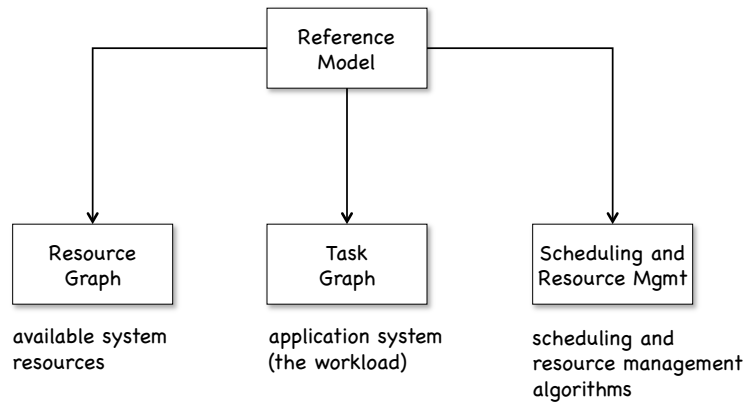


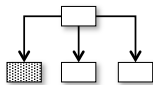
A Reference Model for Real-Time Systems

Goal:
 Abstract away from **functional** characteristics.
 Focus on **timing** and **performance** properties and **resource requirements**.



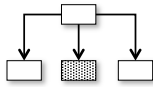
© R. Bettati

Processors and Resources



- **Processors:** servers, active resources
 P_1, \dots, P_m
- **Resources:** passive resources: needed in addition to the processor to make progress.
 R_1, \dots, R_S
- **Example:** sliding-window scheme:
 - Job: message transmission
 - Processor: data link
 - Resource: valid sequence numbers
- **Resource or Processor?** or The Art of Modeling.
- **Example:** I/O bus as resource or as processor?

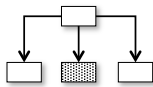
Workload and Temporal Parameters



- J_i : **Job**; unit of work
- τ_j : **Task**, set of related jobs.
e.g. **Periodic task** is sequence of invocations of identical jobs.
- r_j : **Release time** of Job J_j
- D_j : **Absolute deadline** of Job J_j
- d_j : **Relative deadline** of Job J_j
- C_j : (Maximum) **execution time** of Job J_j

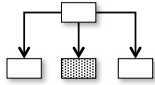
- Q:** Why do we use maximum execution time?
1. Variations of execution times typically small.
 2. Unclaimed portion of time and resources can be used for soft real-time portions.

The Periodic (Sporadic) Task Model



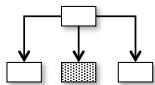
- **Tasks** τ_1, \dots, τ_n
- Each task consists of **jobs**: $\tau_i = \{T_{i1}, T_{i2}, \dots\}$
- ϕ_i : **Phase** of τ_i
- T_i : **Period** of τ_i ; minimum inter-release time
- H : **Hyperperiod** $H = lcm(T_1, \dots, T_n)$
- C_i : **Execution time** of τ_i
- u_i : **Utilization** of τ_i $u_i = C_i/T_i$
- D_i : (Relative) **deadline** of τ_i , typically $D_i = T_i$

Aperiodic and (strictly) Sporadic Tasks

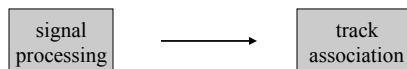


- Capture unexpected events
- $A(x)$: **Interarrival time** distribution
- $B(x)$: **Execution time** distribution
- Definitions:
 - **Aperiodic** tasks: Jobs have either soft or no deadlines.
 - **(strictly) Sporadic** tasks: Jobs have hard relative deadlines.

Precedence Constraints / Precedence Graph



- Precedence constraints reflect **data and control dependencies**
- e.g. Consumer/Producer in radar system:

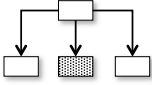


- Precedence relation \rightarrow (partial order)

$$J_i \rightarrow J_j \quad : \quad J_i \text{ is predecessor of } J_j$$

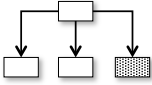
- **Precedence graph:** $G = (J, \rightarrow)$

Functional Parameters



- **Preemptivity:**
 - **Preemption:** Suspension of execution of job to give processor to more urgent job.
 - **Preemptable:** e.g. job on CPU, message in packet-switched network
 - **Non-preemptable:** e.g. data frame in token ring
 - Non-preemptability is typically tied to particular resource:
 - Job still preemptable on other resources.
 - What is the cost of preemption?
- **Criticalness:**
 - Can associate **weight** with jobs to indicate criticalness with respect to other jobs.
 - Schedulers and resource access protocols then **optimize weighted performance measures.**

Schedules and Scheduling Algorithms



- **Schedule:** assignment of jobs to available processors
- **Feasible schedule:** In a feasible schedule, every job starts at or after its release time and completes by its deadline.
- **Optimality** of a scheduling algorithm: A scheduling algorithm is **optimal** if it always produces a feasible schedule if such a schedule exists.
- Performance measures:
 - Number of tardy jobs.
 - Maximum or average absolute **lateness** $L_i = f_i - d_i$
 - Maximum or average **tardiness** $E_i = \max(0, L_i)$
 - Maximum or average **response time** R_i
 - **Makespan** (completion time)
