Unit-I.2

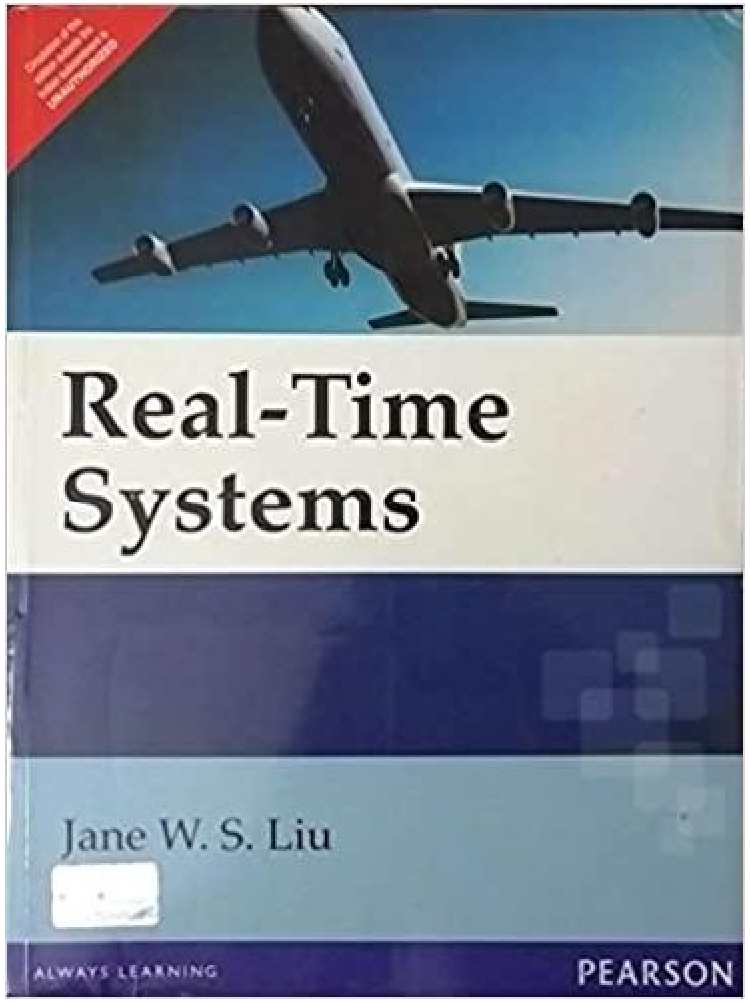
RTS- **Types of Real-Time Systems, characteristic of RTS,**

ACHIEVING HIGH RELIABILITY IN REAL-TIME SYSTEM

Prof Srinivas Prasad

Text Book

Jane W.S. Liu, Real Time System, Pearson Education Asia, 2001.



