Real-Time Architectures
2003/2004

Reference model
Johan Lukkien

Contents
• Real-time task concepts and terminology
  – static
  – behavior
• Resources and scheduling
• Scope and place of model

Example: robots at conveyor belt
Model

- Model (of a system): Abstraction (of that system) leaving out details irrelevant to a given set of criteria
  - preserving the properties of interest

- (Scope) Our reference model
  - explicitly addresses relevant issues in real-time systems
  - ...but must be mapped eventually onto an execution environment
    - OS, hardware, run-time system, ...

Events and tasks

- Event: state change
  - usually at the RCTS boundary, under control of the environment, or from a timer
  - an event indicates a state change that requires a response

- Task: the sequence of actions that must be carried out to respond to an event

- Task instance, task execution or job: an instance of the action sequence of a task
  - we can speak about the \(i\)th instance of task \(j\)
  - an event generates a job ("releases a task") ...often 1-1, sometimes 1-n correspondence
    - an (internal) event can also be (partial) execution of a task

Task attributes

- A task has
  - a name (the \(j\)th task) \(\tau_j\)
  - a (worst case) execution time \(C_j\)
  - a (relative) deadline \(D_j\)
  - a period, sometimes \(T_j\)
  - a phase (start of job 0) \(\phi_j\)

- A job \(\tau_{j,i}\) has
  - a release time (or earliest-start-time) \(est_{j,i}\)
  - an (absolute) deadline \(dl_{j,i}\)
  - a start time (or beginning time) \(b_{j,i}\)
  - an end time \(e_{j,i}\)
Task deadlines

- **Deadline**: the maximum time before a job must complete
  - Relative: the value to be added to the earliest start time
  - Absolute: the result of the addition

- The value associated with execution of a job after the deadline determines the type of deadline

Deadline types

- **Soft**
  - A response is still valuable after the deadline, but value decreases steadily after that.
  - Example: interaction with human users. People get impatient.

- **Firm**
  - A response has no value after the deadline.
  - Example: a video frame that cannot be shown in time can be skipped.

- **Hard**
  - Damage is done if a response does not come in time.

Derived attributes

- \( L_{ji} = e_{ji} - d_{ji} \) (lateness)
- \( E_{ji} = \max \{ 0, L_{ji} \} \) (tardiness (active after deadline))
- \( X_{ji} = d_{ji} - e_{ji} - C_i \) (slack time)
- \( R_{ji} = e_{ji} - e_{ji} \) (response time)
- \( U = C_j / T_j \) (utilization for task \( j \))
- \( U = \sum U_j \) (total utilization)
- \( \text{lcm} \{ T_0, T_1, \ldots \} \) or \( T_j \times T_j \times \ldots \) (hyperperiod)
Other task properties

• Preemptable
  – may interrupt during job execution
  – depends on execution environment, e.g., preemptibility of used resources (e.g., packet transmission)

• Criticality: relative importance
  – weight
  – supports decisions under overload conditions
  – criticality of 0 represents an optional task

• Tasks may consist of a mandatory and an optional part
  – different criticality levels
  – IRIS: increased response with increased service
  – the more time given the better the results

System level

• System cycle time: (maximum) time an environment must wait between two ‘cooperations’ with the system
  – the inverse of the system throughput

• System latency (response time): worst-case event response time

Contents

• Real-time task concepts and terminology
  – static
  – behavior

• Resources and scheduling

• Scope and place of model
Task (event) types – arrival patterns

- Periodic
  \[ \text{est}_{j,i} = \phi_j + i \cdot T_j \]
- Sparse or sporadic
  \[ \text{est}_{j,i+1} = \text{est}_{j,i} + T_j \]
- Irregular
  \[ \text{est}_{j,i+1} = f(\text{State}_{j,i}) \]
  \[ \text{arrival time of next event can be computed} \]
- Bursty
  - two additional task attributes
  - burst interval and burst length: \( b_{i,j} \)
  - \( \forall t: \exists i :: t \leq \text{est}_{j,i} \leq t + b_{i,j} \leq b_{i,j} \]
- Unbounded
  - \( \text{est}_{j,i} \) given by probability distribution

