Real -Time Systems

18-849b Dependable Embedded Systems

Kanaka Juvva

01/28/99

Required Reading: Generalized Rate-Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems, Lui Sha, Ragunathan Rajkumar and Sathaye.

Authoritative Books: Real-Time Systems Design Principles for Distributed Embedded Applications, Herman Kopetz

Carnegie Mellon

Overview: Real-Time Systems

- **Introduction**
  - What is a Real-Time System
  - Classification of Real-Time Systems
  - Examples
- **Key concepts**
  - Modeling Real-Time Systems
  - Real-Time Scheduling
  - Real-Time Communication
- **Tools**
  - Real-Time Operating Systems
  - Real-Time Middleware Services
- **Conclusions & future work**

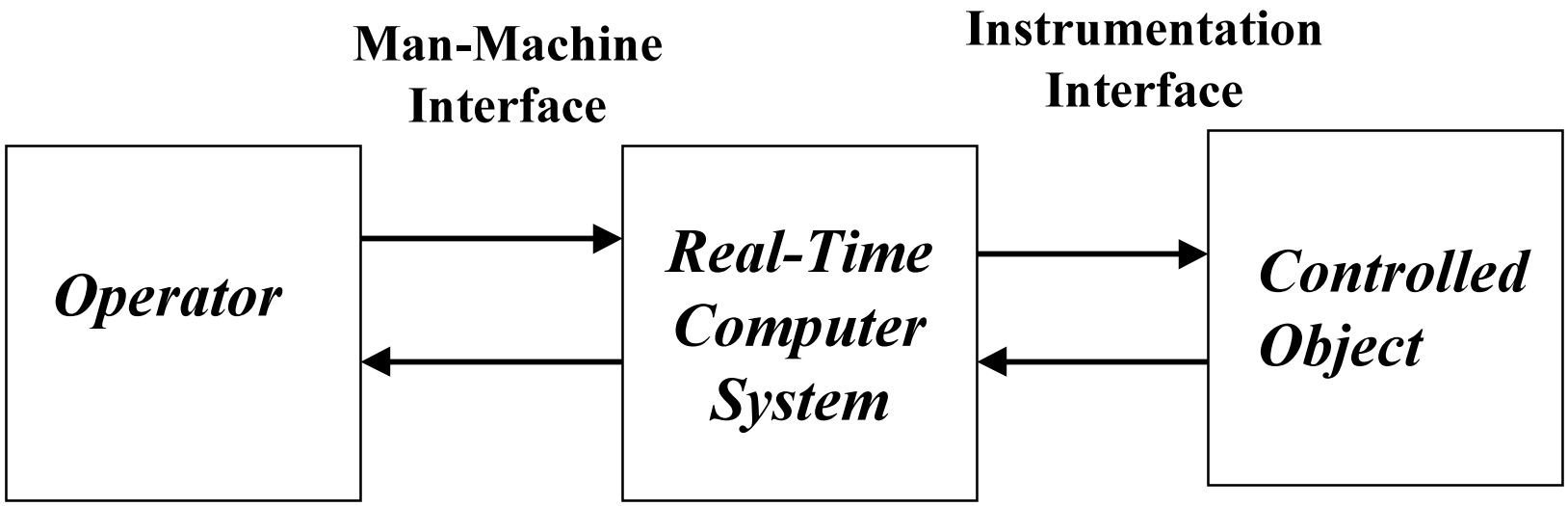
What is a Real-Time System ?

A Real_Time System is the one in which the correctness of the output depends not only on the logical results, but also on the physical instant at which results are produced.

Correct Output = Correct Result + Correct Time

Functional Requirements: What the system has to do ?
Temporal Requirements : deadline for the function

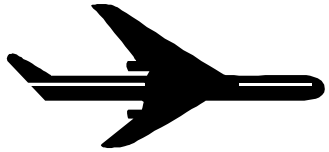
Components of a Real-Time System:



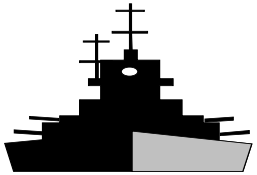
Environment = Operator + Controlled Object

Examples:

