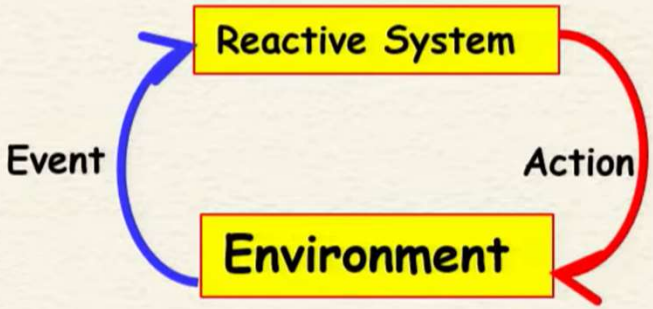
What is a Reactive System?

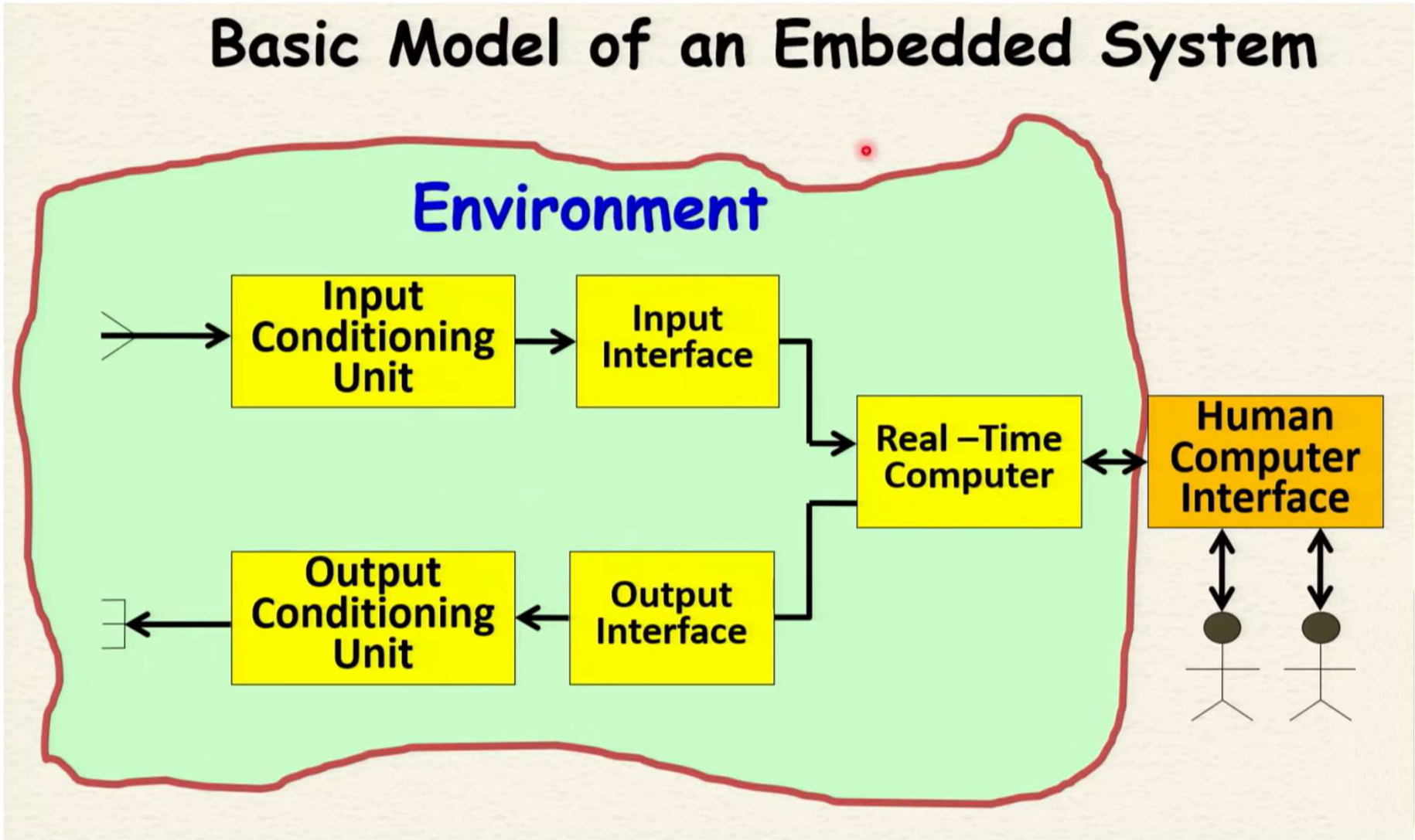
- Non-terminating interaction with Environment:
  - Responds to inputs (events)
  - Events occur due to changes to the environment
  - Responses are called actions
  - Embedded systems are usually reactive, can be described in terms of response to events



Types of Events

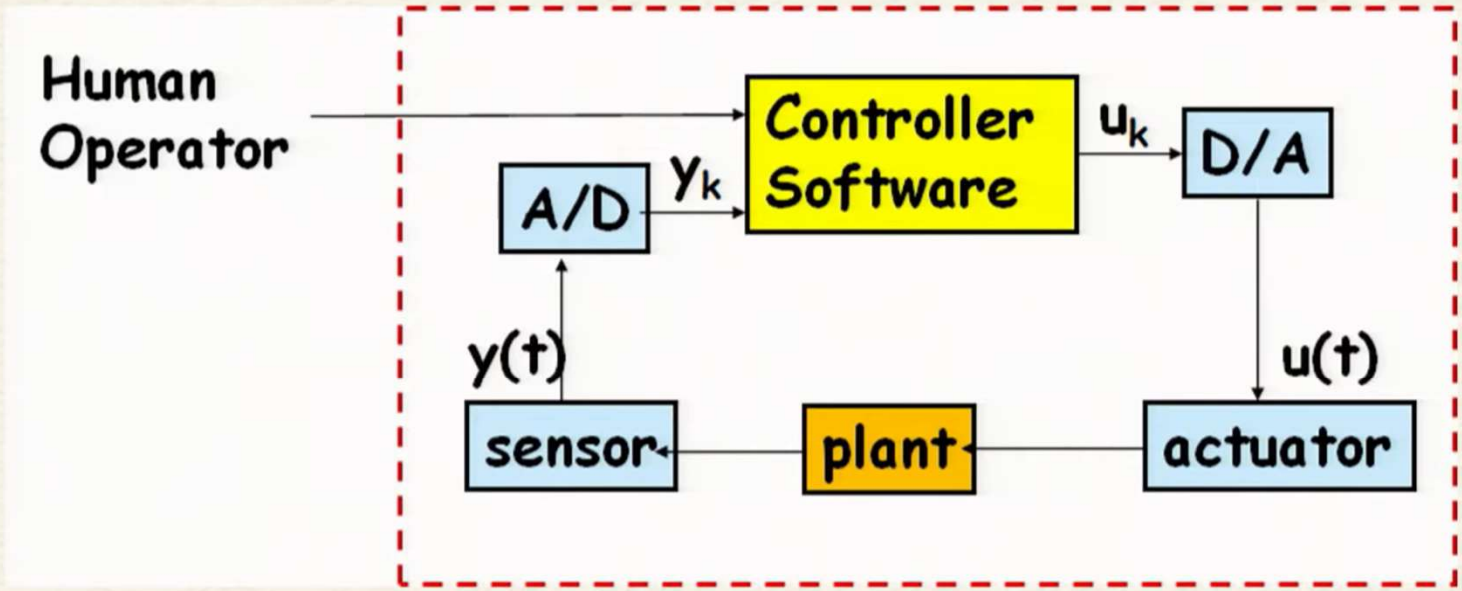
- **Periodic:**
  - Occur according to a timer
  - Vast majority (all activities for small systems) are periodic, e.g. Polling temperature, pressure, etc.
- **Sporadic:**
  - Occur statistically (randomly), but critical, e.g. fire alarm
- **Aperiodic:**
  - Occur randomly but non-critical, e.g user queries for reporting some parameter readings

Basic Model of an Embedded System



Simple Embedded Systems

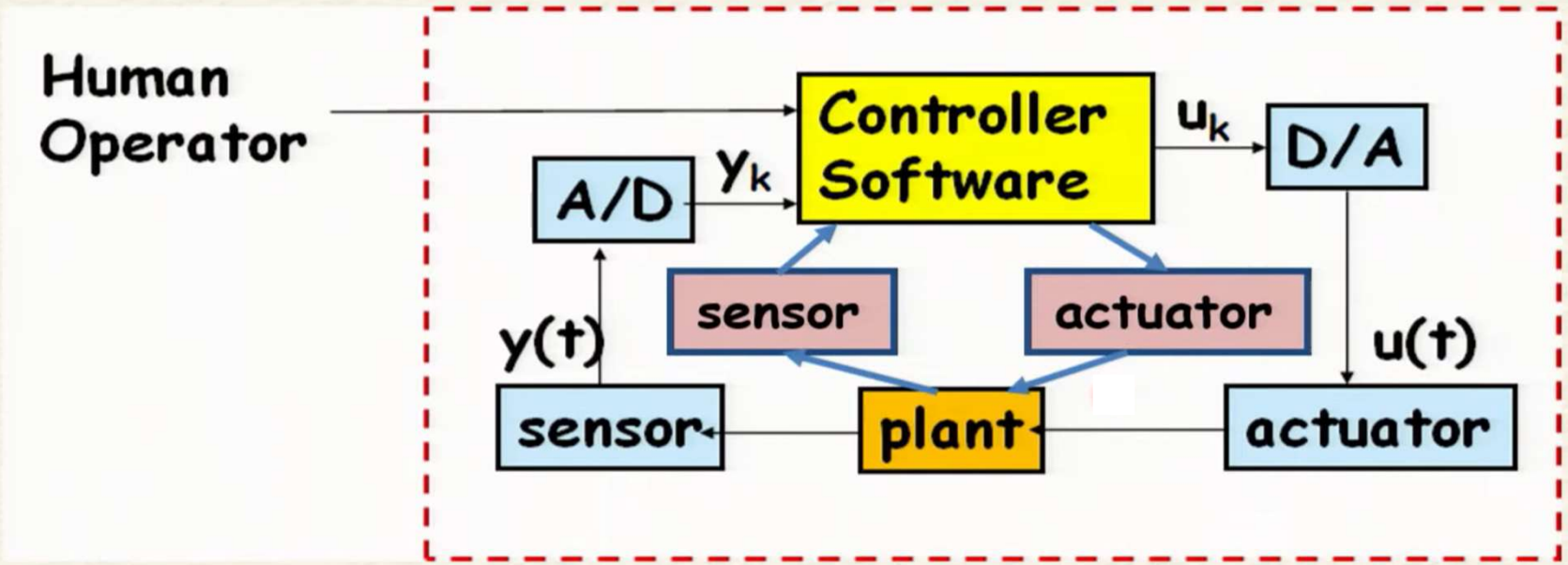
Many simple real-time systems are control systems



sampling period. T is a key design choice.

