

**COMP3301**

---

**Lecture 12**

**Real-time operating systems**

---

School of Information Technology and Electrical Engineering  
The University of Queensland

**COMP3301**

**Real-Time systems**

---

- What is a real-time system?
  - System where timing of computation is as important as the result
  - A late result may be as useful as an incorrect result (i.e. not very useful at all!)
- Soft vs. Hard real-time systems
- What are some examples?
  - ABS in cars
  - Multimedia

2

**COMP3301**

**Real-time systems performance**

---

- Is a real-time system a high-performance system?
  - Not necessarily!
- Safety-critical systems are often not high performance (compared to PCs)
- Correct, timely operation is far more important than increasing the average performance
  - Average vs. Worst-case

3

**COMP3301**

**What is an RTOS?**

---

- How is it different from any other OS?
  - Scheduling algorithms need to complete in guaranteed time
  - Interrupts must be handled within guaranteed time
  - All system tasks need to finish in guaranteed time:
    - Allocation of memory
    - Inter-process communication

4

**COMP3301**

**Designing an RTOS**

---

- Design issues
  - Bounded computation
  - Determinism
  - Verifiability
  - Design integrity
- Some application sectors have more stringent requirements
  - verification of tools
  - verification of platforms
  - redundant hardware
  - redundant software
  - etc.

5

**COMP3301**

**Scheduling in RTOSs**

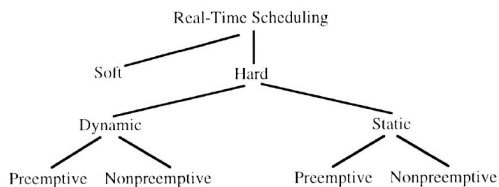
---

- Tasks (processes) usually categorised:
  - periodic – e.g. handling audio samples
  - aperiodic – e.g. responding to brakes locking

6

## Scheduling algorithms

- Categories of algorithms:
  - Which one to choose?



7

## Scheduling algorithms

- Dynamic vs. Static
  - Dynamic schedule computed at run-time
  - Static schedule done at compile-time for all *possible* tasks
- Preemptive vs. non-preemptive
  - Higher priority tasks can preempt lower priority tasks

8

## Schedulability

- Can all the tasks be scheduled and be sure that they all meet their deadlines?
- If there are dependencies, this is usually a hard problem!
- May be possible to prove that a set of tasks definitely CAN or CANNOT be scheduled
  - Often, a set of tasks will fall somewhere in between!

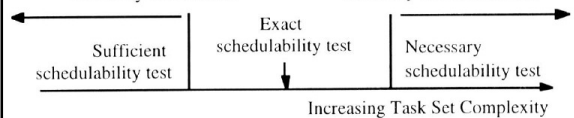
9

## Schedulability

- Sufficient vs. Necessary schedulability

If the sufficient schedulability test is positive, these tasks are definitely schedulable

If the necessary schedulability test is negative, these tasks are definitely not schedulable



10

## Miscellaneous RT terminology

- Jitter
  - Difference between expected arrival of periodic event and actual
- Period
  - Time between runs of a task (or arrivals of events)
- Latency
  - Time from input of one sample to corresponding output
- Deadline
  - Time by which operation must be completed
- Laxity
  - Time allowed before task MUST begin executing
- Worst-Case Execution Time (WCET)
  - May be applied to parts of the OS as well as to applications

11

## Scheduling algorithms

- Back to the algorithms now:
  - Searching for process to run needs to be bounded!
  - Static scheduling is relatively easy to determine
    - Set of processes with known characteristics
    - Determine if all can be fitted into the available CPU time

12

## Scheduling algorithms

- Dynamic scheduling is a lot harder!
  - Earliest Deadline First – given highest priority
  - Least Laxity
- Both can achieve high utilisation
  - Can also be hard to prove that it works in all cases
  - Just because it works for a while, doesn't mean it will always work

13

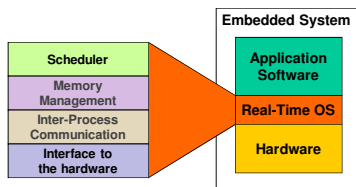
## Scheduling algorithms

- Rate Monotonic Scheduling
  - Priorities are static
  - Shortest period task gets highest priority
  - Can provide guarantees about scheduling
  - Usually can only achieve ~70% utilisation

14

## Verifying an RTOS

- Need to verify both:
  - Timing
  - Functionality



15

## How to verify timing?

- Range of techniques:
  - Usually involves tracing worst-case paths through both application and kernel
  - OS vendor needs to provide timing characteristics:
    - WCET for all system calls
    - WC Interrupt latency
    - Context switch time

16

## Examples

- Many RTOS vendors exist!
- There are a range of licensing models commonly used, so choice is largely application dependent:
  - Cost (free/open-source to \$50k+)
  - Royalty payments may be additional
  - Source code availability
  - Technical support and training
  - Certification

17

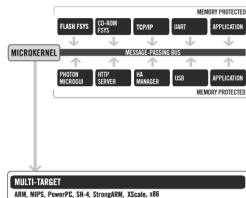
## Examples

- QNX
  - Microkernel-based, closed source
  - Used in many industrial controllers and cars
- VxWorks
  - Kernel-mode only, threads only, source available (for many dollars!)
  - Used by aerospace as well as in various communications products
- $\mu$ C/OS-II
  - Royalty-free, certified for safety-critical systems
  - Used in large range of industrial controllers
- RTAI/RTLinux
  - Open-source (sort-of for RTLinux)
  - Hard to know what it's used in...

18

## A brief look at QNX

- See the differences in microkernel-based OS
- Components can be upgraded while in service



19

## A brief look at QNX

- Microkernel, very similar to Minix 3 in structure:
  - Kernel controls only message passing and scheduling
  - System processes include
    - Process manager
    - Filesystem manager
    - Device manager
    - Network manager

20

## A brief look at QNX

- Scheduling:
  - occurs when:
    - process becomes unblocked
    - timeslice for running process expires
    - running process is preempted
  - All processes assigned priority
  - 3 different scheduling algorithms available on a per-process basis:
    - FIFO
    - Round-robin
    - Adaptive
  - Servers can inherit priority of client to avoid priority inversion

21

## A brief look at QNX

- So what makes it a real-time operating system?
  - Scheduling algorithm is bounded
  - Interrupt latencies are bounded by the kernel
  - Overall design ensures that timing is deterministic

22

## Summary

- Real-time systems are a surprisingly large market – with over a hundred vendors
- There is a huge amount of theory proving the correctness of algorithms
- Real RTOSs may look very similar to traditional OSs
  - Verification is a vital issue!
    - Especially for hard real time

23