Real-Time Operating Systems
RTOS

For Embedded systems

- Rajendra Singh

Real-Time Systems

- A system is said to be Real Time if it is required to complete it's work & deliver it's services **on time**.
- Result in severe **consequences** if logical and timing correctness are not met
- **Example**
  - Flight Control System
  - controlling traffic signal
  - nuclear reactor
- **Non Example** – PC system

What a RTOS is not

- Real time computing is equivalent to **fast computing**.
- Real time systems operate in a **static environment**
- Real time **programming** involves assembly coding, writing device drivers.

Real-Time Systems (cont.)

- Two types exist
  - **Soft** real-time
    - Tasks are performed as fast as possible
    - Late completion of jobs is **undesirable but not fatal**.
    - System performance degrades as more & more jobs miss deadlines
    - Also known as “best effort” systems
    - Example :
      - » multimedia transmission and reception
      - » networking, telecom (cellular) networks
      - » web sites and services, Online Databases
      - » computer games

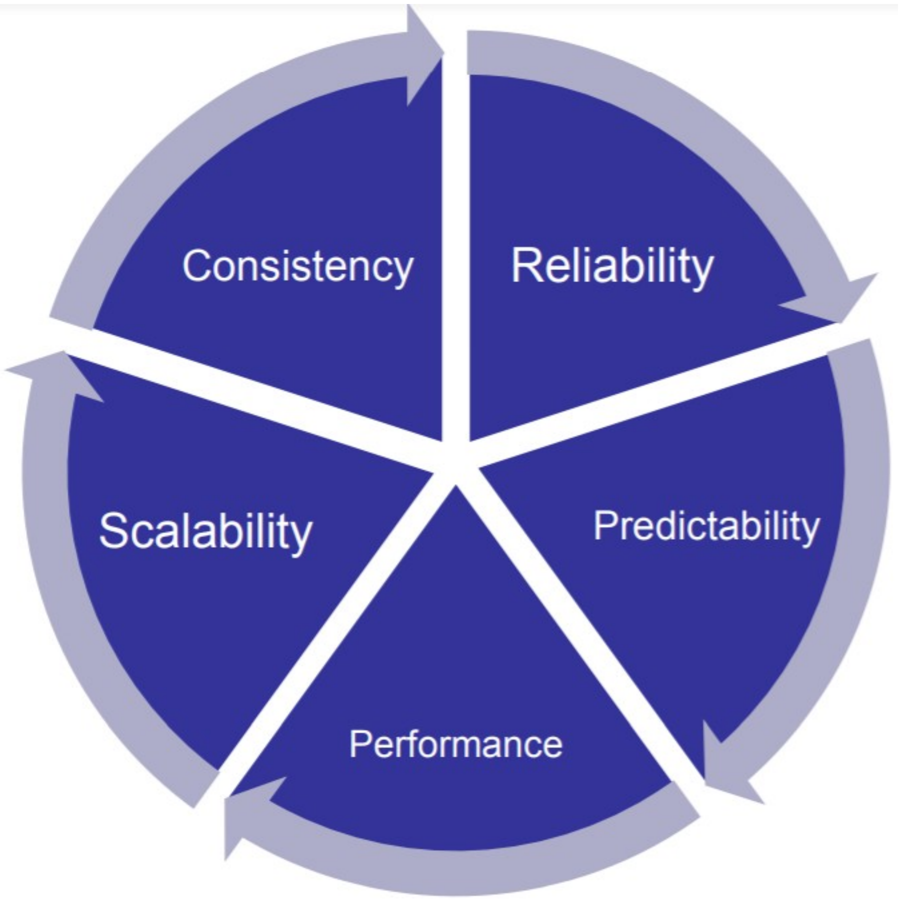
Real-Time Systems (cont.)

- **Hard** real-time
  - Tasks have to be performed on time
  - Failure to meet deadlines is **fatal**
  - Requires formal verification/guarantees of being to always meet its hard deadlines (except for fatal errors).
  - Example :
    - » Flight Control System, air traffic control
    - » vehicle subsystems control
    - » Nuclear power plant control

Most Real-Time Systems are embedded

- An embedded system is a computer built into a system but **not seen by users** as being a computer
- Examples
  - FAX machines
  - Copiers
  - Printers
  - Scanners
  - Routers
  - Robots

CHARACTERISTICS OF RTOS



Role of an OS in Real Time Systems

- **Standalone** Applications
  - Often **no** OS involved
  - Micro **controller** based Embedded Systems
- Some Real Time Applications are huge & **complex**
  - **Multiple** threads
  - Complicated Synchronization Requirements
  - File system / Network / Windowing support
  - OS primitives reduce the software design time