Multi-Rate Systems

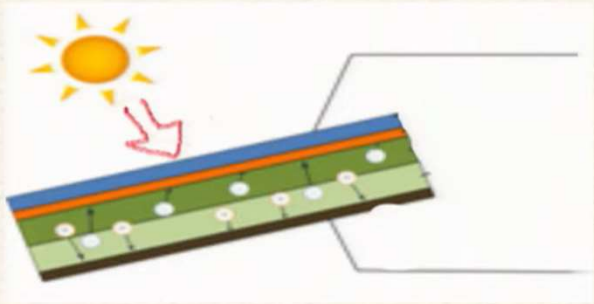
Support control loops of different rates.



Harmonic rates simplifies design of the system

Sensors

- A sensor converts some physical characteristic of its environment:
  - Into electrical signals.



- Example sensors:
  - A photo-voltaic cell converts light energy into electrical energy.
  - A temperature sensor typically operates based on the principle of a thermocouple.
  - A pressure sensor typically operates based on the piezoelectricity principle.

Actuators

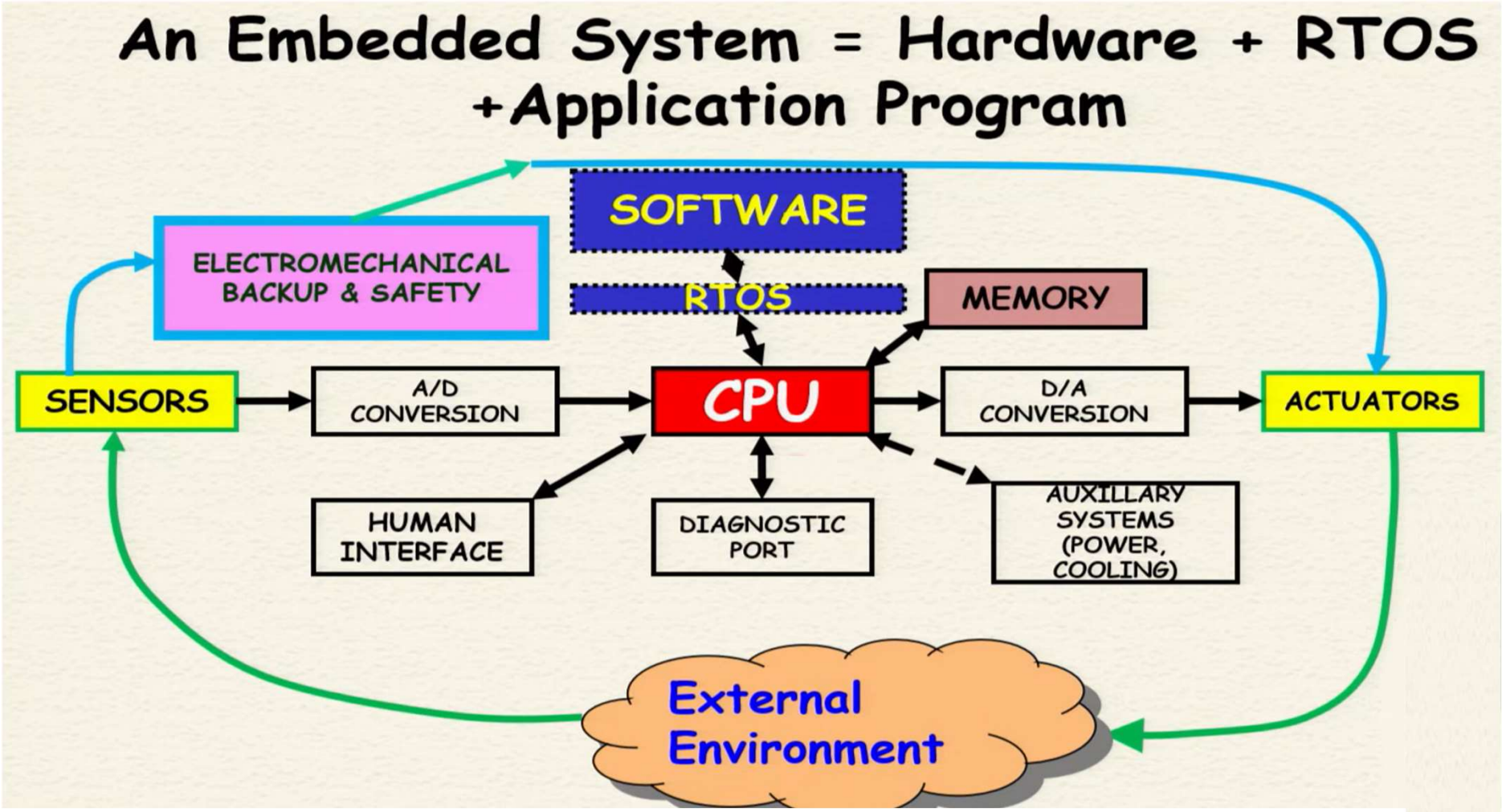
- An actuator converts electrical signals from a computer into some physical actions.
- The physical actions may be:
  - Motion, change of thermal, electrical, pneumatic, or physical characteristics of some objects.
- **Example actuators:**
  - Motors
  - Heaters
  - Hydraulic and pneumatic actuators

Stepper Motor

Signal Conditioning

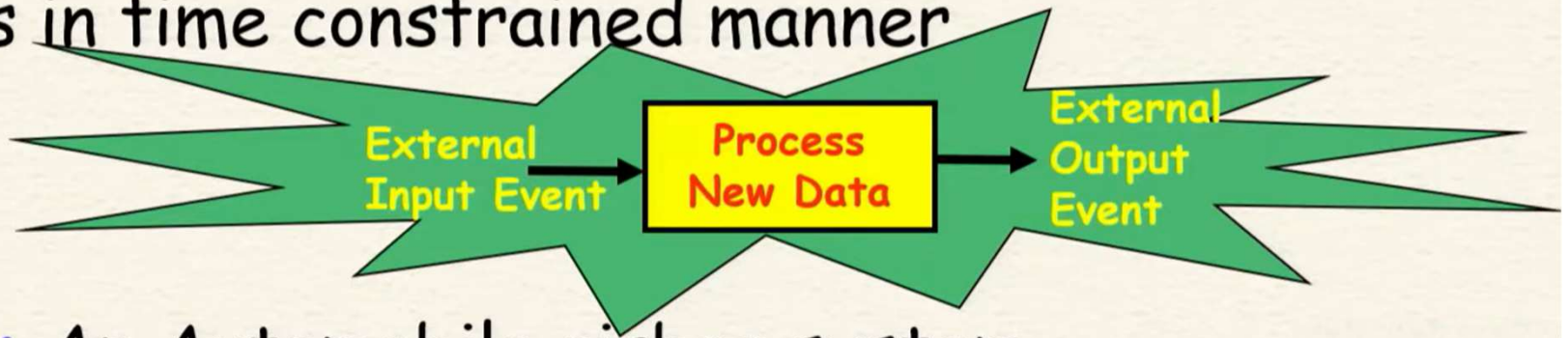
- Sensors generate analog signals:
  - A photo-voltaic cell normally generates signals in the millivolts range.
  - Need to be conditioned before they can be processed by a computer.
- Important types of conditioning:
  - Voltage Amplification
  - Voltage Level Shifting
  - Frequency Range Shifting and Filtering
  - Signal Mode Conversion

An Embedded System = Hardware + RTOS +Application Program



Important Characteristics of Real-Time Systems

- Most important: An embedded system responds to events in time constrained manner



Example: An Automobile airbag system.

When the airbag's motion sensors detect a collision, the system needs to respond by deploying the airbag within 10ms or less.

– or the system fails!

Characteristics of A Real-Time System…

- **Timing Constraints:**
  - Some tasks are real-time, not necessarily all tasks
  - Each real-time task is associated with some time constraints, e.g. a Deadline.
- **New Correctness Criterion:**
  - Results should be logically correct,
  - **And within the stipulated time.**

Characteristics of A Real-Time System…

- **Safety and Task Criticality:**
  - A critical task is one whose failure causes system failure (example: obstacle avoidance).
  - A safe system does not cause damage.
  - **A safety-critical real-time system is one where any failure causes severe damage.**

Characteristics of A Real-Time System

- **Concurrency:**
  - A RT system needs to respond to several independent events.
  - Typically for each independent event, separate tasks (processes or threads) are created.
  - For example, process coolant level, proces temperature and pressure, process user requests, etc.

Characteristics of A Real-Time System

- **Reactive:**
  - On-going interaction between computer and environment.
- **Stability:**
  - Under overload conditions, at least the critical tasks should continue to meet deadlines.
- **Exception Handling:**

Safety and Reliability

- A safe system:
  - Does not cause damage even when it fails.
- A reliable system:
  - Operates for long time without any failure
- Independent concepts in traditional systems.

Safety and Reliability

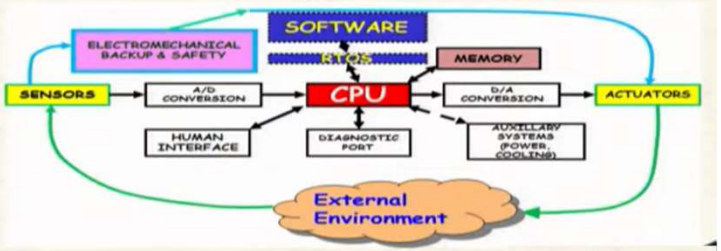
- In traditional systems:
  - Safety and reliability are independent concerns.
  - A system can be safe and unreliable and vice versa.
  - Give examples of:
    - A safe and unreliable system
    - A reliable and unsafe system

Safety and Reliability

- These two concepts are interrelated in safety-critical system.
  - **A safety critical system is one for which any failure of the system would result in severe damage.**
- Safety can be ensured only through increased reliability.

Safety and Reliability

- An unreliable system can be made safe:
  - When a failure occurs, by reverting to a fail-safe state.



- **A fail-safe state:**
  - No damage can result if a system fails in this state.
  - **Example:** For a traffic light, all lights orange and blinking.