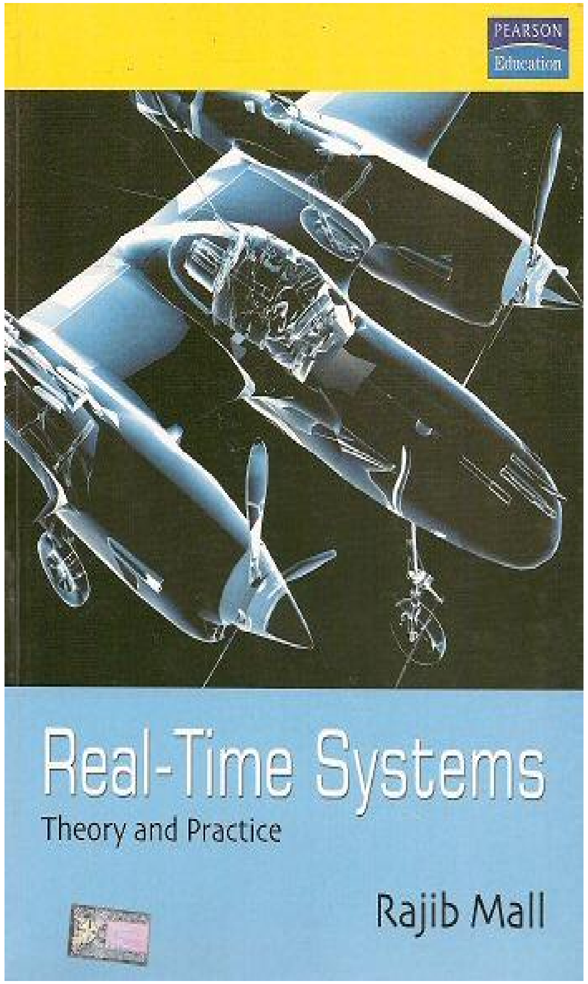
2

Reference Books

- **R. Mall, Real-Time Systems, Pearson, 2008.**

- C. Krishna and K. Shin, Real-TimeSystems, McGraw-Hill, 2000

- Raymond J.A. Buhr, Donald L.Bailey, An Introduction to Real Time Systems, Prentice Hall International,1999



3

Real-Time: Some Items and Terms

Task

- A sequential piece of code,program, perform service, functionality
- requires resources, e.g., execution time

Job

Instance of a task.

Jobs require resources to execute.

- Example resources: CPU, network, disk, critical section.
- We will simply call all hardware resources “processors”.

Deadline

- specified time for completion of, e.g., task
- time interval or absolute point in time
- value of result may depend on completion time

Types of Real-Time Systems

- Real-time systems are different from traditional systems:
  - Tasks have deadlines associated with them.
- Classified largely based on the consequence of not meeting deadline:
  - Hard real-time systems
  - Soft real-time systems
  - Firm real-time systems

5

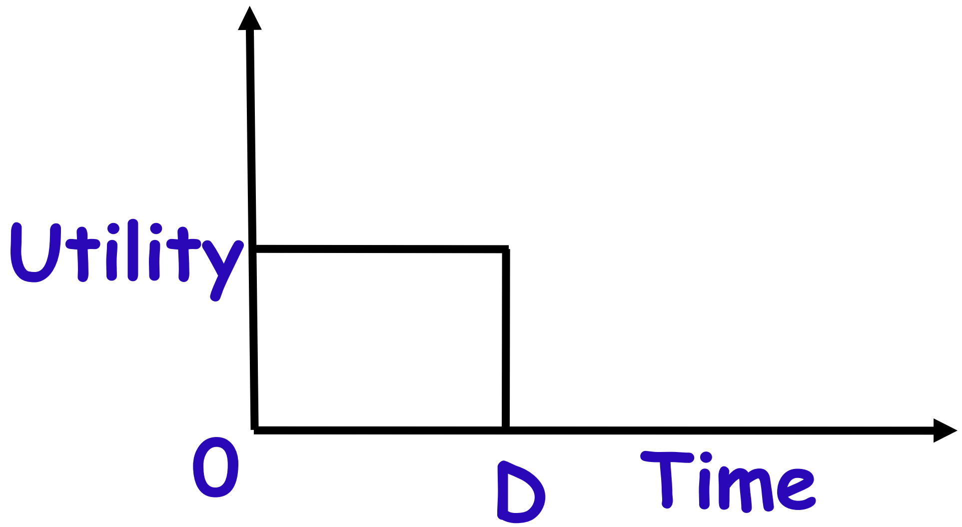
Hard Real-Time Systems

- **If a deadline is not met:**
  - **The system is said to have failed.**
- 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

6

Firm Real-Time Systems

- If a deadline is missed occasionally, the system does not fail:
  - The results produced by a task after the deadline are ignored.



