Real time systems

Fabien Calcado, Thomas Megel

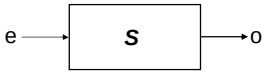
Email: fabien.calcado@gmail.com
thomas.megel@fr.thalesgroup.com

Outlines

- What is real time ?
- Multitask programming
  - Interrupt principle
    - Interrupt management on UNIX
- Characteristics of tasks in real time systems

What is real time ?

- Simple system



  - e : input of S system (entry)
  - S : process
  - o : output of S system

o = F(e)

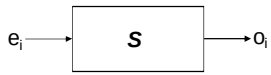
➔ non real time (not related to time execution)

  - Analysis
    - Data structure, Algorithms

State 1 ➔ Program ➔ State 2

What is real time ?

- Classical system (*Multiple data flows*)



  - e_i : input of S system (entry)
  - o_i : output of S system

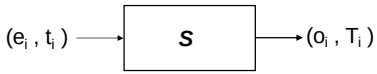
$s_i = F(e_i)$

$s_i = F(e_i, e_{i-1}, e_{i-2} \ldots) = F(E_{i-1}, e_i)$

  - Characteristics
    - Continuous execution
    - Event triggered
    - Processing and I/O parts succede and interleave
    - The system can wait an event

What is real time ?

- **Real time system**

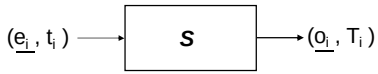


  - Data flows follow temporal laws
  - (e_i , t_i) : received events by *S* system
  - (o_i , T_i) : send events from *S* system

$$o_i = F(E_{i-1}, e_i , t_i)$$
$$T_i = G(E_{i-1}, e_i , t_i)$$

What is real time ?

- **Digital real time system**



  - Continuous environment ➔ discrete environment
  - Physical inputs are discretized
  - Digitalization consequences
    - Approximative I / O
    - Accurate computations (!)
    - Direct applications : signal processing, non linear filtering…

What is real time ?

- **Definition of a real time system**
  - A real time system is a system with the capacity to **handle** asynchronous events from physical environment in a timely manner (bounded response time)

    - Any temporal constraints shall be met
      - » *Otherwise the system is considered as defective*
      - » *Time scale depends on the corresponding application*

    A real time system is not a fast system !

What is real time ?

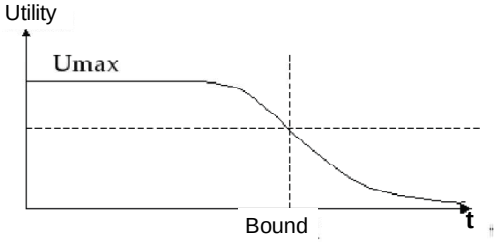
- **Problematic**
  - To verify that any specified temporal constraints are met
    - Before the effective execution of the system
- **To meet temporal constraints**
  - To characterize temporal behavior of each entity in the system
    - Out of the scope
  - To organize the set of processing parts corresponding to the entities, meeting any temporal constraints
    - Real time scheduling

What is real time ?

- **Distinct « real time » notions**
  - **Hard** real time or critical system
    - To miss real time constraint is considered as a system failure
      - » *It can cause a malfunction (incident risk)*
      - » *Subject to severe constraints in terms of safety*
      - » *Example : avionics, aerospace, rail transportation, nuclear powerplant, trading room management, telemedecine*
  - **Soft** real time system
    - Temporal constraints misses are authorized to some extent
      - » *No catastrophic consequences/failures (quality of service)*
      - » *Example : mobile phones, videoconference, video game lans…*

What is real time ?

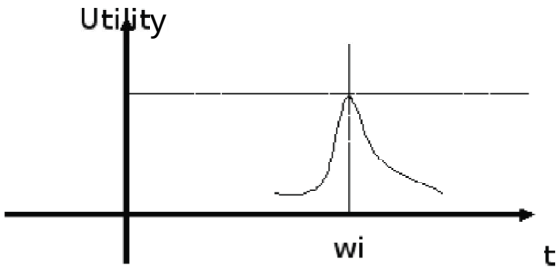
- **Temporal utility**
  - Response validity with respect to the time
    - Zero utility : response is now useless…
  - Soft real time case



  - Hard real time case?

What is real time ?

- **Temporal utility in the critical case**
  - An accurate result is not sufficient, its production time is equally important
    - Wi : optimal instant



What is real time ?

- **The design of a real time system**
  - derived from its specifications
    - System input behaviors are well-known and well-defined
    - Expected outcomes generally are not fully specified :
      - » *interactions on shared data*
      - » *results utility*
    - Two systems with the same implementation can be considered as real time or not with respect to their specifications

What is real time ?

- **Specificities of real time applications**
  - Temporal constraints to meet
    - System failure when missed
  - Dedicated systems
    - Specific software and hardware .. (but mostly with COTS)
  - Industrial constraints
    - Embedded systems => weight, size, power… (SWaP)
    - Environmental constraints

What is real time ?