Safety Critical Systems

- A safety-critical system:
  - One for which a failure can cause severe damages.
- **A safety-critical system does not have any fail-safe states:**
  - Safety can only be ensured through increased reliability.

Safety and Reliability

- What is the fail-safe state of a word processing program?
  - The document being processed has been saved onto the disk.
- **Fail-safe states help separate the issues of safety and reliability.**

Safety and Reliability

- For a safety-critical system
  - No fail-safe state exists.
- Consider the fly-by-wire aircraft:
    - When the on-board computer system fails
    - Shutting down the engine can be of little help!
  - As a result, for a safety-critical system:

How to Design a Highly Reliable System?

- Error Avoidance
- Error Detection and Removing
- Fault Tolerance

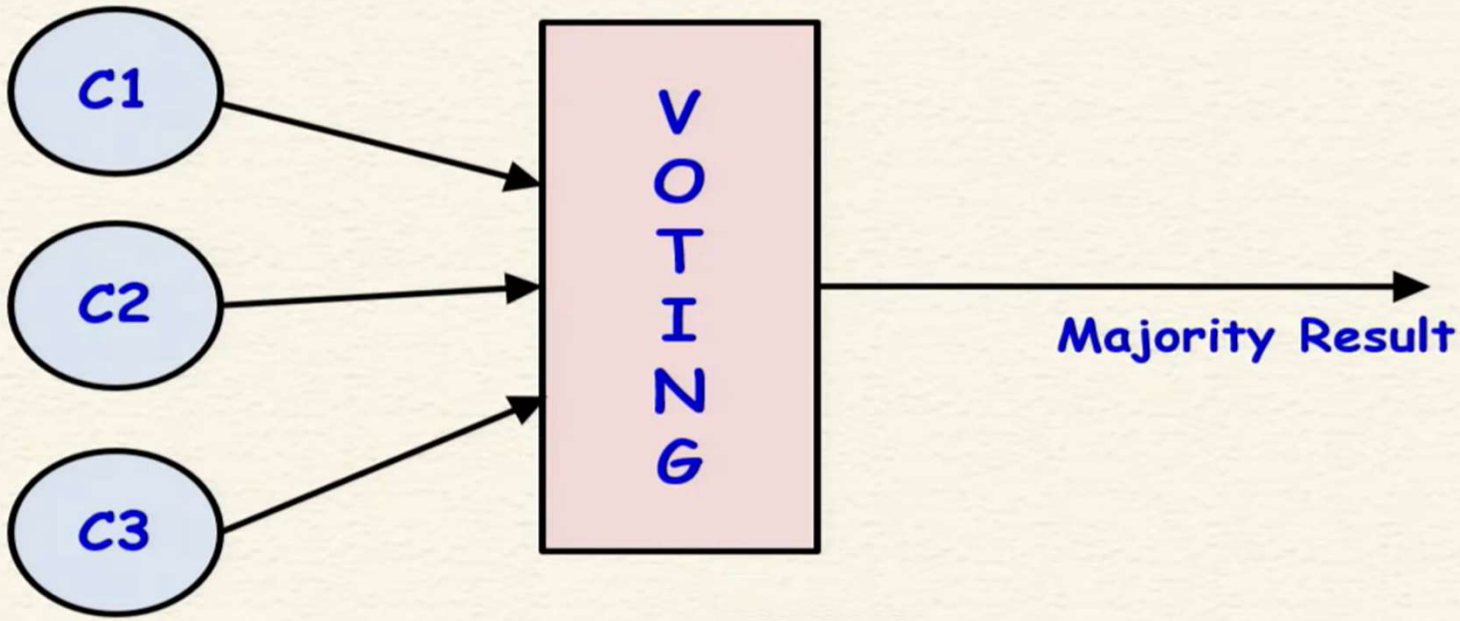
Fault Tolerance in RT System

- The essential idea:
  - **Provide redundancy**
- **Hardware Fault-Tolerance:**
  - Masks the effects of a hardware fault.
- **Software Fault-Tolerance:**
  - Masks the effects of a program fault.

Fault Tolerance in RT System

- **Hardware FT:**
  - Built in self test (BIST)
  - Triple modular redundancy
- **Software FT:**
  - N-Version programming
  - Recovery Blocks

Triple Modular Redundancy



C1, C2 and C3 are redundant copies of a component

N-version Programming

- This software fault tolerance technique is inspired by TMR of hardware.
- Different teams are employed to develop the same software.
- Unsatisfactory performance in practice.
- Reason: Faults are correlated in the different versions.

N-version Programming

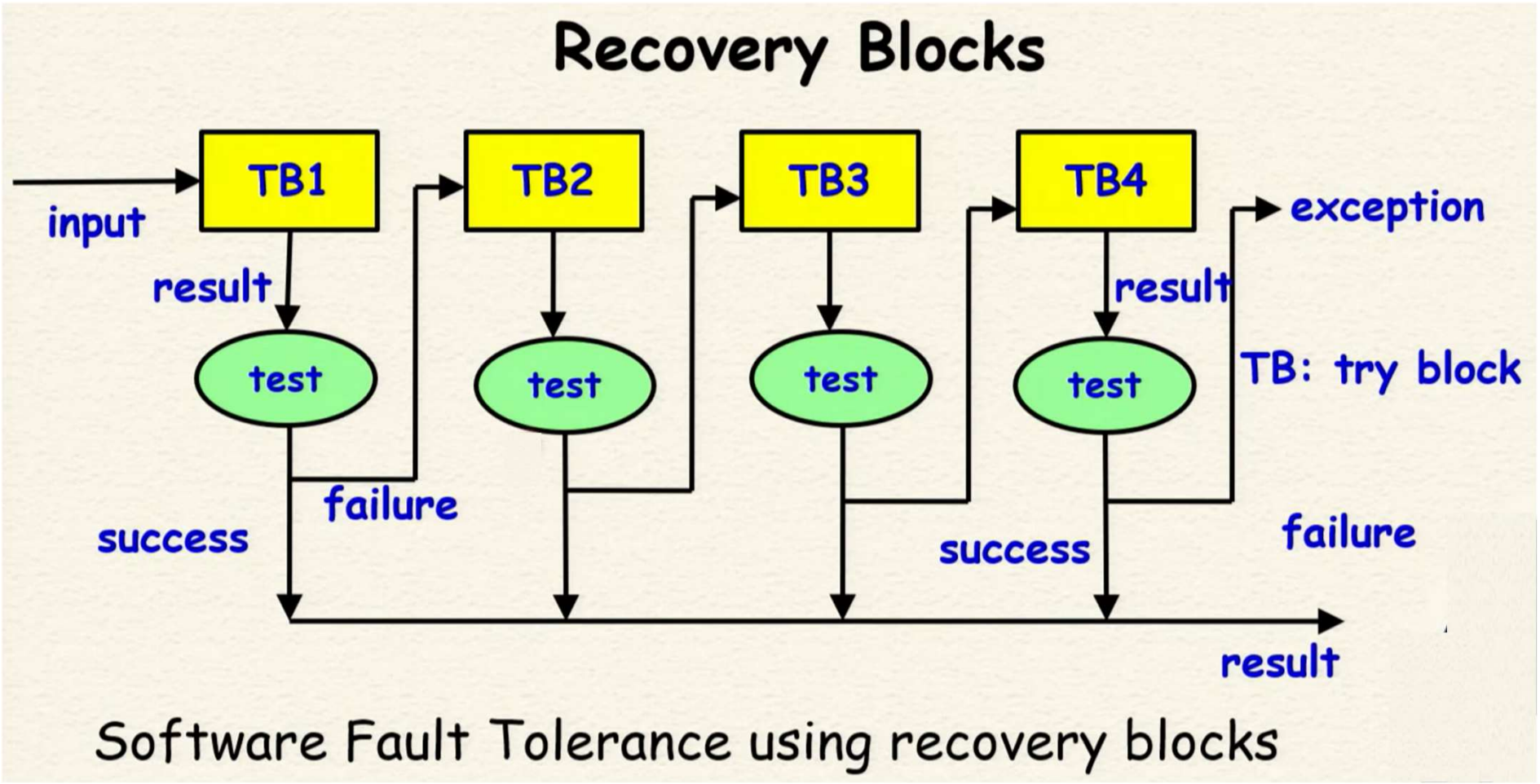
- This software fault tolerance technique is inspired by TMR of hardware.
- Different teams are employed to develop the same software.
- Unsatisfactory performance in practice.
- Reason: Faults are correlated in the different versions.
- All versions fail for similar reasons.

Recovery Block: Essential Idea

- Suppose we can check the result of a software module by subjecting it to an acceptance test

```
Execute primary module
Try acceptance test
If fails.     try first alternate
  ...              ...

else try      last alternate
try final acceptance test
```



Software Fault Tolerance using recovery blocks

Role of RTOS in an Embedded System

- An embedded system responds to external inputs:
  - If response is late, a hard real-time system fails.
- **Real-time OS:**
  - Helps tasks meet their deadline.
  - Should be easily installable on embedded devices.

Why Have an OS in an Embedded Device?

- Can avail support for:
  - Multitasking, scheduling, and synchronization
  - Meeting timing constraints.
  - Memory management
  - File systems
  - Networking
  - Graphics displays
  - Interfacing wide range of I/O devices
  - Scheduling and buffering of I/O operations
  - Security and power Management

Need to have RTOS on an Embedded Device…

- **Example:** A recent cell phone operating system contained over five million lines of code!
- Few, if any projects will have the time and funding:
  - To develop all of this code on their own!
- Typical Embedded RTOS license fees are a few dollars per device --- less than a desktop OS
- Some very simple low-end devices might not need an RTOS:

Need to have RTOS on an Embedded Device…

- **Example:** *A recent cell phone operating system contained over five million lines of code!*
- Few, if any projects will have the time and funding:
  - To develop all of this code on their own!
- Typical Embedded RTOS license fees are a few dollars per device --- less than a desktop OS
- Some very simple low-end devices might not need an RTOS:
  - But new devices are getting more complex.

