Real-Time Systems

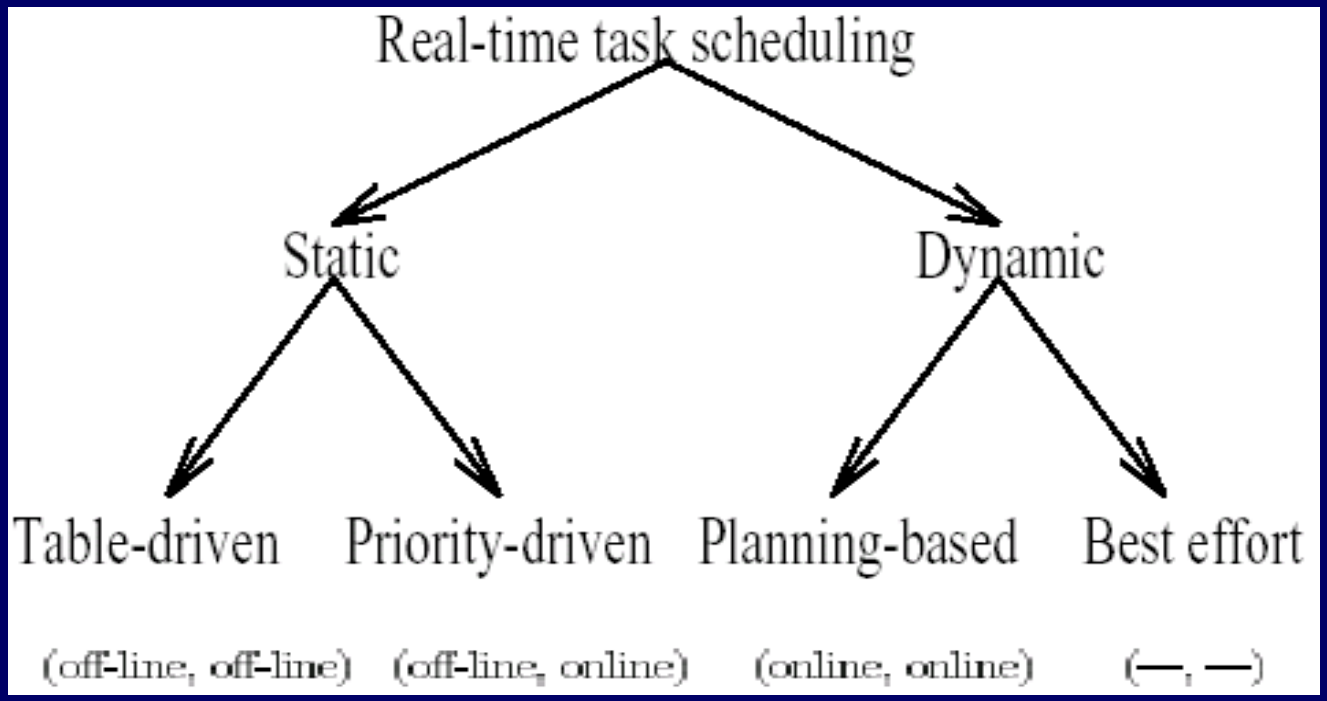
Basic Concepts (Contd.)

Real-Time Systems - Issues

- Resource Management (RM) Issues
  - Scheduling, Fault-tolerance, Resource reclaiming, Communication
- Architectural Issues
  - Computing subsystem, Communication subsystem, I/O subsystem
- Software Issues
  - Requirements, specification, and verification, Real-time languages, Real-time databases

Real-time Scheduling Paradigms – RM Issue

- Allocate time slots for tasks onto processor(s).
- [i.e., Where and When a given task executes]
- Objective: **predictably meeting task deadlines.**
- (schedulability check, schedule construction)



Preemptive vs Non-preemptive scheduling

- **Preemptive Scheduling**
  - Task execution is preempted and resumed later
  - Preemption occurs to execute higher priority task.
  - Offers higher schedulability
  - Involves higher scheduling overhead due to context switching

- **Non-preemptive Scheduling**
  - Once a task starts executing, it completes its full execution
  - Offers lower schedulability
  - Less overhead due to less context switching

Optimal scheduling -- definition

- A **static scheduling** algorithm is said to be optimal if, for any set of tasks, it always produces a feasible schedule (i.e., a schedule that satisfies the constraints of the tasks) whenever any other algorithm can do so.

- A **dynamic scheduling** algorithm is said to be optimal if it always produces a feasible schedule whenever a static algorithm with complete prior knowledge of all the possible tasks can do so.

- Static scheduling is used for scheduling periodic tasks, whereas dynamic scheduling is used to schedule both periodic and aperiodic tasks.

Architectural Issues

- Predictability in: Instruction execution time, Memory access, Context switching, Interrupt handling.
- RT systems usually avoid caches and superscalar features.
- Support for error handling (self-checking circuitry, system monitors).
- Support for fast and reliable communication (routing, priority handling, buffer and timer management).

Architectural Issues (contd..)

- Support for scheduling algorithms (fast preemptability, priority queues).
- Support for RTOS (multiple contexts, memory management, garbage collection, interrupt handling, clock synchronization).
- Support for RT language features (language constructs for estimating worst-case execution time of tasks).

Requirement, Specification, Verification

- **Functional requirements :** Operation of the system and their effects.
- **Non-Functional requirements :** e.g., timing constraints.
- F & NF requirements must be precisely defined and together used to construct the specification of the system.

Requirement, Specification, Verification (contd..)

- A **specification** is a mathematical statement of the properties to be exhibited by a system. It is abstracted such that
  - it can be checked for conformity against the requirement.
  - its properties can be examined independently of the way in which it will be implemented.

- The usual approaches for specifying computing system behavior entail enumerating events or actions that the system participates in and describing orders in which they can occur. It is not well understood how to extend such approaches for real-time systems.

Real-time Languages

- Support for the management of time
  - Language constructs for expressing timing constraint, keeping track of resource utilization.
- Schedulability analysis
  - Aid compile-time schedulability check.
- Reusable real-time software modules
  - Object-oriented methodology.
- Support for distributed programming and fault-tolerance

Real-time Databases

Conventional database systems

- Disk-based.
- Use transaction logging and two-phase locking protocols to ensure transaction *atomicity* and *serializability*.
- These characteristics preserve data integrity, but they also result in relatively slow and unpredictable response times.

Real-time database system, issues include:

- transaction scheduling to meet deadlines.
- explicit semantics for specifying timing and other constraints.
- checking the database system's ability of meeting transaction deadlines during application initialization.

Introduction: Summary

- Real-time systems require logical correctness and timeliness.
- Real-time system consists of a controlling system, controlled system, and the environment.
- Real-time systems are classified as: hard, firm, and soft RT systems.
- Workload (tasks) are periodic, aperiodic, sparodic.
- The notion of predictability is very important in real-time systems.
- Important issues include:
  - scheduling, resource reclaiming, fault-tolerance, communication, architectural issues, system specification and verification, programming languages, and databases.