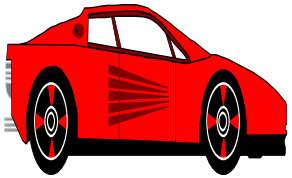
- Air Traffic Control



- Defense Systems



- Embedded Real-Time Systems



- Multimedia Systems

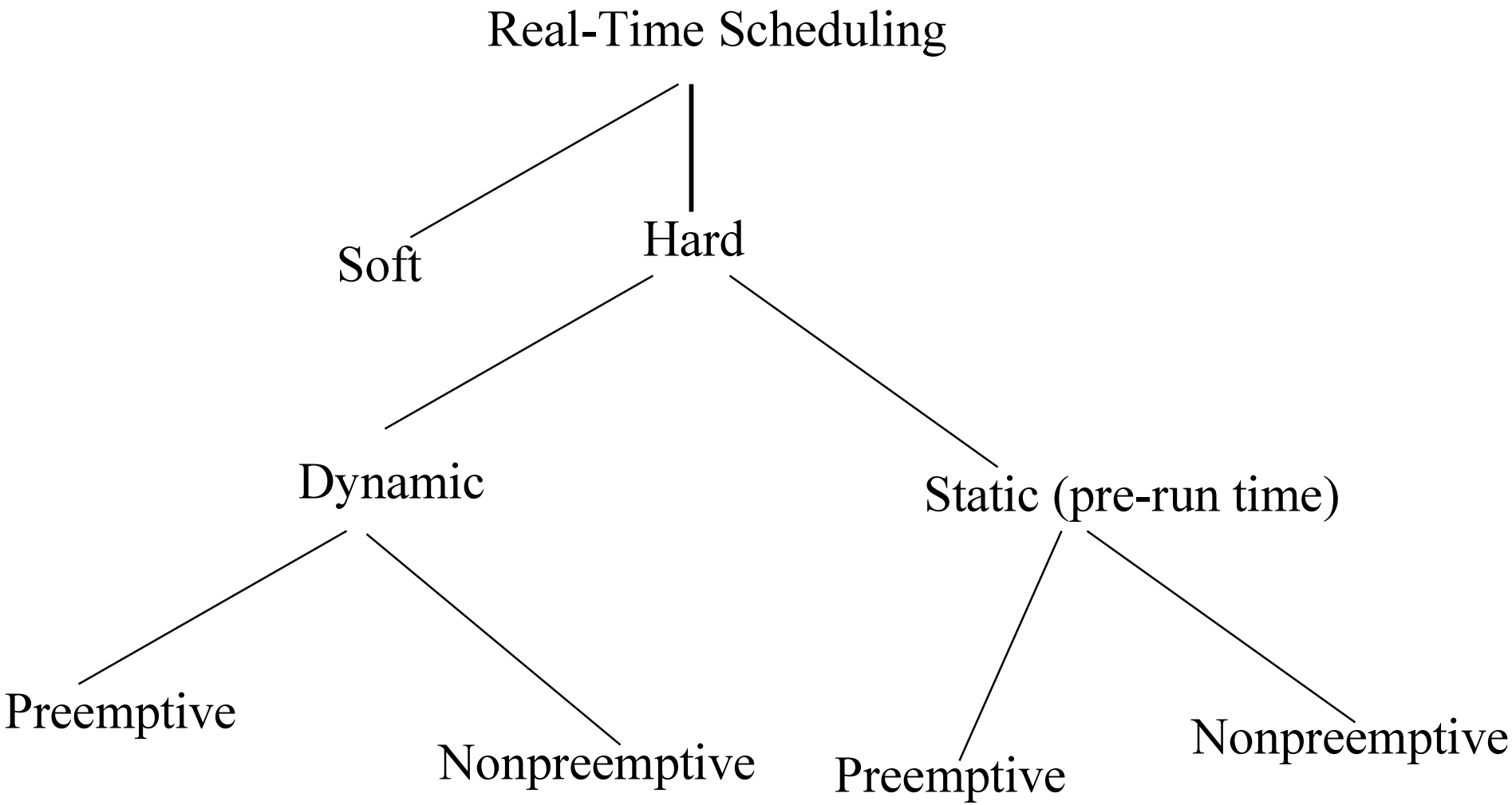
Classification:

Characteristic	Hard real-time	Soft real-time
Response Time	*Hard-required*	*Soft-desired*
Peak-load performance	*Predictable*	*Degraded*
Control of pace	*Environment*	*Computer*
safety	*Often critical*	*Non-critical*
Size of data files	*Small/medium*	*Large*
Redundancy type	*Active*	*Checkpoint-recovery*
Data integrity	*Short-term*	*Long-term*
Error detection	*Autonomous*	*User assisted*