7

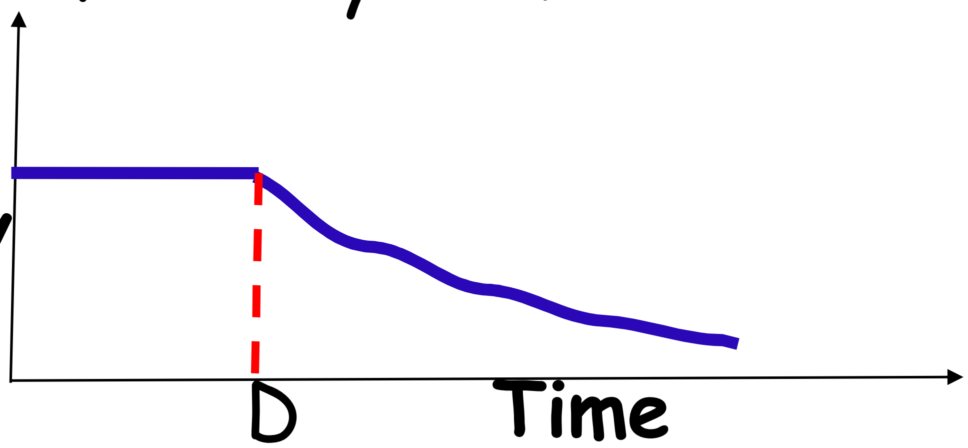
Firm Real-Time Systems

- Examples:
  - A video conferencing application
  - A telemetry application
  - Satellite-based surveillance applications

8

Soft Real-Time Systems

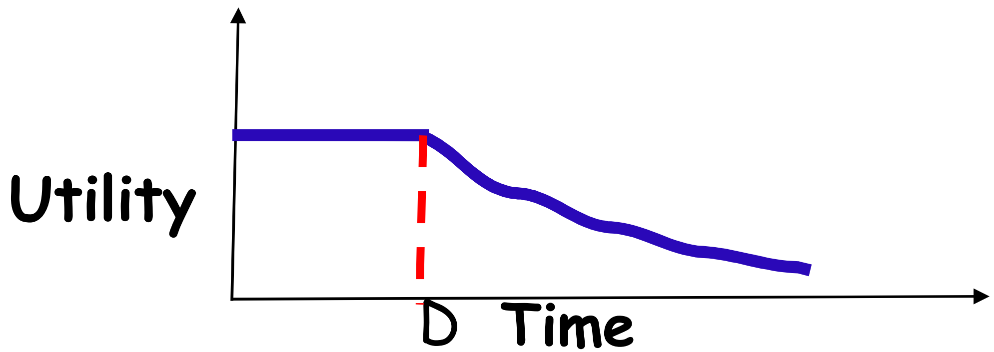
- If a deadline is missed, the system does not fail:
  - The utility of a result decreases with time after the deadline.
  - If several tasks miss deadline, then the performance of the system is said to have degraded.



9

Soft Real-Time Systems

- Use probabilistic requirements on deadline.
- For example, 99% of time deadlines will be met.



Soft Real-Time Systems

- Examples:
  - Railway reservation system
  - Web browsing
  - In fact, all interactive applications

11

characteristic of RTS

Timing Constraints

- **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.**

Real-time means to be in time ---
how do we know something is "in time"?
how do we express that?

- *Timing constraints* are used to specify temporal correctness
e.g., "finish assignment by 2pm", "be at station before train departs".
- A system is said to be *(temporally) feasible*, if it meets all specified timing constraints.
- Timing constraints do not come out of thin air:
design process identifies *events*, derives, models, and finally *specifies* timing constraints