Types of Real-Time Tasks

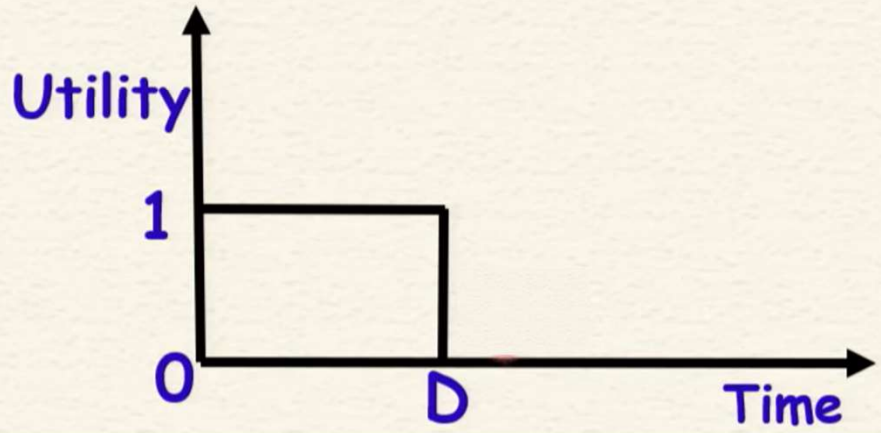
- Real-time Tasks are different from traditional tasks:
  - **Real-time tasks have deadlines associated with them.**
- Classified based on consequence of a failure:
  - **Hard** real-time tasks
  - **Soft** real-time tasks
  - **Firm** real-time tasks

Hard Real-Time Tasks

- If a deadline is not met:
  - **The system fails.**
- The task deadlines are of the order of micro or milliseconds.
- Many hard real-time systems are **safety-critical.**
- **Examples:**
  - Industrial control applications
  - On-board computers
  - Robots

Firm Real-Time Tasks

- If a deadline is missed occasionally, the system does not fail:
  - The results produced by a firm real-time task after its deadline are rejected.



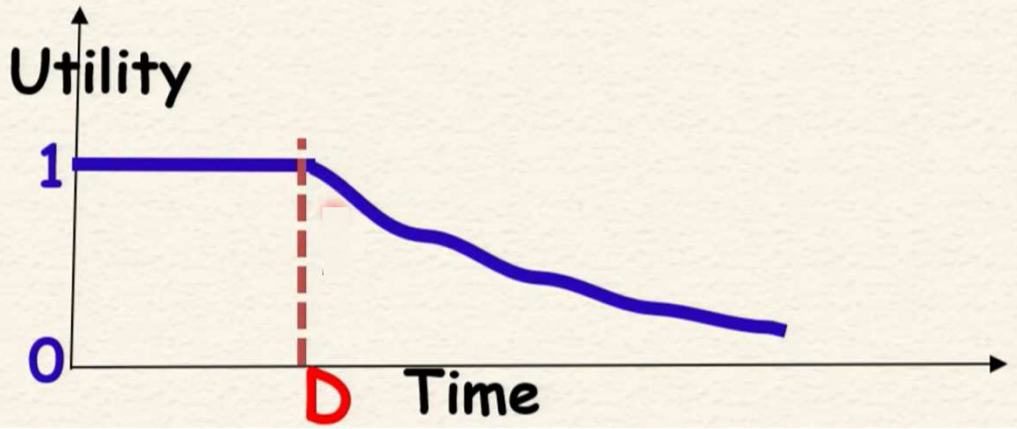
Firm Real-Time System: Examples

- A video conferencing application

- A telemetry application
- Satellite-based surveillance applications

Soft Real-Time Tasks

- If a deadline is missed, the system does not fail:
  - Only the performance of the system is said to have degraded.
  - The utility of results decrease rapidly with time after the deadline.

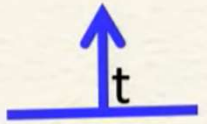


Examples of Soft Real-Time Tasks

- Railway seat reservation task
- Web browsing
- In fact, all interactive applications

Timing Constraints

- A timing constraint:
  - Defined with respect to some event.
- **An event:**
  - Can occur at an instant of time



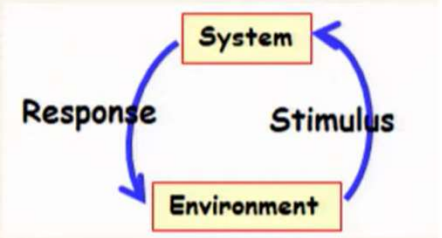
  - May also occur over a duration

  - Generated either by the system or its environment

Events in a Real-Time System

- Events in a real-time system can be classified into:
  - **Stimulus Events**
  - **Response Events**

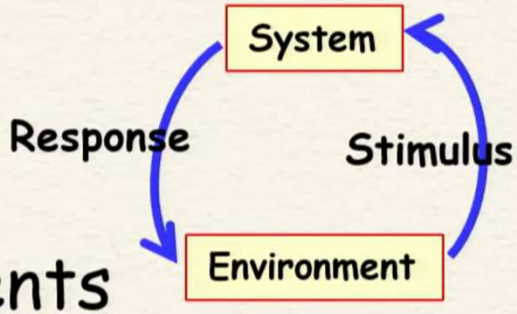
Stimulus Event



- **Generated by the environment:**
  - Act on the system.
- Typically asynchronous in nature:
  - Aperiodic
  - Can also be periodic

- **Asynchronous Stimulus Example:**
  - A user pressing a button on a telephone set
  - Stimulus event acts on the telephone system.

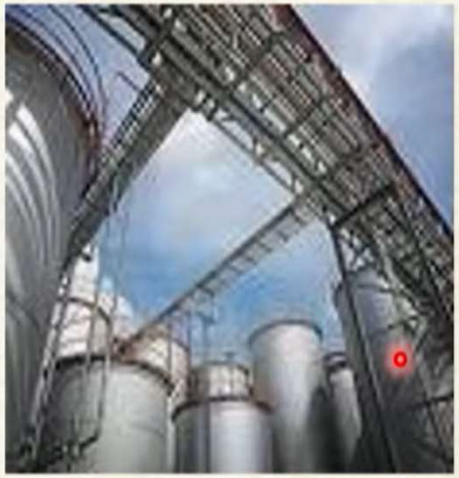
Response Event



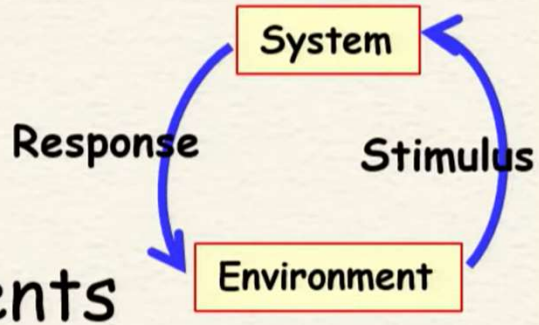
- **Produced by the system:**
  - In response to some stimulus events
- **Example:**
  - In a chemical plant as soon as the temperature exceeds 100°C,
  - The system responds by switching off the heater within 1 Sec.

Periodic Stimulus Event Example

- Periodic sensing of temperature in a chemical plant.



Response Event

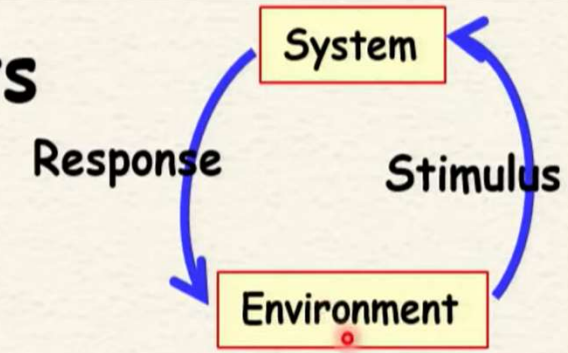


- **Produced by the system:**
  - In response to some stimulus events
- **Example:**
  - In a chemical plant as soon as the temperature exceeds 100°C,
  - The system responds by switching off the heater within 1 Sec.

Classification of Timing Constraints

- Classification of the timing constraints in a system:
  - Can help us to quickly identify and understand these from a problem description.
- Different timing constraints can broadly be classified into:
  - **Performance constraints**
  - **Behavioral constraints**

Types of Timing Constraints



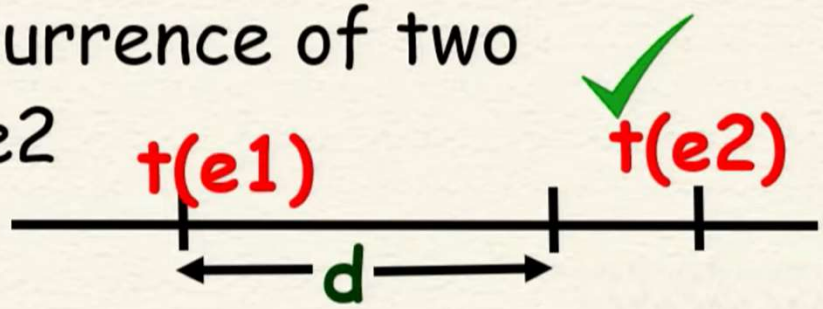
- **Performance constraints:**
  - Imposed on the response of the system.
- **Behavioral constraints:**
  - Imposed on the stimuli generated by the environment.

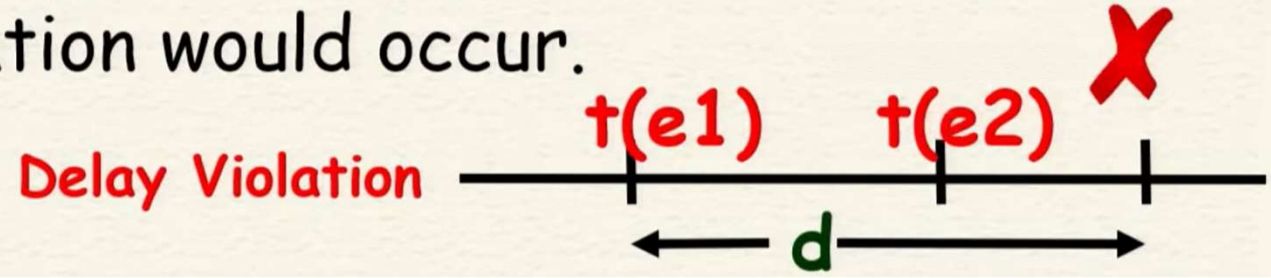
Types of Timing Constraints

- Both performance and behaviorial constraints can be classified into:
  - **Delay Constraints**
  - **Deadline Constraints**
  - **Duration Constraints**

Delay Constraint

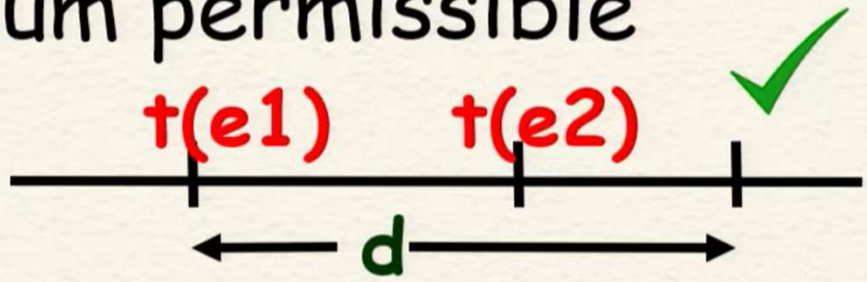
- Expresses minimum time delay d:
  - Needed between the occurrence of two arbitrary events e1 and e2
  - **t(e2) - t(e1) ≥ d**
  - if e2 occurs earlier than d then a delay violation would occur.