Modeling Real-Time Systems

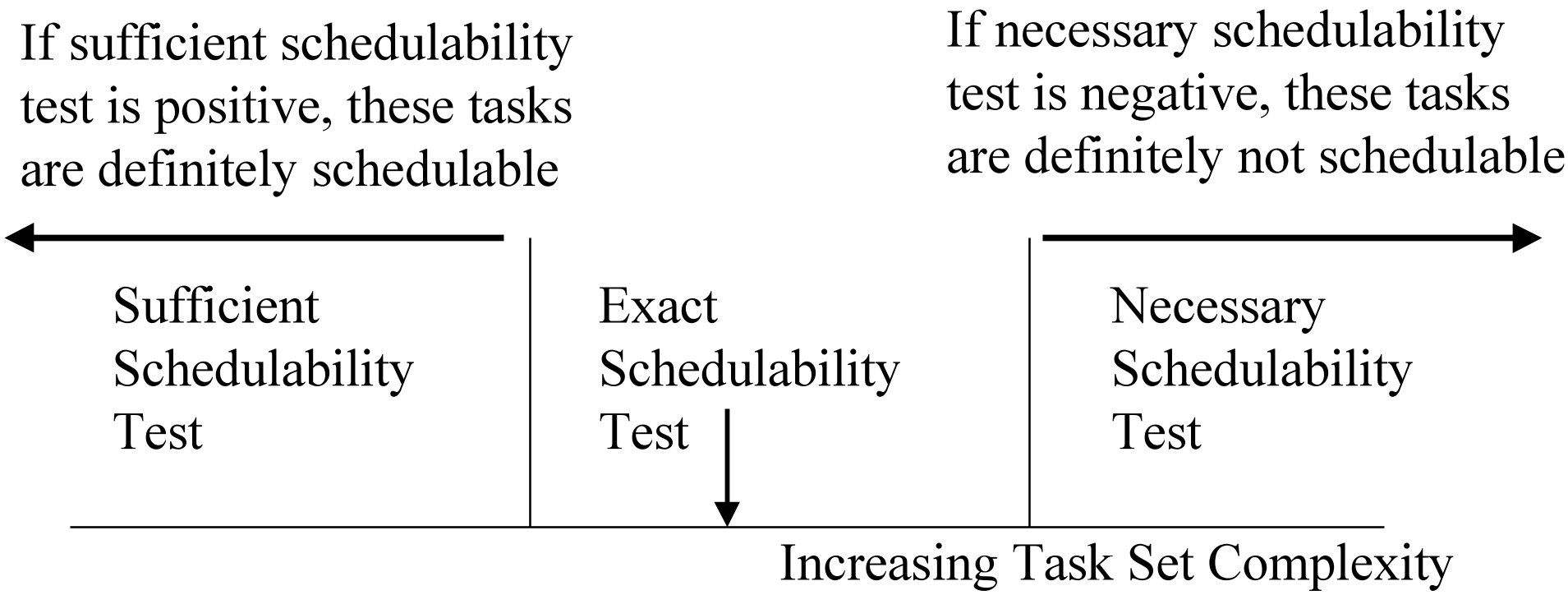
- **Temporal Control**
  - Scheduling:
    - When a task must be execute
  - Worst Case Execution Time (WCET)
  - Influence of Caches, Pipelines and Context Switches
- **Logical Control**
  - Control Flow within a sequential task
- **Merging of the above brings the complexity**
- **Structural Elements**
  - Task
  - Nodes
  - Interfaces

Real-Time Scheduling: Taxonomy



Schedulability Test:

- **A Test that determines whether a set of ready tasks can be scheduled such that each task meets its deadline**
- **Exact Schedulability Test is NP-Complete**



Dynamic Scheduling

- **Rate Monotonic Algorithm (Liu and Layland)**
  - Schedules Independent Tasks
  - Dynamic preemptive algorithm based on static task priorities
  - $\Sigma\,(C_i/T_i + B_i/T_i) \leq U(n) = n(2^{1/n} - 1)$

    C_i = worst-case task execution time of $task_i$
    T_i = period of $task_i$
    U(n) = Utilization bound for n tasks
    B_i = Blocking Time

- **Earliest-Deadline-First (EDF)**

Dynamic Scheduling Contd..

- **Dependant Tasks**
  - Tasks with precedence and mutual exclusion constraints
  - NP Complete again : more intractable
  - Kernelized Monitors
  - Priority Ceiling Protocols
    - Schedules a set of periodic tasks that have exclusive access to common resources protected by semaphores
    -
- **Distributed Systems**
  - Tindell analyzes distributed systems that use CAN bus

Tools:

- **Real-Time Operating Systems**
  - Real-Time Mach (CMU)
  - SPRING (UMASS)
  - Nemesis (Cambridge)
  - Solaris (Sun)
  - VxWorks (WindRiver)
- **Middleware Services**
  - RT-CORBA
  - RT-JAVA
  - Push-Pull Communication Services

Conclusions & Future Work

- **Distributed Real-Time Systems**
  - Opens up several frontiers of Research
- **Composability**
- **Merging with Internet**

GRMS: A Framework for Developing Real-Time Systems

- **Description of an example is the cool thing in this paper**
- **Key Issues**
  - Distributed System Extensions for GRMS
    - Extensions to Schedulability
    - Preemption Control
    - System Consistency
- **Summary**
  - Software Scheduling Abstractions
  - Hardware Scheduling Abstractions
  - Description of an example