Overall Picture…

physical properties of environment

⇓

model - design

⇓

timing constraints

⇓

analysis, testing — functional / temporal

⇓

run-time dispatching

in field use

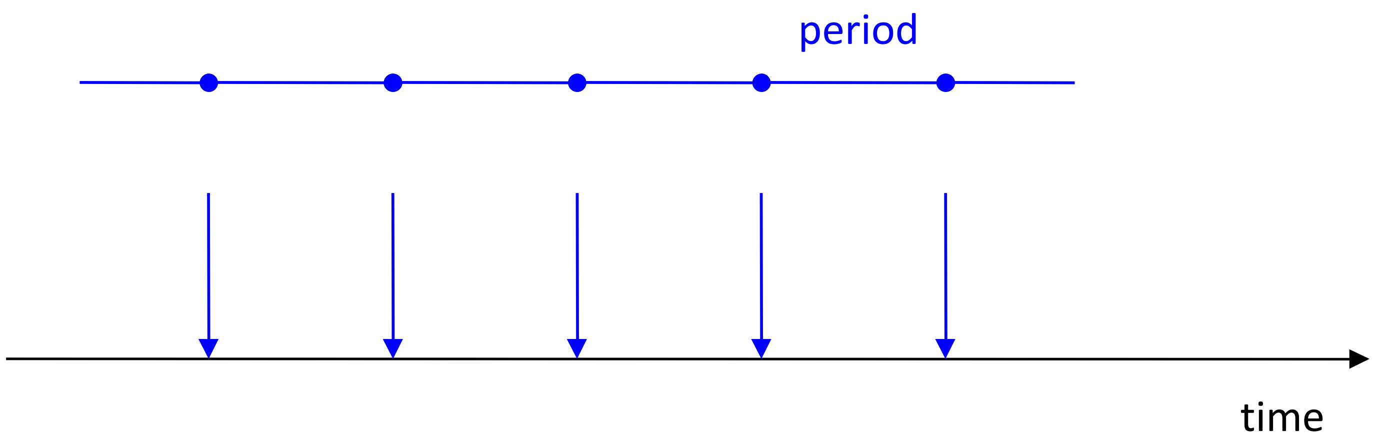
Example Autopilot

- Objective function: to control the direction
and speed of the plane.
- Outputs: actual direction and speed of the plane
- Control inputs: path markings and speed.
- Disturbances: wind, obstacles.
- subsystems: power system,
engines, steering system,
braking system, . . .

Types of Tasks

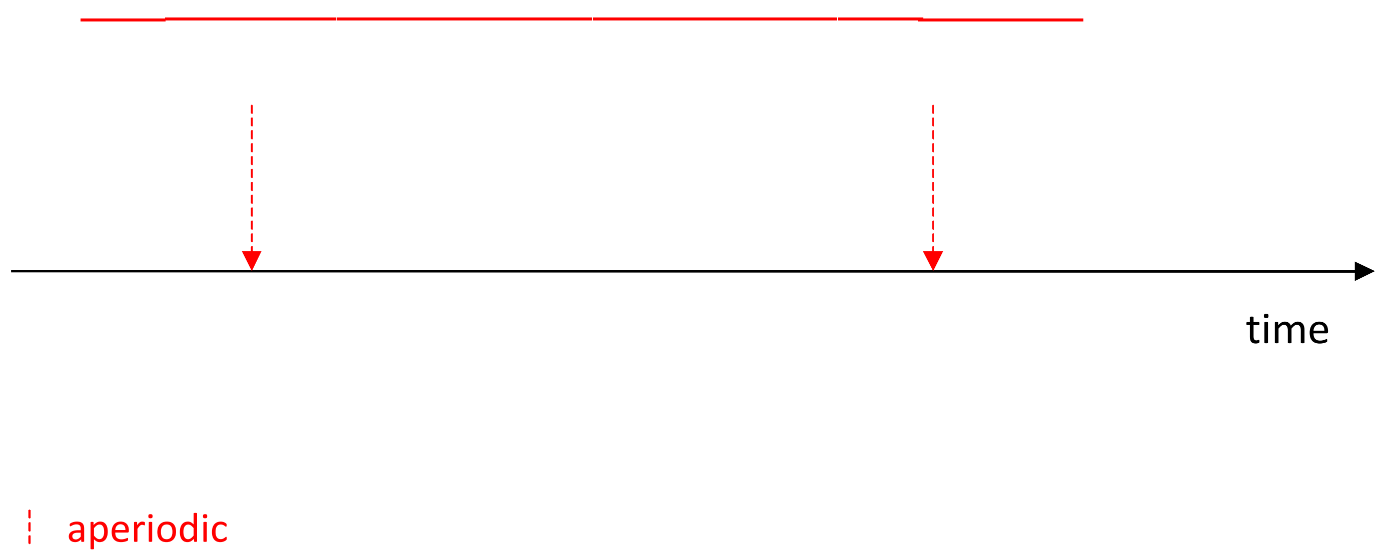
- **Periodic:**
  - Recur according to a timer
  - A vast majority all real-time tasks are periodic
- **Aperiodic:**
  - Recur randomly and are soft real-time tasks
- **Sporadic:**
  - Recur randomly, but hard real-time tasks

- Periodic
  - activity occurs repeatedly
  - e.g., to monitor environment values, temperature, etc.