Job (task instance) constraints

- Precedence
  - enforced limitations on order (producer/consumer relationships)
  - AND-type (wait for a whole set), OR-type (wait for one of a set)
  - typical example: only single task instance active
  - generally: wait for a condition on the state
- Resource sharing
  - mutual exclusion
- Data dependency
  - implied limitations on data usage, e.g. ordering
  (unsynchronized producer/consumer)
- Temporal dependency
  - limitation on time differences between job completions
  - e.g. synchronizing audio/video

Causal relationships

- Jobs can initiate other jobs
  - internal event or
  - just a sub-job (‘action’)
- Relationships can be
  - sequential
    - e.g. pipelined execution
  - parallel
    - e.g. divide and conquer
  - conditional / selective
    - one out of a set of jobs must be executed after a job
- Notes:
  - this division determines scheduling points where resource re-assignment can take place
Contents

• Real-time task concepts and terminology
  – static
  – behavior
• Resources and scheduling
• Scope and place of model

Resources, scheduling (allocation)

• Resources: processors, memory [also: objects], interface cards, bus, ...
  – what a resource is, depends on the model of the execution environment
    • i.e., OS, middleware or a run-time system
  – resources may have attributes (e.g., speed, size, ...)
• Resource scheduling (allocation): assignment of resources to jobs (and, by consequence, to tasks)
  – $S$ is a function that maps a time and a resource to a job: $S(t,r) = \tau_{j,i}$ means that $\tau_{j,i}$ is assigned resource $r$ at time $t$

Valid, feasible schedule

• Let $S(t,r) = \tau_{j,i}$
• $S$ is valid:
  – Every resource is assigned to at most one job
  – $S$ is a function
    – Allocation obeys job timing properties
      • e.g. $t \geq e_{t_{j,i}}$; total assigned time does not exceed $C_{j,i}$
      – Scheduling is consistent with task/job constraints
  – $S$ is feasible:
    – A valid schedule where required timing properties are satisfied
      • e.g. end-times before deadlines (leading to worst case analysis)
      – ... or minimal cost function (like maximum lateness, average response, loss rate, ...)
• Question:
  – What about: “every job assigned at most one processor?”
Algorithm, Schedulability

- Scheduling algorithm:
  - produces a schedule for a set of jobs and resources
- A set of jobs is schedulable if there exists an algorithm that produces a feasible schedule
- An algorithm is optimal if it finds a feasible schedule for every schedulable job-set

- NOTE:
  - both optimality and schedulability are with respect to a set of resources
  - both can be formulated also in terms of tasks

Contents

- Real-time task concepts and terminology
  - static
  - behavior
- Resources and scheduling
- Scope and place of model

Anomalies, limitations

- Theorem [Graham '69]. Suppose that a job set is optimally scheduled on a multiprocessor with some fixed priority assignment. Given are:
  - execution times
  - number of processors
  - precedence constraints
Then, the following can increase the schedule length:
  - reducing execution times
  - increasing the number of processors
  - weakening precedence constraints
Example jobset

- Prio(Ji) < Prio(Jk) = i > k
- Arrows give precedence relation

- Optimal schedule on 3-processor machine

Increased # processors

Reduced Computation time

- Reduce computation time by 1
Weakened precedence constraints

Scope of the model

- The overall task properties come from the system specification
  - external events, response times
- All subsequent details represent design
  - decomposition of tasks
  - internal concurrency, specific implementations
  - choice of preemption,
- ... and are driven by
  - an understanding of the building blocks where the task-set is to be mapped upon
    - i.e., the resource model (model of the execution environment)
      - processor(s), memory, size and number of resources, provided OS-type services, ...
      - pre-emptivity of resources
    - the need to find a feasible schedule
      - e.g., a task might need decomposition to make it schedulable

Misconceptions

- Tasks are not
  - threads
  - functions
  - processes
- though they may be mapped onto them
  - this is determined by the resource model
  - ...we come back to this when discussing mapping