  - Jobs are pre-emptible at any time
  - jobs never suspend themselves
  - Each job has distinct priority
  - The scheduler is event driven and acts immediately
  - These assumptions are often not valid and how does this affect the system

Practical Factors

Blocking and Priority Inversion

- A ready job is blocked when it is prevented from executing by a lower-priority job; a priority inversion is when a lower-priority job executes while a higher-priority job is blocked.
- These occur because some jobs cannot be pre-empted:
  - Many reasons why a job may have non pre-emptible sections
    - Critical section over a resource
    - Some system calls are non-pre-emptible
    - Disk scheduling
  - If a job becomes non-pre-emptible, priority inversions may occur, these may cause a higher priority task to miss its deadline
  - When attempting to determine if a task meets all of its deadlines, must consider not only all the tasks that have higher priorities, but also non pre-emptible regions of lower-priority tasks.
- Add the blocking time when calculating if a task is schedulable

Practical Factors

Self-Suspension and Context Switches

1. Self-Suspension
   - A job may invoke an external operation (e.g. request an I/O operation), during which time it is suspended.
   - This means the task is no longer strictly periodic and again need to take into account self-suspension time when calculating a schedule.
2. Context Switches
   - Assume maximum number of context switches K_i for a job in T_i is known; each takes t_{CS} time units.
   - Compensate by setting execution time of each job, $e_{actual} = e + 2t_{CS}$ (more if jobs self-suspend, since additional context switches)

Practical Factors

Tick Scheduling

- All of our previous discussion of priority-driven scheduling was driven by job release and job completion events. Alternatively, can perform priority-driven scheduling at periodic events (timer interrupts) generated by a hardware clock i.e. tick (or time-based) scheduling.
- Additional factors to account for in Schedulability analysis:
  - The fact that a job is ready to execute will not be noticed and acted upon until the next clock interrupt; this will delay the completion of the job
  - A ready job that is yet to be noticed by the scheduler must be held somewhere other than the ready job queue, the pending job queue
  - When the scheduler executes, it moves jobs in the pending queue to the ready queue according to their priorities; once in ready queue, the jobs execute in priority order
  - Clear that non-ideal behavior can affect the Schedulability of a system

Self Assessment

Fill in the blanks:

1. ..... analysis method requires the periods and execution times of all the tasks.
2. As long as the total utilization of a system satisfies, it will never miss any.....
3. The multi-frame task model is a more ..... model.
4. ..... is the schedulable utilization of the RM algorithm when Di= pi for all.
5. A system of n independent, pre-emptible periodic tasks with relative deadlines equal to their respective periods can be feasibly scheduled on a .....
6. A ready job is blocked when it is prevented from executing by a ..... job.
7. A priority inversion is when a lower-priority job executes while a ..... job is blocked.
8. 8. The fact that a job is ready to execute will not be noticed and acted upon until the next ..... interrupts.
9. When the scheduler executes, it moves jobs in the ..... queue to the ready queue according to their priorities.
10. 10. A ready job that is yet to be noticed by the ..... must be held somewhere other than the ready job queue

Self Assessment

1. Time-demand
2. Deadline
3. Accurate
4. $U_{RM}(n)$
5. Processor
6. Lower-priority
7. Higher-priority
8. Clock
9. Pending
10. Scheduler

Further Readings

- Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Prentice Hall, 1988.
- Data Structures and Algorithms; Shi-Kuo Chang; World Scientifi c.
- Data Structures and Effi cient Algorithms, Burkhard Monien, Thomas Ottmann, Springer.
- Kruse Data Structure & Program Design, Prentice Hall of India, New Delhi
- Mark Allen Weles: Data Structure & Algorithm Analysis in C Second Adition. Addison-Wesley publishing
- RG Dromey, How to Solve it by Computer, Cambridge University Press.
- Shi-kuo Chang, Data Structures and Algorithms, World Scientifi c
- Sorenson and Tremblay: An Introduction to Data Structure with Algorithms.
- Thomas H. Cormen, Charles E, Leiserson & Ronald L. Rivest: Introduction to Algorithms.
- Prentice-Hall of India Pvt. Limited, New Delhi Timothy A. Budd, Classic Data Structures in C++, Addison Wesley