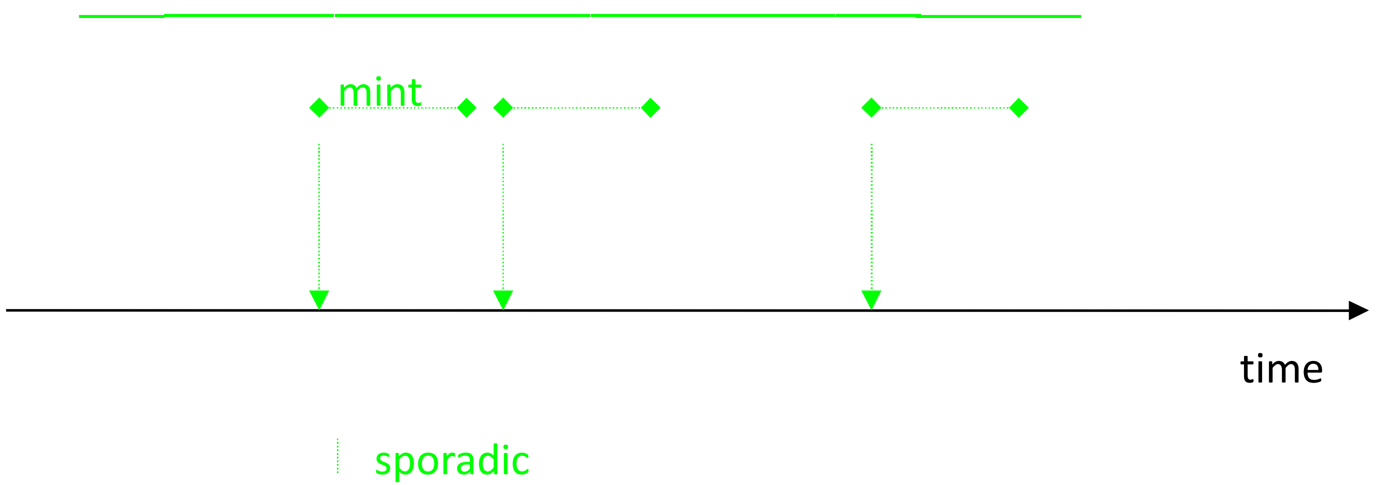


| periodic

- Aperiodic
  - can occur any time
  - no arrival pattern given



- Sporadic
  - can occur any time, but
  - minimum time between arrivals



Who initiates (triggers) actions?

