Reference Model of Real-Time Systems
Introduction

• Keep track of relevant details.

• A good model focuses on:
  – timing properties,
  – Resource requirement,
  – Way the OS assigns available resources to system components.

• In this chapter, points 1 and 2 will be covered.
Processors and Resources

• Processors (servers or active resources):
  – Computers: execute machine instructions,
  – Transmission links: move data from place to another,
  – Disks: Retrieve files,
  – DB servers: process queries.

• Identical Processors: Can be used interchangeably.

• \( m \) processors are denoted \( P_1, P_2, \ldots, P_m \).
Processors and Resources

- Resources (passive resources): Memory, sequence numbers, mutexes, DB locks.
- Example 1: Job sharing data with other jobs. Data protected by semaphores. The lock of a semaphore is a resource.
- Example 2: DB jobs (update, query). The DB server uses a locking mechanism. The lock is a resource.
- Reusable Resources.
- A resource has one or more units, each unit is used by one job.
- Plentiful resources (memory) are not mentioned in the model.
- Resources are denoted with the letter $R$. 
Jobs Characteristics

• Temporal parameters.
• Functional parameters.
• Resource parameters.
• Interconnection parameters.
Temporal Parameters of RT Workload

• Reminder: Job and Task definitions.
• Assumption: Many parameters of hard RT jobs (tasks) are known. Example: Number of tasks is known.
• Number of jobs of a hard RT system is known.
• Reminder: Release Time ($r_i$), absolute deadline ($d_i$), relative deadline ($D_i$).
• Feasible interval: ($r_i, d_i$)
Fixed, Jittered and Sporadic Release Times

• $r_i \in [r_i^-, r_i^+]$
• $[r_i^-, r_i^+]$ is called the jitter in $r_i$ or release-time jitter.
• If jitter negligible compared to other temporal parameters $\rightarrow$ Fixed release time.
• Release time of jobs in response for an external events: Sporadic release time.
  – $A(x)$ probability distribution of the interrelease time.
Execution Time

• Execution time \( (e_i) \): amount of time required to complete execution of \( J_i \) when it executes alone and has all resources required.

• \( e_i \in [e_i^-, e_i^+] \)

• \( e_i \) is unknown a priori.

• \( e_i^- \) and \( e_i^+ \) are known for hard real-time job.
Periodic Task Model

- Well known deterministic workload model.
- Characterizes many hard real-time applications.
- Good performance and well-understood behavior.
Periods, Execution Times and Phases of Periodic Tasks

• Periodic Task: jobs executed repeatedly at regular or semi-regular time intervals.
• Period $p_i$ of a periodic task $T_i$: minimum length of all time intervals between release times of consecutive jobs.
• Execution Time of $T_i$: Maximum execution time of all the jobs in $T_i$. 
Remark

• Definition differs from one found in literature.
• Accuracy of periodic task model decreases with:
  – Increasing jitter in release times
  – Variation of execution times.
Notations

• Tasks in system: $T_1, T_2, \ldots, T_n$.
• Jobs in task $T_i$: $J_{i,1}, J_{i,2}, \ldots, J_{i,k}$.
• Phase of $T_i$ (Release time of the $1^{\text{st}}$ job of $T_i$): $\phi_i = r_{i,1}$.
• Hyperperiod of periodic tasks $H$: least common multiple of $(p_i)_{i = 1, 2, \ldots, n}$
• Max number of jobs in $H = \text{sum } (H/p_i)$
• Example: periods 3, 4 and 10. Hyperperiod = 60. $N = 61$
Notations

- Utilization of the task $T_i$: $u_i = e_i / p_i$
- Total utilization $U$: sum of $u_i$
- Example:
  - Execution times: 1, 1 and 3
  - Periods: 3, 4 and 10
  - Utilizations: 0.33, 0.25 and 0.3
  - $U = 0.88$
  - These tasks can keep processor busy 88% of time.
Aperiodic and Sporadic Tasks

• Aperiodic and Sporadic jobs: jobs executed in response to an external events.
• Their release-times are unknown a priori.
• Aperiodic Task: jobs have soft deadlines or deadlines.
• Sporadic Task: jobs have hard deadlines.
Precedence constraints and data dependency

• Execution order constraints.
• Jobs with *precedence constraints* must be executed with some order.
• *Independent* jobs: no precedence order.
• *Consumer* job can’t begin until *Producer* job completes.
• Examples: Radar surveillance system, information system.
Precedence Graph and Task Graph

• $J_i$ is a predecessor of $J_k$: $J_k$ can’t begin until $J_i$ completes. Notation: $J_i < J_k$.

• $J_k$ is a successor of $J_i$.

• *Immediate* predecessor/successor.
Data Dependency

- Jobs communicating via shared data.
- Data dependency cannot be captured by a classical precedence graph.
Other Types of Dependencies

• **Temporal Dependency**: Jobs constraints to complete within certain time from one another.

• **AND/OR precedence constraints**:
  – **AND precedence constraints**: All immediate predecessors must complete before execution.
  – **OR precedence constraints**: One or more of immediate predecessors must complete before execution. Represented by a square.
Conditional Branches

• Only one immediate successor to be executed.
• Represented by filled circles on task graph.
Pipeline Relationship

- Consumer (job) can begin execution when Producer (job) have completed.
Functional Parameter

• Preemptivity of jobs.
• Criticality of jobs.
• Optional Execution.
• Laxity type and Laxity Function.
Preemptivity of jobs

• Preemptable Job: can be suspended and resumed from the suspension point.
• Examples: Computations.
• Nonpreemptable: Data transmission.
Criticality of Jobs

- Criticality of a Job: Positive number indicating how critical (important) the job is.
- Criticality increases with the importance of the job.
- Don’t use the terms *Priority* and *Weight*. 
Optional Executions

- If an *Optional job* or *optional portion* completes late or is not executed at all, the system still functions satisfactorily.
- Non-Optional job: Mandatory.
Laxity Type and Laxity Function

- Laxity type: indicates whether the job is hard or soft.
- Recall: Usefulness function that gives usefulness of job results as a function of tardiness.
- Hard RT Jobs: Usefulness becomes zero or negative as soon as job is tardy. “Better never than late”.
- Soft RT Jobs: Usefulness decreases gradually.
Resource Parameters of Jobs and Parameters of resources

- Preemptivity of Resources.
- Resource Graph.
Preemptivity of Resources

• Describing processors and resources independently of application → Parameter of resources: Preemptivity.

• Nonpreemptable resource: Each unit is to be used serially.
Resource Graph

• Use letters $P$ for processors and $R$ for (passive) resources.

• Resource parameters:
  – Type.
  – Number of available units.
Scheduling Hierarchy

• Task graph:
  – Processor time
  – Resource requirement
  – Timing constraints

• Resource graph:
  – Amounts of available resources
  – Resources attributes
  – Usage rules
Scheduler and Schedules

• Scheduler: implements scheduling algorithms
  – Schedules Jobs,
  – Allocates Resources. Example: Processors assignment on jobs (or vice versa).

• Schedule: Assignment of all system jobs on available processors produced by the scheduler.
Valid schedule

• Every Processor is assigned to at most one job at any time.
• Every job is assigned at most one processor at any time.
• No job is scheduled before its release-time.
• Total amount of processor time assigned to a job is equal to its execution time.
• Precedence and resource usage constraints are satisfied.
Feasibility, Optimality and Performance measures

• Feasible schedule: valid schedule where every job meets its timing constraints.

• Schedulable set of jobs: the scheduler always produces a feasible schedule.

• Optimal scheduling algorithm: always produces a feasible schedule, if it exists, for a set of hard real-time jobs.