A Reference Model for Real-Time Systems

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

![Diagram of reference model]

- **Resource Graph:** available system resources
- **Task Graph:** application system
- **Scheduling and Resource Mgmt:** scheduling and resource management algorithms

Processors and Resources

- **Processors:** servers, active resources
  \[ P_1, \ldots, P_m \]

- **Resources:** passive resources: needed in addition to the processor to make progress.
  \[ R_1, \ldots, R_s \]

- **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?
Temporal Parameters

- $J_i$: Job; unit of work
- $T_i$: Task, set of related jobs.
  e.g. Periodic task is sequence of invocations of identical jobs.
- $r_i$: Release time of Job $J_i$
- $d_i$: Absolute deadline of Job $J_i$
- $D_i$: Relative deadline of Job $J_i$
- $e_i$: (Maximum) execution time of Job $J_i$

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 Task Model

- Tasks $T_1, \ldots, T_n$
- Each task consists of jobs: $T_i = \{J_{i1}, J_{i2}, \ldots\}$
- $\phi_i$: Phase of $T_i$
- $p_i$: Period of $T_i$: minimum inter-release time
- $H$: Hyperperiod $H = \text{lcm}(p_1, \ldots, p_n)$
- $e_i$: Execution time of $T_i$
- $u_i$: Utilization of $T_i$ $u_i = e_i/p_i$
- $D_i$: (Relative) deadline of $T_i$, typically $D_i = p_i$
Aperiodic and 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.
- **Sporadic** tasks: Jobs have hard relative deadlines.

Precedence Constraints / Precedence Graph

- Reflects data and control dependencies
- e.g. Consumer/Producer in radar system:

  \[
  \text{signal processing} \rightarrow \text{track association}
  \]

- Precedence relation $< \text{ (partial order)}$
  \[J_i < J_j : J_i \text{ is predecessor of } J_j\]

- Precedence graph: $G = (J, <)$

- Precedence constraints can be quite exotic, e.g. AND/OR:
  \[
  \begin{align*}
  \text{AND} & : \text{all predecessors must complete} \\
  \text{OR} & : \text{only some predecessors must complete}
  \end{align*}
  \]

- Not all dependencies can be captured by task graphs: e.g. access to shared data
  - imposed by scheduling algorithm
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: 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 **tardiness**.
  - Maximum or average absolute **lateness**
  - Maximum or average **response time**
  - **Makespan**