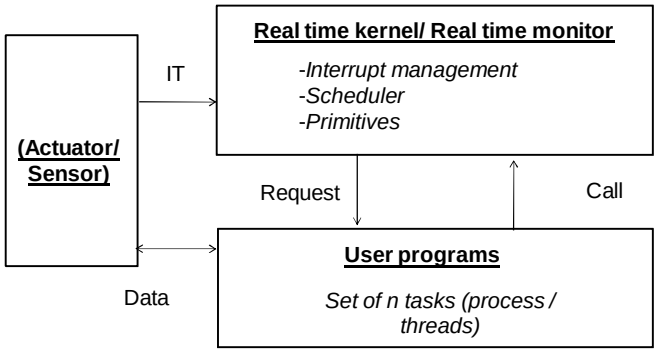
- **Real time term is rightly or wrongly used!**
  - Interactive systems
    - We are looking for the shortest time constraints
  - Email
    - Time constraints can be negligible
  - By malapropism, a system based on commonly used techniques from real time is sometimes called « real time » system
    - Multi-users system

What is real time ?

- **Important attributes of a real time system**
  - Determinism
    - Given a specific context, it always behaves identically
      - » *No uncertainty with regard to the system behavior*
  - Predictability
    - Application performances shall be defined in any cases to meet temporal constraints
      - » *worst case execution time analysis (WCET)*
  - Reliability
    - Capacity of a system to meet and to ensure its functionnalities under the normal conditions of use
      - » *In real time, it is related to meet time constraints*
      - » *In a case where the system remains safe even when some failures occured, it is the fault tolerance*

What is real time ?

- **Simplified scheme of a real time system**



Outlines

- **What is real time ?**
- **Multitask programming**
  - Interrupt principle
    - Interrupt management on UNIX
- **Characteristics of tasks in real time systems**

Multitask programming

- **Interrupt principle**
  - To notify the OS of the occurence from an internal or external event, expected or not
    - Hierarchical interruption (several level of priorities)
  - Interrupt is the only way for the OS to take back control of the processor
    - TP n°2 : software interrupt management
  - Instructions are sequentially executed
    - Jump
    - Exception (invalid instruction)
    - Trap (expected interruption)
    - Hardware or software interruption (hardware or software event)

Multitask programming

- **Atomicity of instructions**
  - Interruptible instant
    - The CPU has an observable state (that can be memorized) only at precise given instants where register values (state register, program counter…) are valid (coherent)
  - An assembly instruction is atomically executed
    - An interrupt can only occurs between two instructions
      - » *Be careful, instruction execution generally requires several cycles*
      - » *Atomicity of instruction is ensured between two preemptive tasks scheduled*

Multitask programming

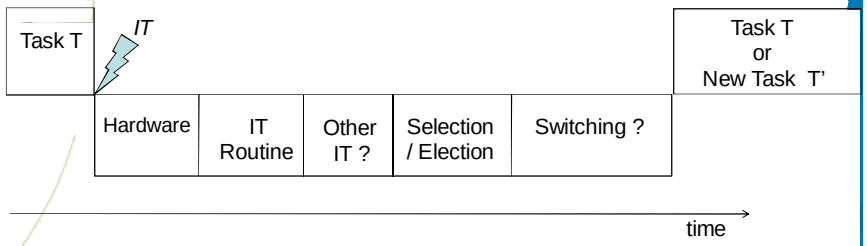
- **Interrupt management**
  - Manager
    - Software/Hardware called by the processor when interrupt occurs
    - In charge of controlling the consequences of the interrupt occurence
    - Is responsible for saving the execution context before preocessing (interrupt routine) then to restore it
      - » *One must not disturb the execution of the interrupted program*
  - **Context switching** or **task switching**
    - Operation caused by the system each time the processor should be allocated to another task that the one currently running

- **Steps for interrupt management**
  - Not maskable interrupt is received
  - Execution context saving of the current task
  - To enable of the interrupt manager
  - Interruption handling
    - Considering the interrupt type and priorities between interrupts
  - To resume the interrupted task (or to elect a new one ➔ switching)

- **Condition to be taken into account**
  - An interrupt request is not automatically handled as soon as it occurs
    - If an incident interrupt is not masked
    - If CPU is in a interruptible state
    - If there is not a higher priority interrupt waiting or currently handled
  - The quality of a real time system can be evaluated by the duration of the response time to handle an interrupt and its associated jitter

- **Task characteristics**
  - **Non-preemptible** task: task currently executed that cannot be interrupted
  - **Preemptible** task: task currently executed that can be interrupted (to the benefit of another one)
- **Constraints on some computation resources**
  - Multiple resources or not
  - **Interruptible** resources (CPU)
    - » *It indicates that the resource is preemptible*
  - **Non-interruptible** resources (DSP, DMA)
    - » *Different of non-preemptible, it only indicates its execution can be interrupted but with loss*

- **Temporal overhead of interrupt handling**
  - often neglected…
    - It becomes problematic in a real time context (performance issue)!