Role of an OS in Real Time Systems

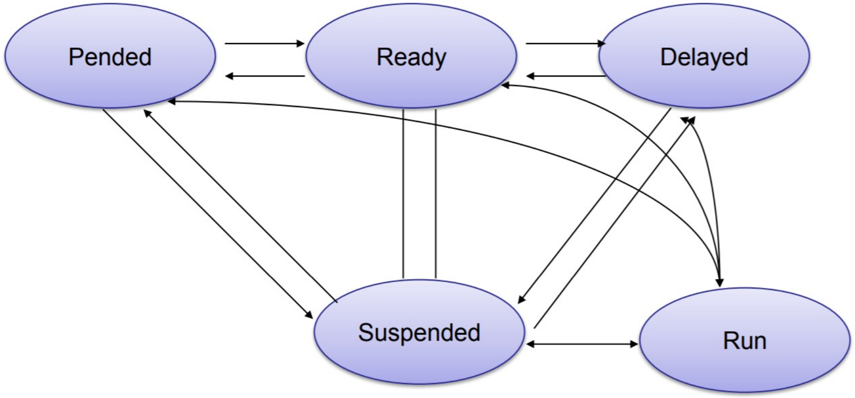
- **Standalone** Applications
  - Often **no** OS involved
  - Micro **controller** based Embedded Systems
- Some Real Time Applications are huge & **complex**
  - **Multiple** threads
  - Complicated Synchronization Requirements
  - File system / Network / Windowing support
  - OS primitives reduce the software design time

FUNCTIONS OF RTOS

- Task management
- Scheduling.
- Resource Allocation.
- Interrupt Handling.

Task management

- In Real Time Applications the **Process** is called as Task which takes execution time and occupies memory.
- Task management is the process of **managing** tasks through its **life cycle**.



Scheduling in RTOS

- More **information** about the tasks are known
  - Number of tasks
  - Resource Requirements
  - Execution time
  - Deadlines
- Being a **more deterministic** system better scheduling algorithms can be devised.

Scheduling Algorithms in RTOS

- Clock Driven Scheduling
- Weighted Round Robin Scheduling
- Priority Scheduling(Greedy / List / Event Driven)

Scheduling Algorithms in RTOS (*cont.*)

- **Clock Driven**
  - **All** parameters about jobs (execution time/deadline) **known** in advance.
  - Schedule can be computed **offline** or at some **regular time** instances.
  - **Minimal** runtime **overhead**.
  - **Not** suitable for many applications.

Scheduling Algorithms in RTOS (*cont.*)

- **Weighted Round Robin**
  - Jobs scheduled in Round Robin manner
  - Time quantum given to jobs is proportional to it's weight
  - Example use : High speed switching network
  - Not suitable for precedence constrained jobs.
    - Job A can run only after Job B. No point in giving time quantum to Job B before Job A.

Scheduling Algorithms in RTOS (*cont.*)

- **Priority Scheduling**
  - Processor **never** left **idle** when there are **ready** tasks
  - Processor allocated to processes according to priorities
  - Priorities
    - **Static** - at design time
    - **Dynamic** - at runtime

Priority Scheduling

- Earliest Deadline First (EDF)
  - Process with earliest deadline given highest priority
- Least Slack Time First (LSF)
  - slack = (relative deadline – execution left)
- Rate Monotonic Scheduling (RMS)
  - For periodic tasks
  - Tasks **priority inversely** proportional to it's **period**

Resource Allocation in RTOS

- Resource Allocation
  - Resources can be allocated in
    - Weighted Round Robin
    - Priority Based

- Some resources are non preemptible
  - Example : semaphores

Assigning Task Priorities

- In most systems, **not all** tasks are **critical**
  - Non-critical tasks are obviously low-priorities
- Most real-time systems have a **combination** of soft and hard requirements

Other RTOS issues

- **Interrupt Latency** should be very **small**
  - Kernel has to respond to real time events
  - Interrupts should be disabled for minimum possible time
- For embedded applications **Kernel Size** should be **small**
  - Should fit in ROM
- **Sophisticated features** can be removed
  - No Virtual Memory

CONCLUSION.

- RTOS have been the heroes in most of the technological areas, right from **fuel** injection system to **nuclear** reactor control, **satellite** control, global positioning systems(**gps**), and fully equipped high-tech warfare **aircrafts**.

And the best is yet to come!

Thank you!