Example: Chemical process

– controlled so that temperature stays below *danger* level

– warning is triggered before danger point

…… so that cooling can still occur

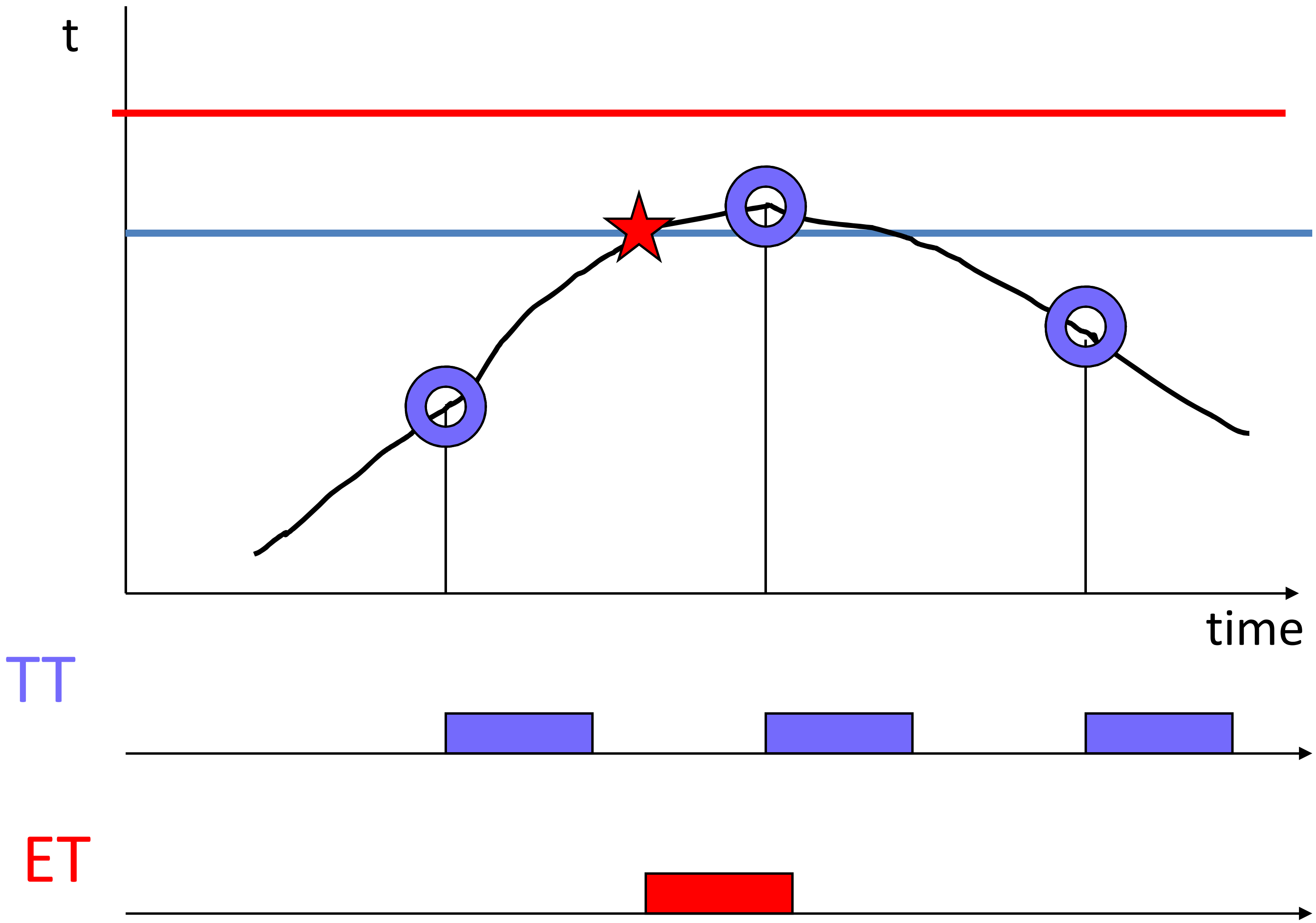
Two possibilities:

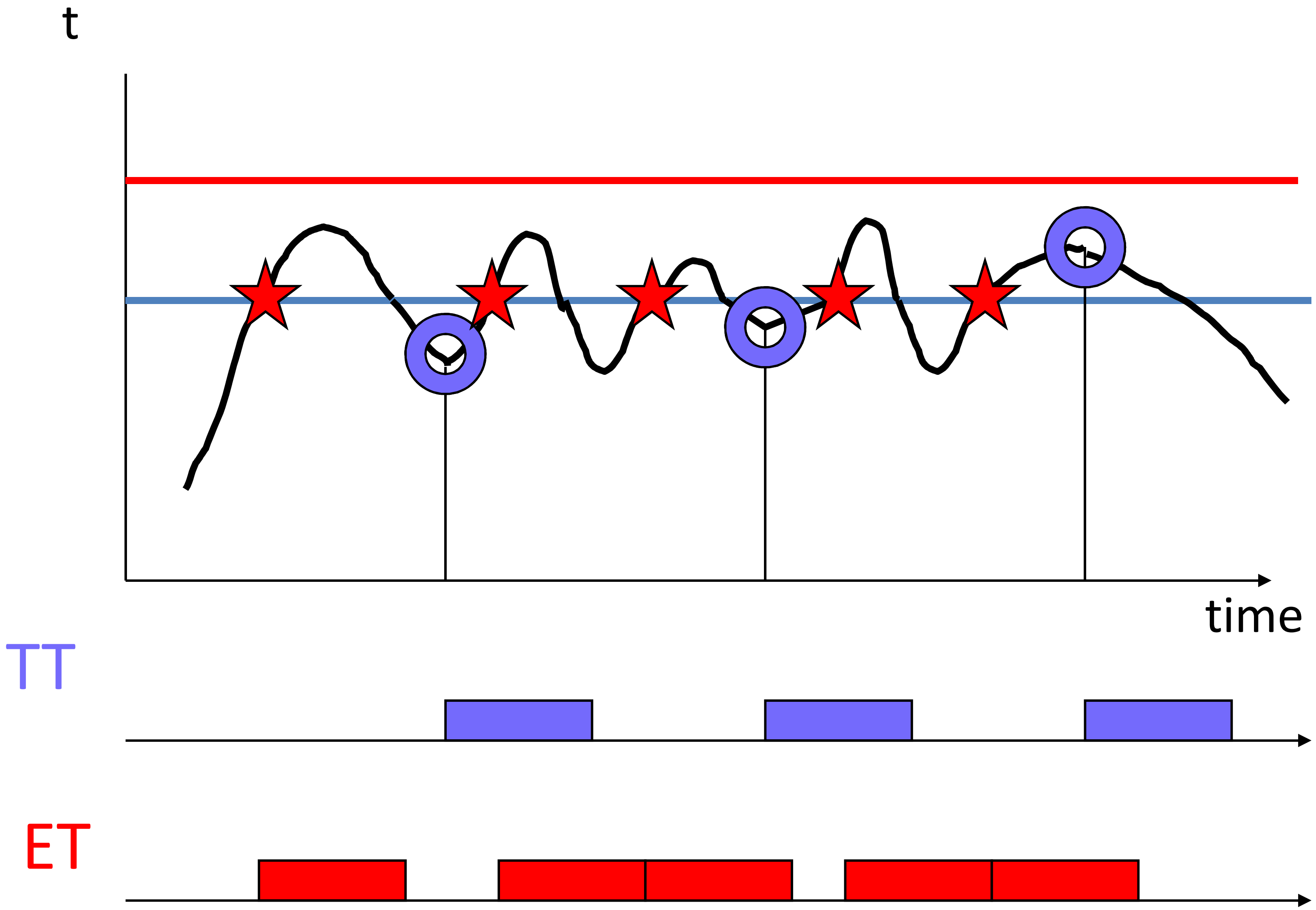
– action whenever temp raises above *warn* -- *event triggered*