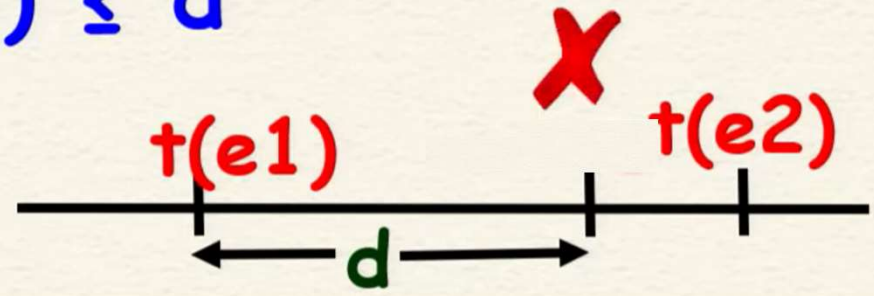
Deadline Constraint

- Expresses the maximum permissible separation:



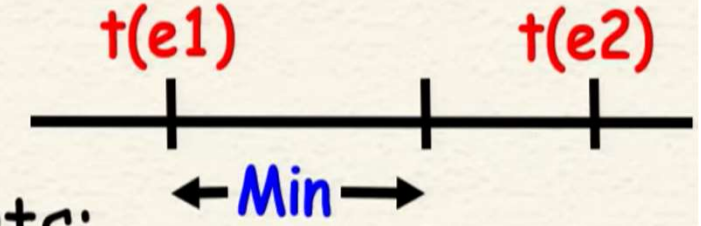
  - Between any two arbitrary events.
  - $t(e2) - t(e1) \leq d$

Deadline Violation



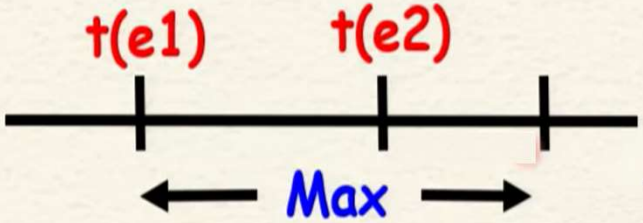
Duration Constraints

- **Minimum:**



  - Once a duration event starts:
    - It must not end before a certain minimum time.

- **Maximum:**

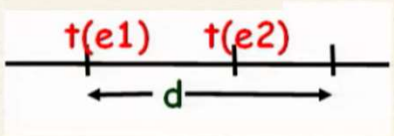


  - Once a duration event starts:

SS Deadline Example

- Deadline is defined between two stimuli.
  - A behavioral constraint.
  - Imposed on stimulus.

- Once a user completes dialling a digit,
  - He must dial the next digit within the next 5 seconds.

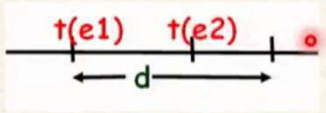


  - Otherwise an idle tone is produced.

RS Deadline Example

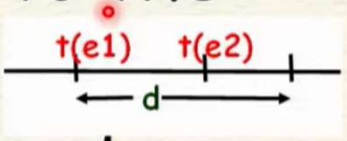
- Deadline is defined on the stimulus from the respective response event.
  - A behaviorial constraint.
  - Imposed on stimulus.

- Once the dial tone appears:
  - The first digit must be dialled within 30 seconds,
  - Otherwise the system enters an idle state and an idle tone is produced



RR Deadline Example

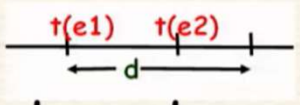
- Deadline is defined on the response time from another response.
- A performance constraint.
  - Imposed on response.
- **Example:**
  - Once ring tone is given to the callee,
  - Ring back tone must be given to the caller within two seconds,
  - Otherwise the call is terminated.



SR Deadline Example

- Deadline is defined on the response from the respective stimulus.
  - A performance constraint.
  - Imposed on response.

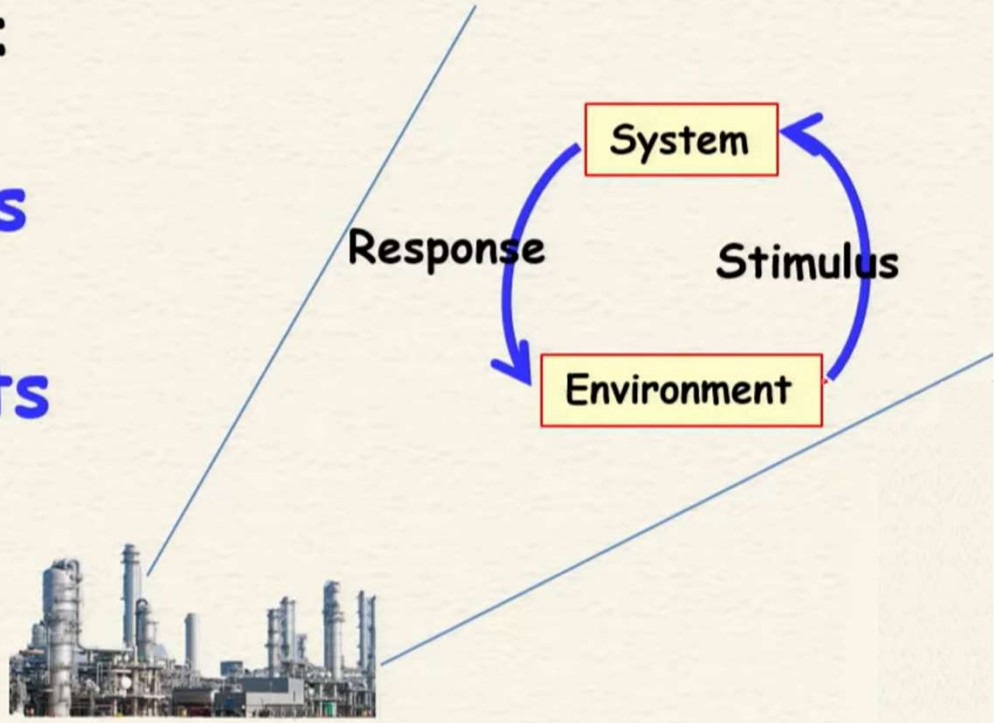
- **Example:**
  - Once the receiver of the hand set is lifted:
  - The dial tone must be produced by the system within 2 seconds,



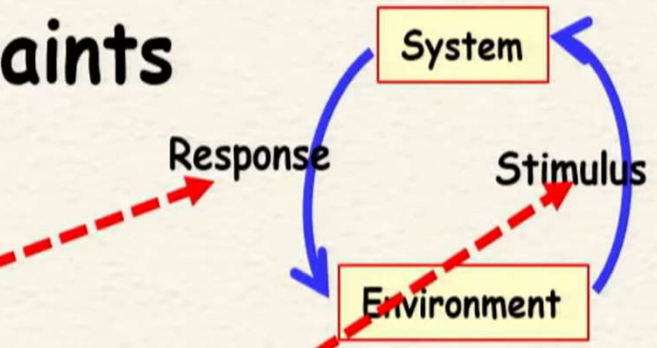
  - Otherwise a beeping sound is produced until the handset is replaced.

Events in a Real-Time System

- Events in a real-time system can be classified into two main types:
  - **Stimulus Events**
  - **Response Events**



Types of Timing Constraints



- **Performance constraints:**
  - Imposed on the response of the system.
- **Behavioral constraints:**
  - Imposed on the stimuli generated by the environment.

Further Classification of Timing Constraints

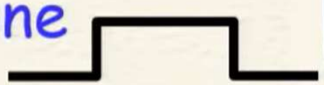
- Each of performance and behaviorial constraints can be classified into 3 types:
  - Delay Constraints
  - Deadline Constraints
  - Duration Constraints