- **Interrupts under UNIX**
  - **A « signal » indicates an interrupt or a trap**
    - Either an external event outside the process/thread (task)
      - » *Keyboard hit, send signal by another thread thanks to Kill* primitive, *clock signal…*
    - Or an internal event from the process/thread (task)
      - » *Corresponds to an error (floating point error, memory protection…)*
  - **Possibility to mask signals**
  - **Signal waiting to be taken into account**
    - Corresponding bit in the signal register is at 1 for waiting

    ➔ If another signal of the same type arrives, it is lost! (see TP)

- **Interrupts under UNIX**
  - **Taking into account a signal = execution of a specific function called « handler »**
    - Predifined routine in the system
      - » *Standard processing or by default*
    - Routine implemented by user to personnalize the signal processing
      - » *TP n°2*

- **Multitask system**
  - **General purpose (classical system)**
    - Optimisation of resources usage
  - **Real time objectives**
    - To provide (correct) results to given dates
    - Manage different time scales
      - » *Mandatory to handle different periods of time*
  - **OS role**
    - Hardware interface
    - Scheduling of several tasks
    - Communication between tasks
    - To support independent execution of unrelated tasks
      - » *CPU, hard drive share, …*
        - » Robust partitioning in IMA approach

- **Benefit of a multitask conception**
  - **Initially: optimize hardware usage**
    - Several fonctionnalities on the same computer
    - Parallelize I/O with the CPU

      ➔WCET optimization of various tasks to execute

      (in a non real time context : average execution time)

      To take advantage of multiple computing resources
    - Multiprocessor architectures
  - **To ease the design**
  - **To handle asynchronous events**
    - Interrupt time contraints not entirely defined

      ➔ notion of importance or deadline

Multitask programming

- **Benefit of a multitask conception**
  - To handle a general real time goal
    - Different time scales
      - » *Short processing parts before long processing parts.*
    - Different degrees of criticality
      - » *Critical processing go first (if it is not possible otherwise)*
  - Often contradictory objectives
    - Maximize the number of functionnalities successfully completed
    - Minimize the cost (sizing)

Constraints and objectives in avionics

- Main objective:

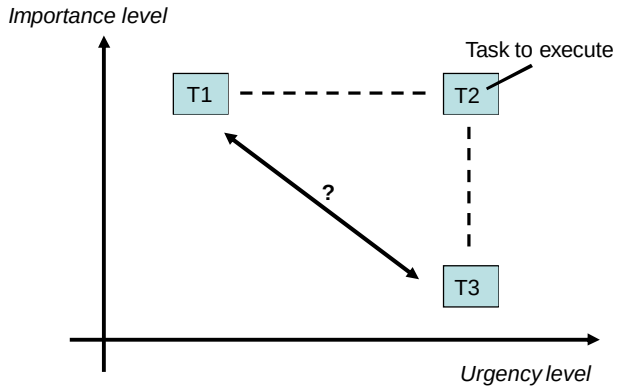
To guaranty the safety of people being transported and below

- There are also secondary objectives:
  - To ensure any kind of mission with success
  - To keep performance and overall cost under control
  - To protect environnement
- The actors:
  - State or supra-state governments
  - Individual or group of industries
  - User groups

Multitask programming

- **Other task characteristics**
  - Urgency level
    - It indicates the urgency of data provided by the task
    - Defined by task deadline
  - Importance level
    - Given a set of tasks, it allows introducing the capacity to be tolerant to temporal faults from some of them (failure)
    - The system shall be able to remove the execution of some tasks
      - » *To continue the execution of the most important ones (in degraded mode)*
  - To distinguish two tasks with the same urgency or importance
    - To deduce task priorities

Multitask programming



- **Classification of tasks by importance**
  - Critical
    - Shall always be ensured (to guarantee safety properties)
  - Essential
    - Shall be ensured as much as possible
      - » *i.e. at least from time to time (to guarantee punctuality properties)*

    ➔ ***In any cases for a real time system, one must ensure punctuality of any processing***

  - Safety
    - To bring proof that some event cannot happen
  - Liveness
    - To bring proof that some event will not happen after a period of time
  - Punctuality
    - To bring proof that processing parts will finish in time

- **To correctly schedule a multitask real time system**
  - 100% of critical tasks shall meet their (temporal) constraints
    - ➔ proof
  - For essential tasks
    - ➔ best effort

  - Difference between soft and hard real time system
    - Hard: no temporal fault tolerated
      - » *Catastrophic damages*
    - Soft : a temporal fault is acceptable
      - » *Damages which involve a low cost and can be tolerated compared to its probability of occurence*