– look every fixed time interval;

action taken when temp above *warn*

-- *time triggered*


t
time
TT
ET


t
time
TT
ET

ET vs TT

- Time triggered
  - Stable number of invocations
- Event triggered
  - Only invoked when needed
  - High number of invocation and computation demands if value changes frequently

Characteristic of RTS

Correctness Criterion

Correctness in RTS implies not only logical correctness of results but the time at which they are produced.

Safety-Critically

A safe system does not cause any damage even when it fails.

A reliable system can operate on long duration of time without exhibiting any failures.

Safety and reliability are bound together to form safety-critical.

Concurrency

- RTS must process data from all sensors connected in the system concurrently, otherwise system may malfunction.

Task Criticality

Task criticality is measure of cost of failure of a task.

Determined by examining how critical are the results produced by the task to proper functioning of system.

higher the task criticality, more reliable it should be made

Custom Hardware

– RT system is employed on a custom H/W that is designed and developed for a specific purpose.

Stability

Under any overloaded conditions, RTS need to continue to meet the deadlines of the most critical tasks.

Exception Handling

Exception handling in RTS ensures that even if failure occurs, the system should continue to operate in a degraded mode rather than shutting down abruptly.

FAIL-SAFE STATE

A fail-safe state of a system is one which if entered when the system fails, no damage would result.

- i.e, if a safe state can be identified and quickly reached upon the occurrence of a failure, then we call the system fail-safe.

Ex. :-

Blinking of orange light in a traffic controller system :

SAFETY-CRITICAL SYSTEM

- A safety-critical system is one whose failure can cause severe damages.

- Ex. :-

  In the traffic controller system, all lights turn green or red is not in fail-safe state :

  - green can lead to cause accidents or
  - red can cause traffic jams respectively.

ACHIEVING HIGH RELIABILITY IN REAL-TIME SYSTEM

ERROR AVOIDANCE

For achieving high reliability, every possibility of occurrence of errors should be minimized during product development.

ERROR DETECTION AND REMOVAL

- In spite of using best error avoidance techniques, still some errors creep into the codes.
- These errors needs to be detected and removed

FAULT TOLERANCE

To achieve high reliability, the system should be able to tolerate the faults and compute the correct results.

HARDWARE FAULT-TOLERANCE

(i) Built-In Self Test (BIST)

- System consists of replica of each component.

- Upon failure, system automatically reconfigures by switching out faulty component and switching in one of the redundant good components.