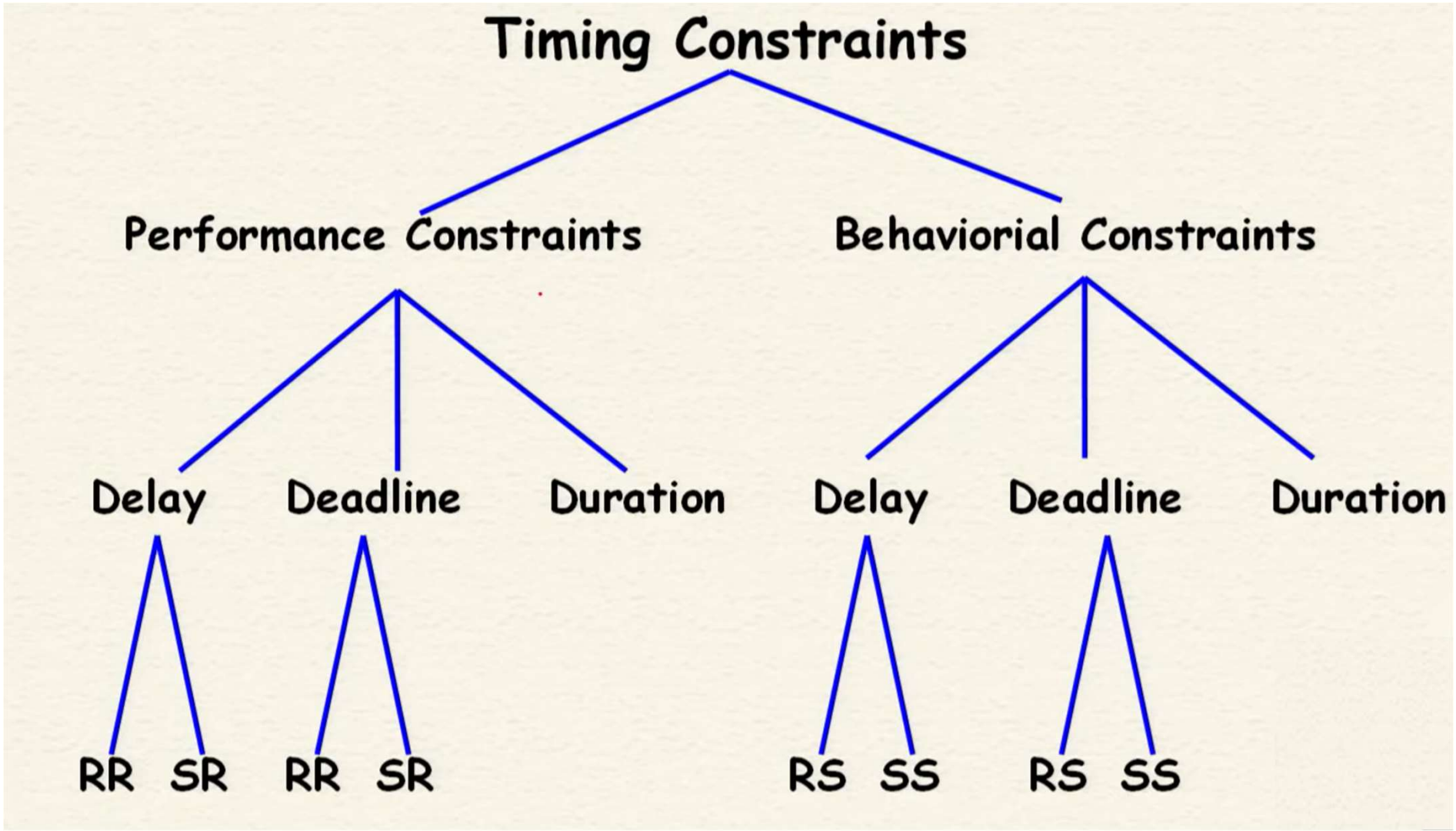
Duration Constraint Example

- Used to specify the time interval over which an event acts.
- If you press the button of the handset for less than 15 seconds,

  - It connects to the local operator.
- If you press the button for any duration lasting between 15 to 30 seconds,

  - It connects to the international operator.
- If you keep the button pressed for more than 30 sec
  - Then on releasing, it produces dial tone




Timing Constraints
Performance Constraints
Behaviorial Constraints
Delay
Deadline
Duration
Delay
Deadline
Duration
RR
SR
RR
SR
RS
SS
RS
SS

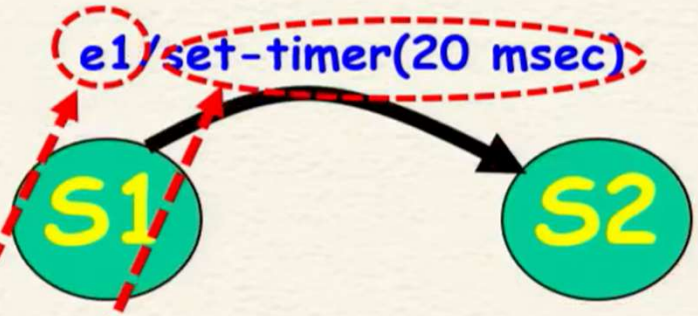
Why Model Timing Constraints?

- Modelling time constraints in a system:
  - Can serve as a specification of the system.
  - Can be used to automatically generate code.
  - Can help to understand real-time behavior.
  - Can help to design test cases

Modelling Time Constraints

- Several approaches can be used.
- We discuss an approach based on FSM proposed by Dasarathy (IEEE TSE, 1985).
- A state is defined in terms of the values assumed by some attributes.
- The states of an elevator may be denoted in terms of its directions of motion.
- Values of the attribute "direction" define the states up, down, and stationery.

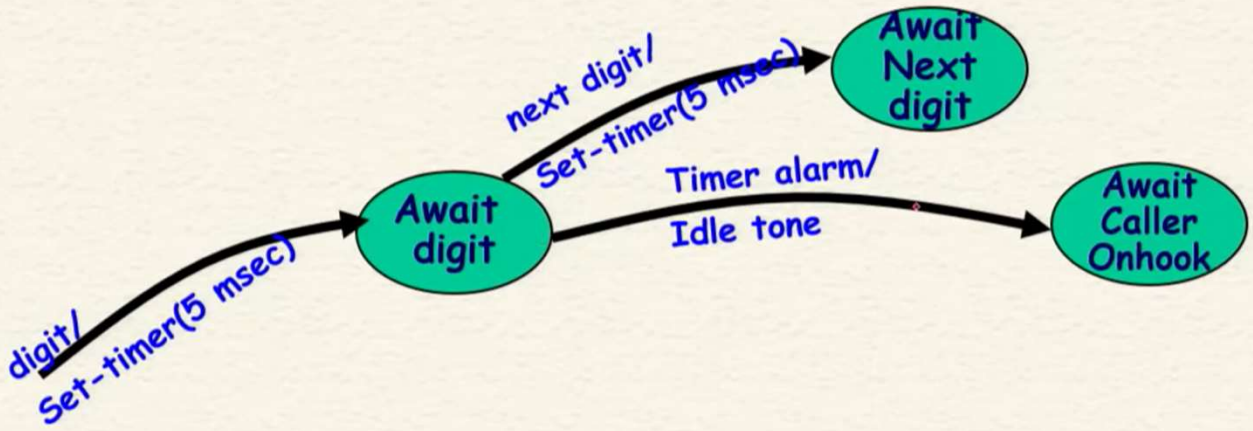
Basic FSM Notation Used



- A transition is annotated with:
  - Enabling event
  - Action that would takes place during transition

Model of An SS Deadline Constraint Example

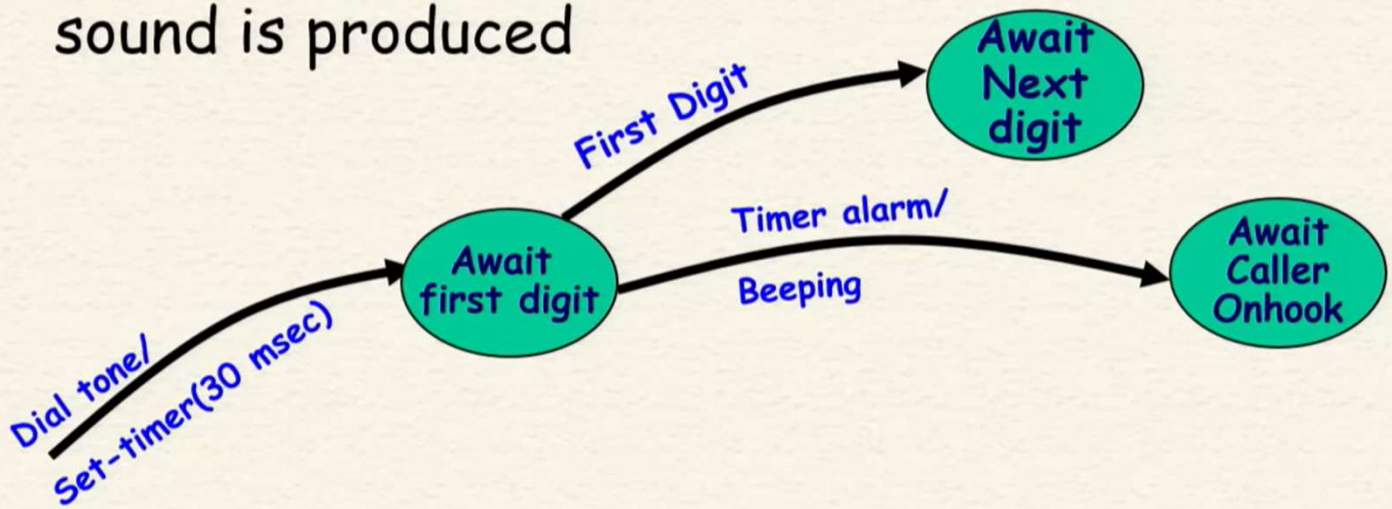
- Once a user completes dialling a digit,
  - He must dial the next digit within the next 5 seconds.
  - Otherwise an idle tone is produced.



Model of An RS Deadline Constraint Example

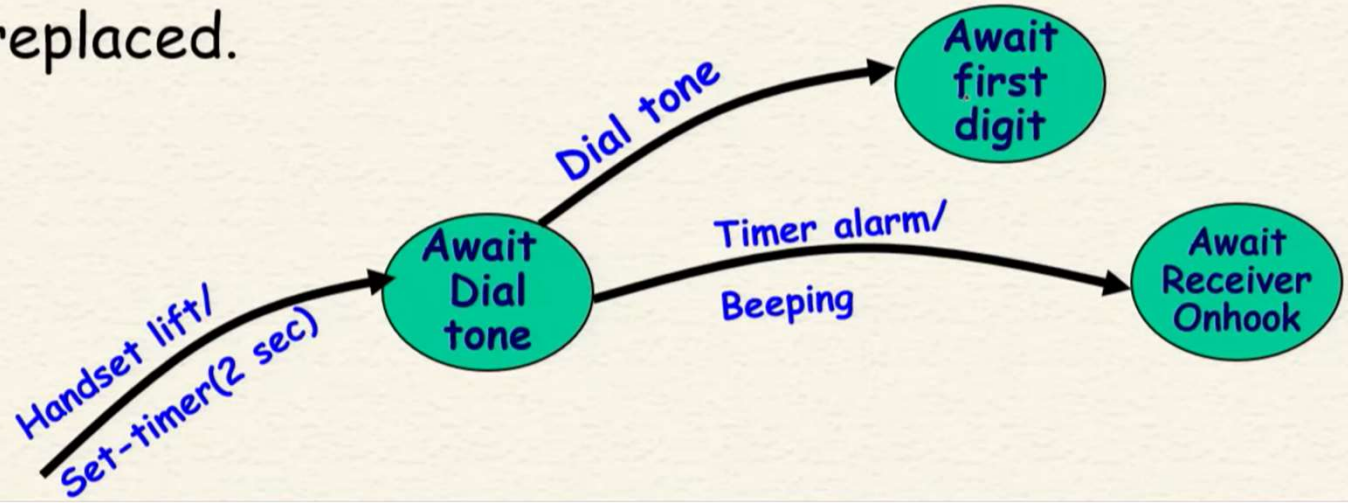
Once the dial tone appears:

- The first digit must be dialed within 30 seconds,
- Otherwise the system enters an idle state and a beeping sound is produced



