Exam Review
Generics

- Definitions: hard & soft real-time
- Task/message classification based on criticality and invocation behavior
- Why special performance measures for RTES?
- What’s deadline and where is it coming from?
Estimation of task exec time

• Factors affecting task exec time
• Data-dependent exec path (conditional branches), interrupts
• What are 3 main features used to improve avg performance, and how (un)helpful to predictability?
• Why is cache OK, but not virtual memory?
Concurrent task exec time

- 4 basic building blocks for task composition
- Contiguous stretches of code $\Rightarrow$ activities
- GSPN $\Rightarrow$ CTMC $\Rightarrow$ concurrent task exec stages $\Rightarrow$ task completion times
- Modeling of send-receive-reply with GSPN
Task scheduling

- Uniprocessor or multiprocessor
- Preemptive or non-preemptive
- Offline or online
- Static or dynamic priority assignment
- Common approaches
  - Time-driven, e.g., cyclic executive
  - WRR
  - Priority-driven
Classical uniprocessor scheduling

- Rate Monotonic (RM)
- Deadline Monotonic (DM)
- Earliest-Deadline-First (EDF)
- Minimum-Laxity-First (MLF)

**Common task models**
- Characterized by period/interarrival time, phase, exec time, absolute/effective release time & deadline
- Task vs. job

**Common assumptions:** fully preemptable, independent, CPU only, relative deadline=period
Rate Monotonic Scheduling

• Optimal fixed priority sched alg
• Sufficient condition based on utilization bound
• Necessary & sufficient condition based on time demand analysis
Important Concepts

• Critical time instant: worst time $x^*$ at which to release $T_i$, i.e., $R_i(x^*) \geq R_i(x) \forall x$

• Critical time zone: $[x^*, x^* + R_i(x^*)]$

• Full utilization of a processor by a set of tasks
  – if RM schedule meets all deadlines, and
  – if increase of the execution time of any task in the set violates the RM schedulability.

• What if relative deadline $\neq$ period?
Sporadics and transient overload

• Sporadic tasks
  – Treat them as periodic
  – Use a periodic polling server
  – Use a deferred server
  – Which one is better and why?

• Transient overload
  – Period aggregation
  – Period splitting
Priority Inversion and Resolution

• Resource sharing via semaphore may cause priority inversion

• Solutions:
  – Priority inheritance: simple but deadlock
  – Priority ceiling protocol: complex but no deadlock
  – How to modify schedulability condition?
  – Need to be careful when priority ceiling is to be changed.
Preemptive EDF

- *Dynamic priority* scheduling algorithm
- Tasks don't have to be periodic
- **Optimal** uniprocessor sched alg.
- When all tasks are periodic and have relative deadlines = their periods, the task set is schedulable on a uniprocessor by EDF alg iff

\[ U = \sum_{i=1}^{n} \frac{e_i}{P_i} \leq 1.0 \]

- If relative deadlines != periods, the problem is complex
Precedence and Exclusion Constraints

- A excludes B ⇔ A is not allowed to preempt B
- Task containing OR subgraphs must be converted to one without them
- A schedule is valid if
  - processor is not left idle when one or more tasks ready to run
  - precedence, exclusion, and preemption constraints are all met throughout the schedule
Scheduling general task sets

• Generate valid initial schedule
• Partition tasks into subsets based on busy periods
• Shuffle the order of execution within each busy period using lateness-based heuristics
Other important special cases

- When there are primary and alternative versions for each task (interesting special case with harmonically-related task periods)
- When there are mandatory and optional parts of each real-time task => IRIS (increased reward with increased service) or imprecise computation model
- Mode changes
- Fault-tolerant scheduling
Multiprocessor scheduling

- Utilization-balancing algorithm
- Next-fit alg for RM scheduling
- Bin-packing assignment for EDF
- Myopic offline scheduling alg
- Combined assignment and scheduling
  - System hazard as objective
  - Two B&B (one for assignment and the other for scheduling assigned tasks)
Online load sharing of aperiodics

- 3 policies: xfer, location and info
- Bidding with focused addressing
- Adaptive LS
  - Buddy sets and preferred list
  - Bayesian decision to cope with comm delays
  - Operations at each node
  - How to model performance?
Real-Time Operating Systems

• Small proprietary (homegrown and commercial) kernels
• RT extensions to UNIX and others
• Research kernels
  – EMERALDS: CSD scheduling, task synchronization, IPC
  – RTDVS: CPU power conservation