Real-Time Systems

Week 1 - Concept

What is a real-time system ?

- A real-time system is a system that must complete its tasks and provide its service in a time sensitive manner.
- It typically must monitor, respond to, or control an external environment.
- It is connected to its environment through sensors, actuators, and other input-output devices.
- Must meet timing constraints imposed by external environment.

Examples of Real-time systems

- Vehicle systems
  - ABS breaking, power steering, dynamic traction control
- Traffic control
  - Traffic lights, air traffic control
- Process control
  - Power plants, chemical processes, consumer applications
- Communication systems
  - Voice and video transmission, data transmission
- Medical systems
- Automated manufacturing

Jobs & Tasks

- As we wish to characterize a wide range of real-time systems, we want to ***discuss the work done in general terms.***
- Every unit of work scheduled and executed on system is referred to as **a job.**
- We refer to a set of related jobs that together perform a system function, as **a task.**
- So, we might have a task T_i, composed of k jobs
  - $J_{i,1}$, $J_{i,2}$, . . . , $J_{i,k}$

Release time

- The instant a job is available to be executed, we refer to this as its **release time.**
- A job can't be scheduled and executed before its release time.
- After its release time, must also wait for any data or control dependencies before it can be scheduled and executed.

Release time

- Example: Say a system monitors and controls several furnace.
  - After it starts executing at t = 0, it reads each temperature sensor and stores the value in memory every 100ms.
  - It also computes the control law for each furnace every 100ms, with the first computation occurring at t = 20ms.
- We can capture the fact control laws are computed periodically in terms of release time.
  - i.e. Jobs J_0, J_1, . . . , J_k have release times of 20 + k × 100ms.

Deadlines

- A job's **deadline** is point in time that its execution must be completed by.
- If a job has no deadline, we say its deadline is at infinity.
- We refer to a job's **response time** as the length of time between its release time, and moment it completes.
- A job's maximum allowed response time is called its **relative deadline**.
- ***A job's deadline is thus equal to its release time plus relative deadline***.

Hard and soft timing constraints

- A **hard real-time** timing constraint is a constraint that causes system failure, if violated.
  - controlling the motion of an elevator
  - stopping a train before a collision, that a track switch position before train reaches switch
- A **soft real-time** timing constraint is a constraint that allows the system to operate effectively if a few deadlines are missed.
  - voice transmission data during a telephone call (a few packets dropped probably won't be missed)
  - electronic games (if a frame is slightly delayed now and then, not a big deal.)

Reference model

- A real-time system is characterized by three things:
  - **Workload model**: describes the system's applications.
  - **Resource model**: describes the system's resources.
    - This includes processors as well as passive resources such as memory, mutexes, etc.
  - **Scheduling algorithms**
    - Describes how the system makes use of its resources.

Workload models

- Periodic tasks
  - a deterministic workload model.
- Aperiodic tasks
  - have soft or no deadlines.
- Sporadic tasks
  - have hard deadlines.

Periodic taks - I

- A **periodic task** is a computation or data transfer that occurs on a regular or semiregular basis in order to provide an ongoing function of the system.
- Each periodic task T_i is composed of constituent jobs
  - $J_{i,1}$, $J_{i,2}$, . . . where $J_{i,k}$ is the kth job in T_i
- The release time, or phase, of task T_i is the release time ($r_{i,1}$) of $J_{i,1}$, and is labelled (Φ_i)
- The period of task T_i is (p_i). This means the jobs of T_i get released every p_i time units.

Periodic taks - II

- We use (D_i) to refer to the relative deadline of task T_i, and (d_i) to refer to the absolute deadline.
- We often assume that $D_i = \Phi_i$,
  - a job from T_i has time till the next job is released to complete.
- Sometimes we have $D_i < \Phi_i$ in order to reduce jitter (variation in the completion times).

Periodic taks - II

- The **execution time** (e_i) of task T_i is the maximum time it takes for any one of its jobs to complete.
- The **utilization** of task T_i, $u_i = e_i/\Phi_i$, is the fraction of time the task keeps a processor busy if it executes for its maximum time for each job.
- Total utilization of processor is
  - $u = \sum_{i=1}^{n}(ei/\Phi i)$

Aperiodic and Sparodic tasks

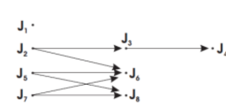
- Jobs in a non-periodic task arrive (are released) according to some probability distribution A(x).
  - This is the probability that the time between releases is x.
- The execution time of the task is given by some probability distribution B(x).
- The time between successive job arrivals can be arbitrarily small.
- **Aperiodic tasks** have soft or no deadlines. Our goal is to optimize response time of these tasks, but not at expense of hard real-time tasks.
- **Sporadic tasks** have hard deadlines.

Precedence Constraints

- Jobs that can execute in any order are said to be independent.
- Jobs have precedence constraints when they are constrained to execute in a given order.
  - We use the relation $J_i < J_k$ to indicate that job J_k may not start until job J_i has completed.
  - We say that J_i is a predecessor of J_k, and that J_k is a successor of J_i.

Precedence Constraints

- A job is ready to execute after its release time and when all of its predecessors have completed.
- Directed graph G = (J,<) is called a precedence graph and shows the predecessor relationships.



Preemptivity of jobs

- Execution of jobs can occur in interleaved fashion.
- Execution of less urgent job could be suspended in order to give processor to a more urgent job.
- When job completes, processor is returned to less urgent job to continue execution.
- When we interupt job execution like this, it's called preemption.
- We say a job is preemptable if it can be suspended at any time, allow other jobs to execute, and then be resumed from suspension point.

Scheduler

- A scheduler implements a set of scheduling algorithms and resource access-control protocols used to decide when individual jobs are to be executed and what resources they should be allocated.
- In particular, a scheduler assigns jobs to processors for execution.
- We assume that a scheduler will always produce a valid schedule.

Scheduler

- A valid schedule satisfies the following conditions:
  - 1. Every process is assigned at most one job at a time.
  - 2. Every job is assigned to at most one processor at a time.
  - 3. No job is scheduled before its release time.
  - 4. The amount of processor time assigned to every job is equal to actual execution of job or its maximum execution time.
  - 5. All precedence and resource usage constraints are satisfied.

Terminology for Schedules

- A **schedule** is an assignment of all jobs in system on the processors, which are available.
- A **feasible schedule** is one where all jobs complete by their deadlines.
- A group of jobs are **schedulable** according to an algorithm, if algorithm always produces a feasible schedule.
- A scheduling algorithm is **optimal** if the scheduling algorithm always produces a feasible schedule when one exists.

Performance measures

- Tardiness
  - Difference between a job's completion time and its deadline, zero if deadline met.
  - Interested in maximum and average tardiness.
- Lateness
  - Difference between a job's completion time and its deadline, negative if early.
- Response time
  - The length of time between a job's release time, and moment it completes.

Performance measures

- Miss rate
  - Percentage of jobs executed but complete too late.
- Loss rate
  - Percentage of jobs discarded.
- Invalid rate
  - Percentage of jobs that either complete too late orare discarded.

Next week …

- Scheduling algorithms
  - clock- driven approaches
  - priority driven approaches
  - preemption
  - online vs offline
  - …