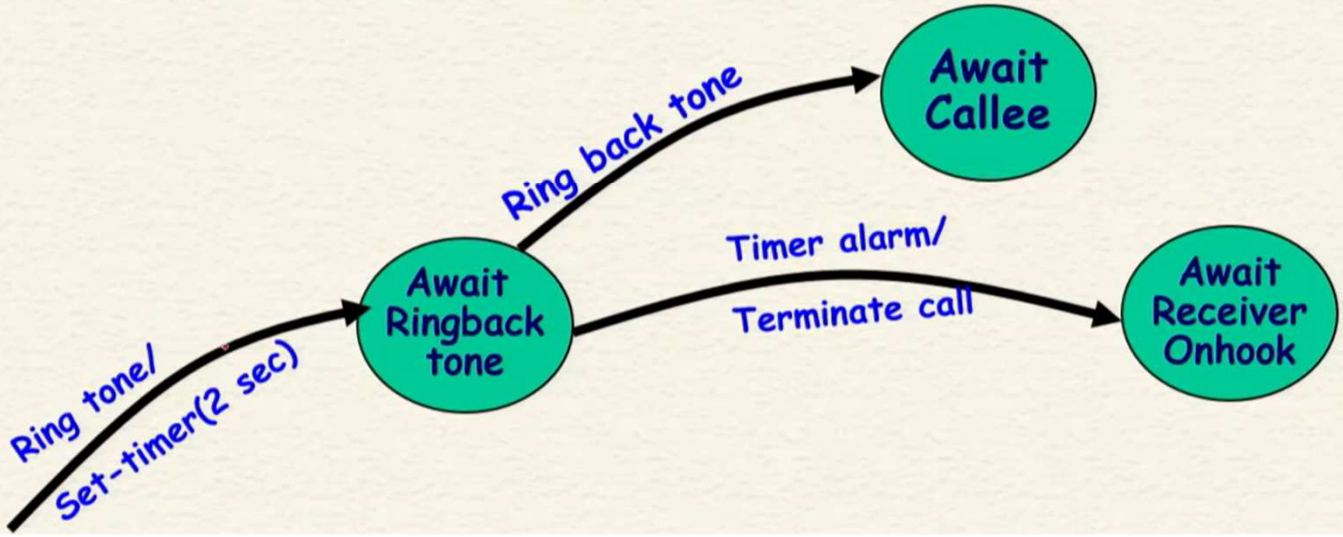
Modelling an SR Deadline Constraint

- Once the receiver is lifted from the hand set:
- The dial tone must be produced by the system within 2 seconds,
- Otherwise a beeping sound is produced until the handset is replaced.



Modelling an RR Deadline Constraint

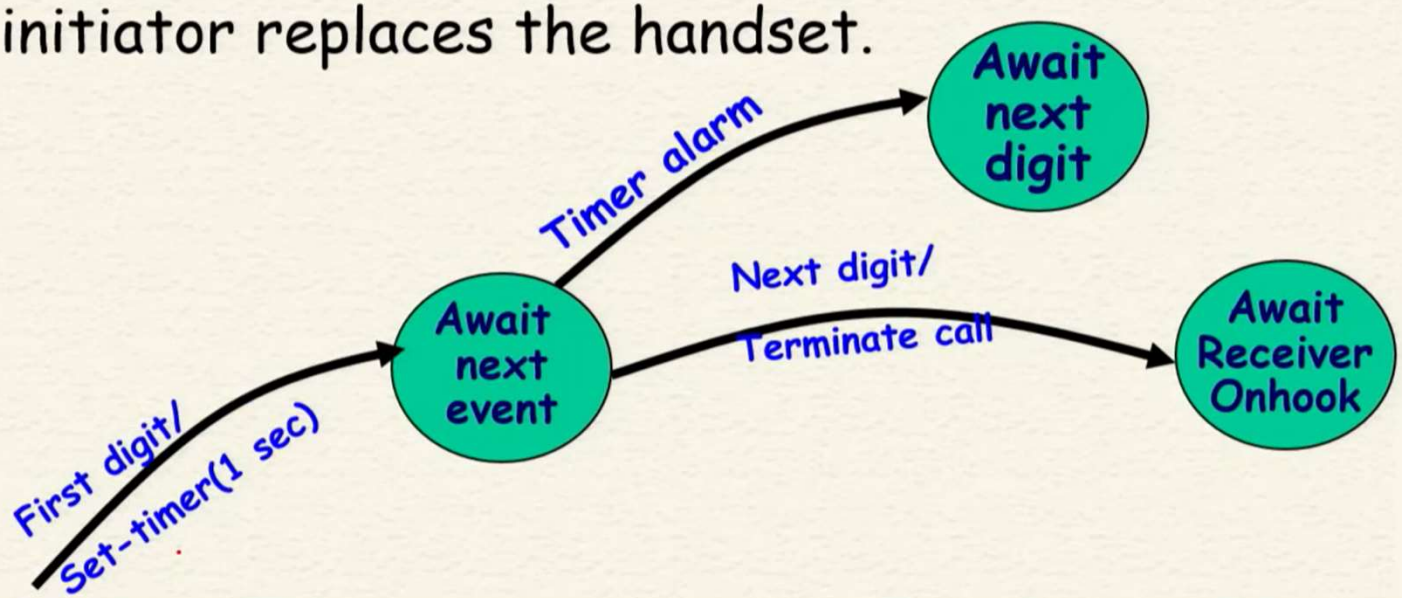
- Once ring tone is given to the callee,
  - Ring back tone must be given to the caller within two seconds,
  - Otherwise the call is terminated.



Modelling a Delay Constraint

Once a digit is dialled:

- The next digit should be dialled after at least 1 second.
- Otherwise, a beeping sound is produced until the call initiator replaces the handset.

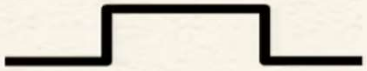


Duration Constraint Example

- If you press the button of the handset for less than 15 seconds,

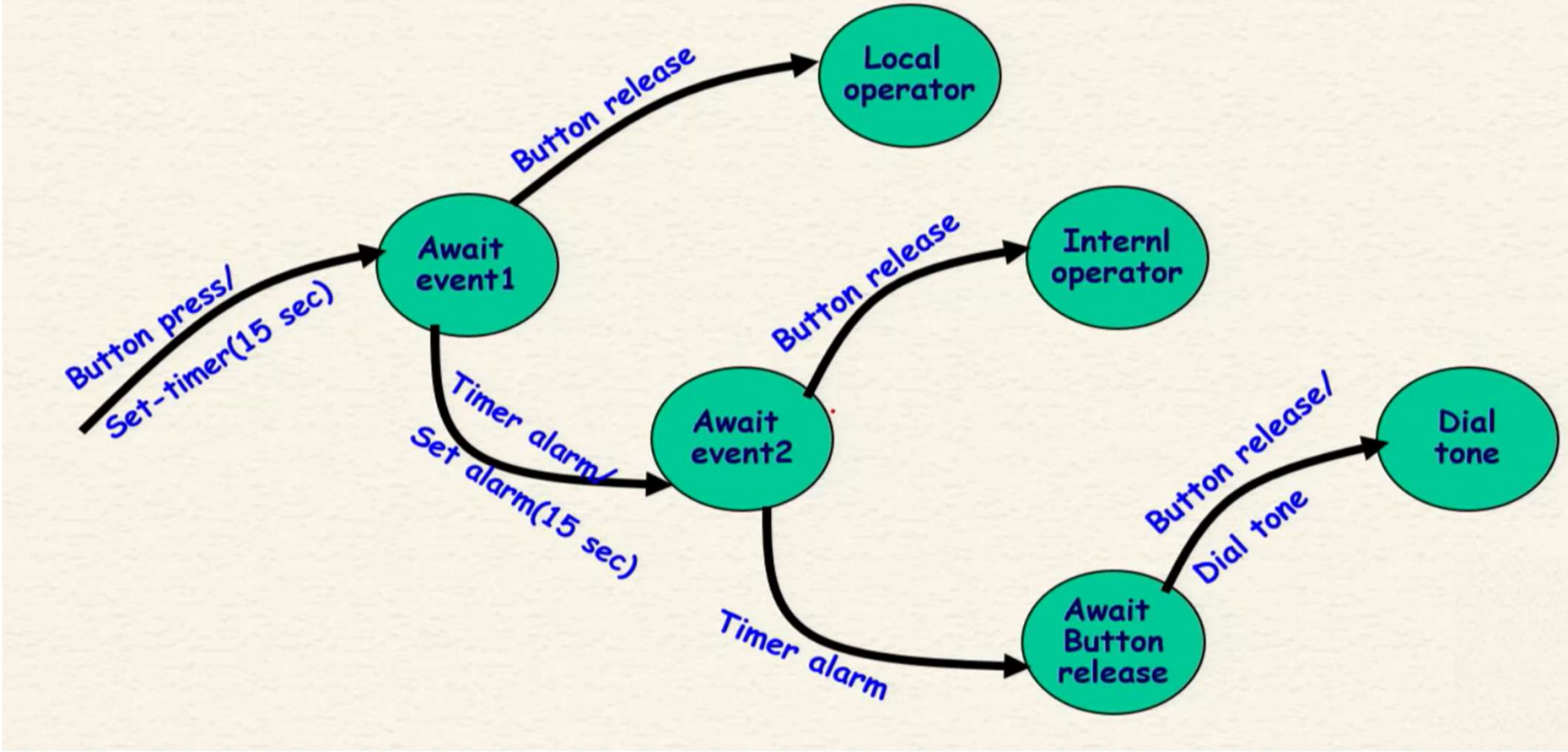
  - It connects to the local operator.
- If you press the button for any duration lasting between 15 to 30 seconds,

  - It connects to the international operator.
- If you keep the button pressed for more than 30 seconds,



  - Then on releasing it produces dial tone.

