Introduction

An operating system is the software installed on a computer that is in charge of managing all other software and even hardware on the computer. It is the environment through which programs are executed. The real time factor means that the operating system has time constraints within which it has to deliver certain things or fulfill a certain criterion. These systems are able to resolve issues simultaneously such as rapid response, failure of components & connections (Microsoft Word for example automatically creates a back up of the document being edited even in the event of the computer crashing), need to adapt over time, bursty stimuli and continuous operation.

Design Issue of Real Time Systems

# The Timing Requirements

In an RTOS the system’s performance is not just governed by the quality of its results but on the timeframe within which it was able to deliver such results. Just like traditional real-time systems there are also hard and real-time operating systems. A ‘hard’ RTOS would be that of a driverless car, it must always be able to make decisions in a specific timeframe and if not it would most likely crash, which could be classified as failure. For a soft one like that on a laptop (Note that a laptop may also have features of a ‘hard’ real time system) it is not a failure that a video or program is taking a long time to open. For a soft RTOS it’s a non-functional requirement to deliver results in the required timeframe. Optimally an RTOS should be able to deliver as many results as it can in a given timeframe.

* Predictability: all parts of the system must be time bounded to achieve this. Predictability depends on hardware, software, implementation & the human factor (it may be hard to predict how the end user will use the RTOS)
* Temporal consistency: This refers to the need to maintain a (time-based) consistency between an environment (controlled system) and the RT system (controlling system). Behind the scenes an ‘image’ of the state of the environment is maintained in system. In a car for example the OS must keep track of how fast it’s been going, how close it is to other cars etc.

# Scheduling Paradigms

Tasks may have different time constraints and a periodic (within pre-specified time intervals) or aperiodic (whenever there’s external stimulus) execution behavior. Aperiodic tasks may be transformed into periodic tasks.

An RTOS is able to execute multiple simultaneous tasks which may have different constraints within which they operate. Tasks are either periodic meaning they are executed within a pre-specified time interval (if you set an alarm in your phone it will always ring at the specified time) or they are aperiodic meaning they are a response to external stimulus (such as interaction with the end user, failure of a component etc.)

1. Static table-driven approaches: the RTOS analyses the tasks based on a schedule and create a table that will then be used to choose which tasks to execute at a runtime. This enforces hard deadlines as whenever the system encounters a new task the table has to be reworked.
2. Static priority-driven pre-emptive approaches: In this case tasks are executed based on priority
3. Dynamic planning –based approaches: system checks how easy it would to be to execute a task at runtime
4. Dynamic best effort approaches: System just does its best to meet deadlines which may be lead to tasks been aborted while being executed. This is therefore a risky method and may lead to crashes given the task is quite an important one but it may be useful to end tasks that waste time while executing

Today’s Real Time Systems & Problems with Current Approaches

1. Important properties of the environment: processors and microcontrollers are embedded in most of our devices e.g. phones, cameras, cars etc. They are forced to work with limited resources (size, energy etc. constraints are enforced so that they fit in the devices). They must also be able to deliver within time limits. These systems must be adaptive enough to cope with ever changing environments. They must be able to react well to the state of the environment and not just focus on preparing for the worst case that could be encountered as that in itself is not really adaptive
2. Novel applications requirements: In most consumer electronics the three most common types of activities exhibited by application software are control (these usually are executed in hard deadlines, tasks here are usually periodic), media processing (these consume a lot of resources, they are also soft and aperiodic) and interaction (these also consume quite a lot of resources)
3. Approach to achieve some level of adaptiveness today: work should be done to determine, design and test use cases. Developers may op to use tools to do this. What happens when a feature is added or removed from the system must be predictable. In the event of a failure occurring it should be easy to reconfigure the system.
4. Problems with current approach: Priorities are used to a large extent to determine when tasks are executed. However, there are many activities that the system may find difficult to allocate an appropriate priority level to. Basing tasks on priority also leads to certain constraints being ignored, which may lead to failure or breakdown. Priorities aren’t necessarily dynamic which is a problem as the importance of a task may wane over time or may not be as useful at certain points.

Conclusions

The eternal challenge of the RTOS is how to execute applications efficiently given constraints and also fulfill nonfunctional requirements. More work is being done to make these systems more flexible and predictable. Areas to possibly researched regarding RTOS are flexible scheduling services, protection, dynamicity, quality of service, multiprocessor support, drivers, networks, modeling

REFERENCES

* Nakate Shraddha S., Meshram B. B., Chavan Jayamala P., “New Trends in Real Time Operating Systems” <http://www.techrepublic.com/resource-library/whitepapers/new-trends-in-real-time-operating-systems/> Accessed February 26,2016
* Marshall L., “The Nature of Real-Time and Embedded Systems” Accessed February 26, 2016