Topic 4 Notes

Scheduling theory and algorithms

Broad definition: It is the temporal allocation of activities to resources to achieve some desirable objective. Real time scheduling determines the order of dynamic vs. static real time task executions. Dynamic schedule computed at run-time based on tasks really executing, priorities computed on the fly while Static schedule is done at compile time for all possible tasks, priorities are already known

There are six popular scheduling algorithms.

i. First-Come, First-Served (FCFS) Scheduling
ii. Shortest-Job-Next (SJN) Scheduling
iii. Priority Scheduling
iv. Shortest Remaining Time
v. Round Robin(RR) Scheduling
vi. Multiple-Level Queues Scheduling

i. First Come First Serve (FCFS)

Jobs are executed on first come, first serve basis. It is a non-preemptive, pre-emptive scheduling algorithm thus easy to understand and implement based on FIFO queue. Poor in performance as average wait time is high.

ii. Shortest Job Next (SJN)

This is also known as **shortest job first**, or SJF which is a non-preemptive, pre-emptive scheduling algorithm. Best approach to minimize waiting time since it is easy to implement in Batch systems where required CPU time is known in advance. Impossible to implement in interactive systems where required CPU time is not known, the processer should know in advance how much time process will take.

iii. Priority Based Scheduling

Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems where each process is assigned a priority. Process with highest priority is to be executed first and so on and processes with same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

iv. Shortest Remaining Time

Shortest remaining time (SRT) is the preemptive version of the SJN algorithm where the processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion. It is impossible to implement in interactive systems where required CPU time is not known. Often used in batch environments where short jobs need to give preference.

v. Round Robin Scheduling

Round Robin is the preemptive process scheduling algorithm where each process is provided a

fix time to execute, it is called a **quantum**. Once a process is executed for a given time period, it is preempted and other process executes for a given time period and context switching is used to save states of preempted processes.

vi. Multiple-Level Queues Scheduling

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics. Multiple queues are maintained for processes with common characteristics where each queue can have its own scheduling algorithms priorities assigned to each queue.

Types of Real Time Systems

1. **Hard real time systems** which Must always meet all deadlines and if this is not the case System may fails if deadline window is missed
2. **Soft real time systems-** Must try to meet all deadlines and System does not fail if a few deadlines are missed
3. **Firm real time systems-** Result has no use outside deadline window and Tasks that fail are discarded
4. **Rate Monotonic-** Simplest type of real time scheduling where Tasks are periodic, with hard deadlines completely independent and do not communicate with each other according to priority and task priorities are fixed and also Computation time is known and constant. Higher priorities usually assigned to tasks with smaller periods

   If t(h) < t(l), then PR(h) > PR(l),
   where t indicated period and PR indicates the priority.
   This is called rate monotonic priority assignment.

   Calculates the critical instant for each task that Occurs when task Ti and all higher priority tasks are scheduled simultaneously If tasks deadline scheduled at critical instant, then the task can always meet its deadline.
5. **Deadline Monotonic -** Tasks with shorter deadlines get higher priority with Static Scheduling.

   If D(h) < D(l), then PR(h) > PR(l),
   where D indicates the deadline.
   This is called Deadline Monotonic priority assignment.

   Dynamic Scheduling assume a preemptive system with dynamic priorities like deadline monotonic, the task with shortest deadline gets highest priority, but the difference is real time priorities can vary during the system's execution. Priorities are reevaluated when events such as task arrivals, completions occur

Real time tasks

1. Periodic- Each task is repeated at a regular interval, Max execution time is the same each period, Arrival time is usually the start of the period and Deadline is usually the end

2. Aperiodic - Each task can arrive at any time

Real Time Synchronization

This is required when tasks are not independent and need to share information and synchronize. If two tasks want to access the same data, semaphores are used to ensure non-simultaneous access.

Potential Issues are:

1. Chain Blocking – A situation in which more than two resources are available and a high priority task is blocked off from both because of two or more lower priority tasks holding locks on one or both of the resources.