(ii) Triple Modular Redundancy (TMR)

- Three redundant copies of all critical components are made to run concurrently.
- System performs voting to select a single output.
- If any one of the three fails, other two can replace and mask the fault.

(b) SOFTWARE FAULT-TOLERANCE

(i) N-Version Programming

- Different teams are employed to develop N-different version of same software component concurrently.
- Results of different versions are subjected to voting at run time and the result on which majority components agree is accepted.

Drawbacks of N-Version Programming :

- Not so successful in achieving fault tolerance.

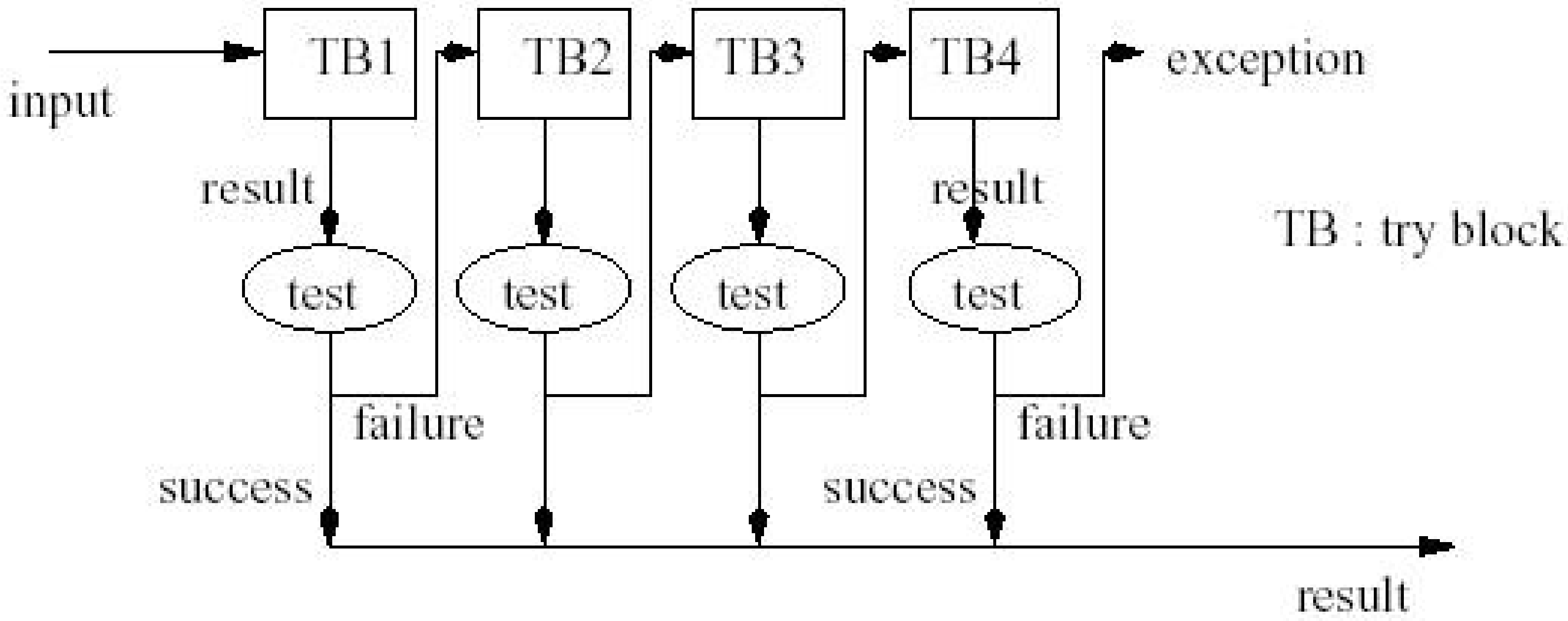
- Different versions of a component show similar failure patterns.

- Reason of failure :

    - Faults are co-related in the different versions.

    - All versions fail for similar reasons.

(ii) Recovery Blocks



Software Fault Tolerance using recovery blocks

(iii) Check point and Roll-back Recovery

- As the computation proceeds, the system state is tested each time after some meaningful progress in computation is made.
- After a successful state-check test, the state of the system is stored in a stable storage.
- In case the next test does not succeed, the system can be made to roll back to the last checkpointed state.