Model of Duration Constraint Example



Basics of Real-Time Task Scheduling

Introduction

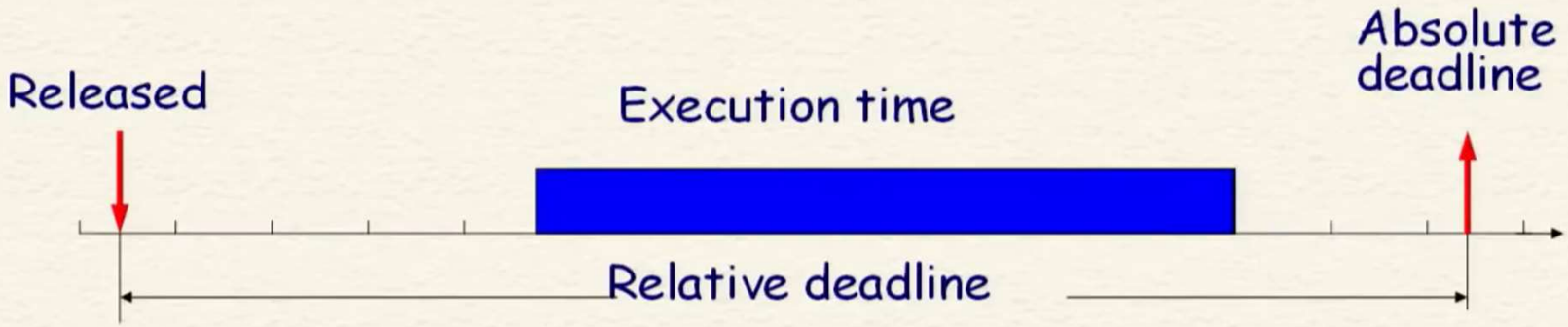
- Real-time tasks get generated due to certain event occurrences:
  - Either internal(system) or external(environment) events.
- **Example:**
  - A task may get generated due to a temperature sensor sensing high-level.
  - When a task gets generated: It is said to be released or to have arrived .

Real-Time Task Scheduling

- Essentially refers to the order in which the various tasks are to be executed.
- It is the primary means adopted by an operating system to meet task deadlines.
- Obviously, scheduler is a very important component of RTOS.

Real-Time Workload

- **Job:**
  - A task instance
  - A unit of work
  - A computation, a file read, a message transmission, etc
- **Task:** a sequence of similar jobs



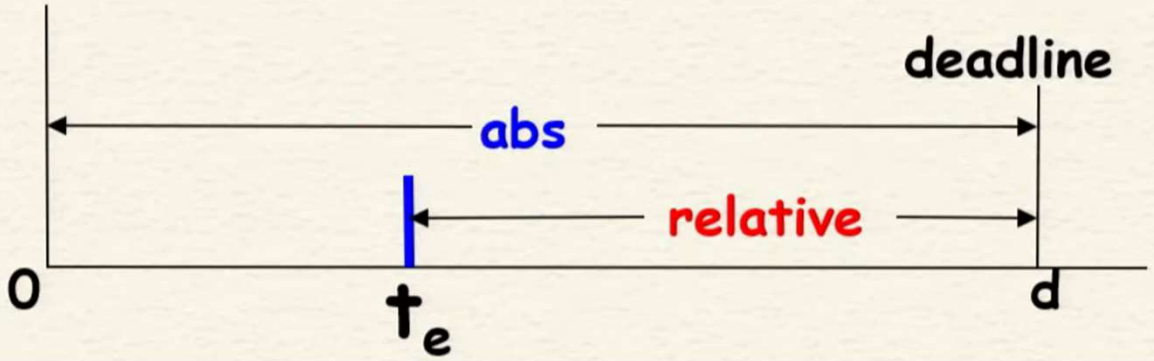
Task Instance (Job)

- A task typically recurs a large number of times:

  - Each time triggered by an event
  - Each time a task recurs, an instance of the task is said to have been generated.
- The ith time a task T recurs:
  - Task instance (job) Ti is said to have arrived or released.

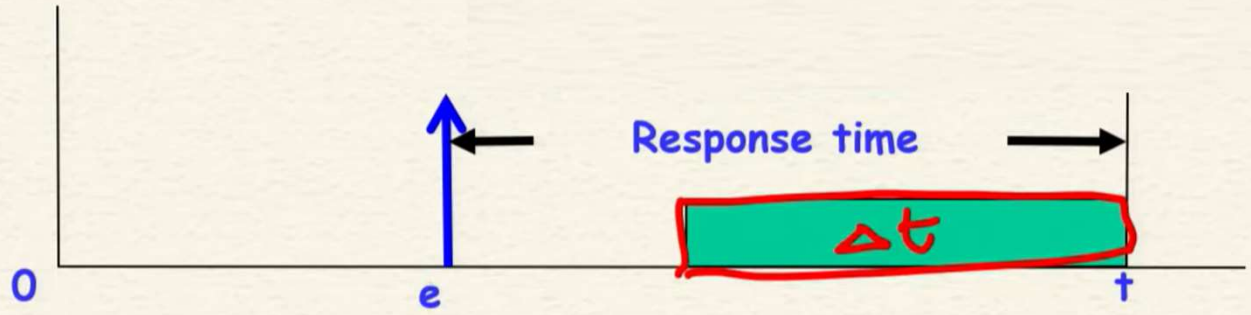
Relative and Absolute Deadlines

- **Absolute deadline:**
  - Counted from time 0.
- **Relative deadline:**
  - Counted from time of occurrence of task.



Response Time

- It is the time between release time and completion time.
- Release time
  - Time of occurrence of the event generating the task.
- Completion time
  - Time at which result is pro

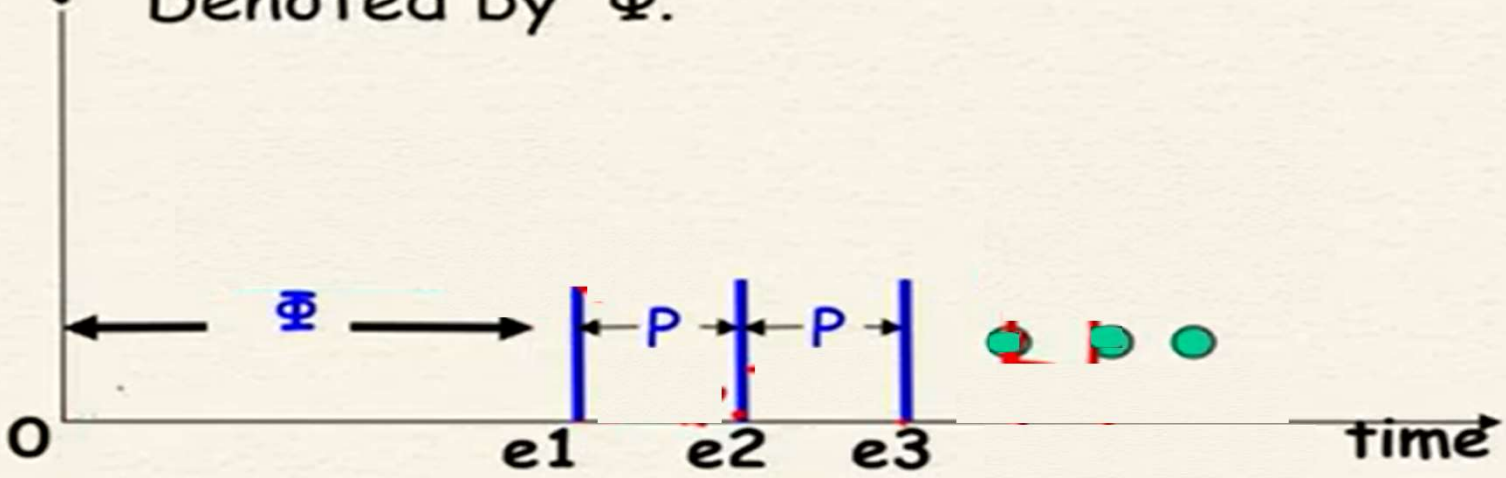


Response Time

- For soft real-time tasks:
  - The response time needs to be minimized.
- For hard real-time tasks:
  - As long as the task completes within its deadline. no advantage of completing it any early.

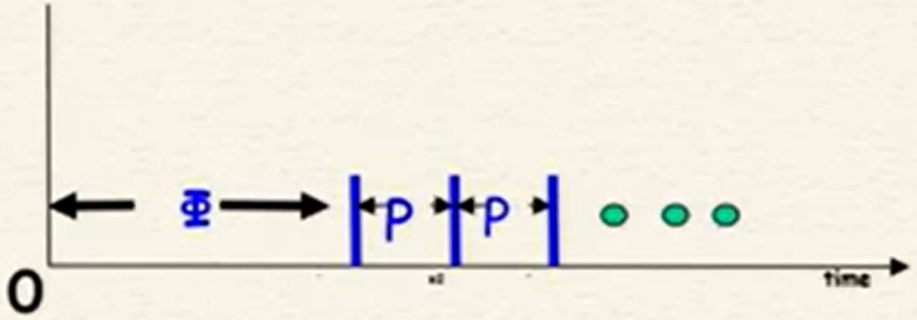
Phase of a Periodic Task

- **Phase for a periodic task:**
  - The time from 0 till the occurrence of the first instance of the task.
  - Denoted by Φ.



Phase Example

- The track correction task starts 2000 mSecs after the launch of the rocket:
- Periodically recurs every 50 milli Seconds then on.
- Each instance of the task requires a processing time of 8 mSecs and its relative deadline is 50 mSecs



A Few Important Scheduling Terminologies

- **Valid Schedule:**
  - At most one task is assigned to a processor at any time.
  - No task is scheduled before it is ready.
  - Precedence and resource constraints of all tasks are satisfied.

- **Feasible Schedule:**
  - A valid schedule in which all tasks meet their respective time constraints

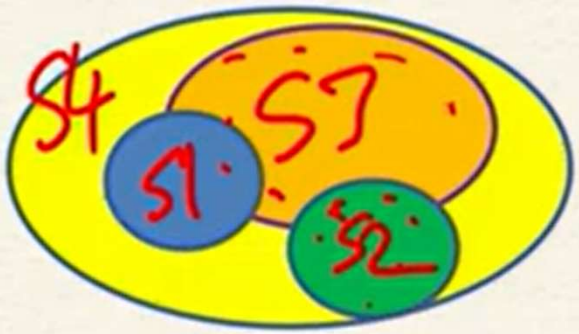
A Few Important Scheduling Terminologies

- **Proficient Scheduler:**
  - A scheduler S1 is as proficient as another Scheduler S2:
  - If whichever task sets that S2 can feasibly schedule so can S1, but not vice versa.

- **Equally proficient schedulers:**
  - If a task set scheduled by one can also be scheduled by the other and vice versa

A Few Important Scheduling Terminologies

- **Optimal Scheduler:**
  - An optimal scheduler can feasibly schedule any task set that can be scheduled by any other scheduler.



Scheduling Points

At these points on time line:

- Scheduler makes decision regarding which task to be run next.

Clock-driven:

- Scheduling points are defined by interrupts from a periodic timer.

Event-driven:

- Scheduling points defined by task completion and generation events