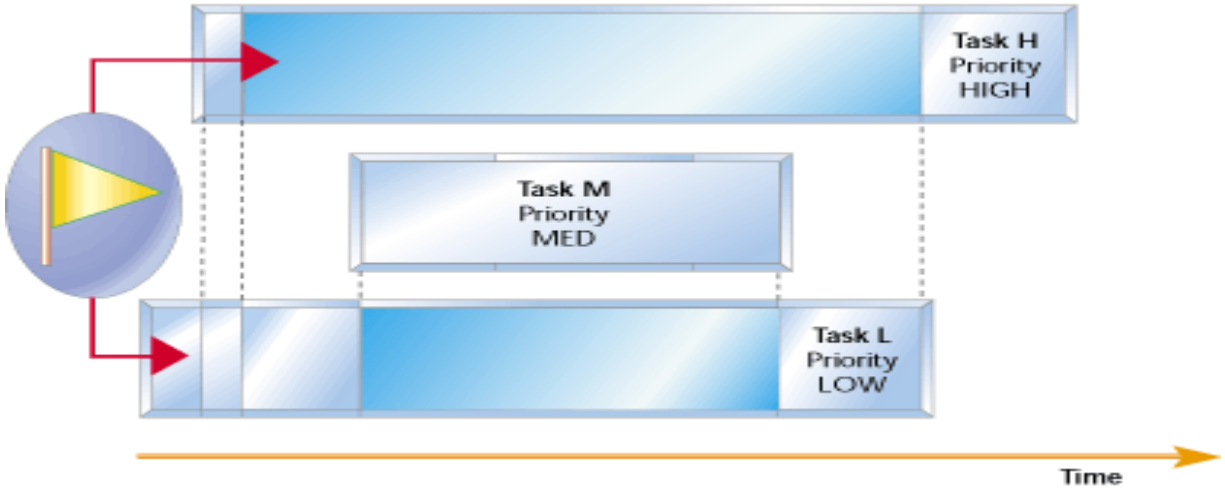
2. Priority Inversion – A situation in which a higher priority job is blocked by lower priority jobs for an indefinite period of time

Priority Inversions

Low priority task holds resource that prevents execution of higher priority tasks.

1. Task L acquires lock
2. Task H needs resource, but is blocked by Task L, so Task L is allowed to execute
3. Task M preempts Task L, because it is higher priority

Thus Task M delays Task L, which delays Task H



Priority Inheritance Protocol
PIP eliminates priority inversion problem and the algorithm will increase the priority of a task to the maximum priority of any task waiting for any resource the task has a resource lock on i.e. if a lower priority task L has a lock on a resource required by a higher priority task H, then the priority of the lower task is increased to the priority of the higher task, Once the lock is released the task resumes back its original priority.

Priority Ceiling Protocol
Each resource is assigned a priority ceiling, which is a priority equal to the highest priority of any task which may lock the resource and the PCP eliminates chain blocking which considers the use of more than one resource or semaphore. A task can acquire a lock on resource S only if no other task holds a lock on resource R. Thus higher priority tasks will not be blocked through both S and R If a high priority task is blocked through a resource, then the task holding that resource gets the priority of the high priority task. Once the resource is released, the priority is reset back to its original Semaphore queue priority assignment for real-time multiprocessor synchronization, Hardware support for Priority Inheritance using System-on-a-Chip Lock Cache (SoCLC) Separate

cache allocated on a chip for handling lock hand-offs. Improves real-time predictability of the system and improves performance Deferrable scheduling for fixed priority systems E.g. Actively schedule the maximum time between periodic sensor updates to minimize energy consumption in a wireless sensor network. Enhancements in Multiprocessor and Distributed environments resulting in better utilization of resources Real time scheduling algorithms for Artificial Intelligent systems that take decisions and prioritize execution of tasks based on heuristics Context-Aware Scheduling the decision of which job to schedule next should be based on the deadline of the job as well as the context of resources being managed.