Real-Time Operative Systems Course

Course Presentation

Paulo Pedreiras

DETI/UA/IT

October 7, 2021

Faculty

- Paulo Pedreiras (Course Coordinator)
- pbrp@ua.pt
- http://ppedreiras.av.it.pt/

Preliminaries

What are real-time systems?

- Computational systems
- Subject to the evolution of real-time (or physical time, if you wish)

Systems in which it is required to do the right thing at the right time, otherwise things can go (very) wrong!

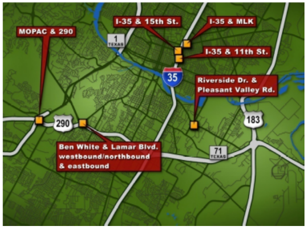
Course objectives

- Main topic:
  - **Operating Systems** , **Programming Techniques** and **Analysis tools** to support the design of systems that interact with (or simulate a) physical process (or environment) so that they can do the **right thing at the right time**
- We will address:
  - The source and characterization of the restrictions imposed by the environment to the temporal behavior of the computational system;
  - Approaches to allow the computational system to be aware of the state of the environment that surround it;
  - The scheduling theory of concurrent tasks associated with real-time processes;
  - The structure and functionality of real-time operating systems
  - **And apply all these knowledge in practice!**

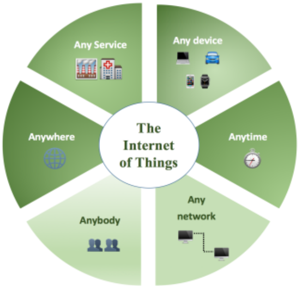
Examples of Real-Time Systems

Agile Manufacturing

Traffic control system



MOPAC & 290
I-35 & 15th St.
I-35 & MLK
I-35 & 11th St.
Riverside Dr. & Pleasant Valley Rd.
Ben White & Lamar Blvd. westbound/northbound & eastbound
35
290
183
71
1



Collision free motions are determined by Realtime Robotics controller
RTR Connected
SIEMENS

Image Credits: Realtime Robotics

Common misunderstandings

- **Isn't a quick processor sufficient?**
  - For a simple program, with a single task, that may work. However, when CPU(s) have to execute several tasks concurrently, just a quick processor may not be enough! Some tasks may block others and cause big and/or unpredictable delays.
- **Then, what do we need?**
  - Scheduling! Which means, select the right task to execute in every instant.
    There are scheduling algorithms that allow predicting and minimizing the maximum delays that tasks may suffer.

Common misunderstandings

- **So all this stuff applies only when we need multi-tasking... ?**
  - As stated above, we are considering situations where a computer has to perform several tasks simultaneously.
    It is normal that in such situations we use one multi-tasking operating system/kernel.
  - But often, even when the main body of the program is a simple endless loop, there are various pseudo-tasks, part of asynchronous interrupt routines, which lead to the same situation.
    The triggering of the interrupt routines may also be delayed, or even discarded. We must use proper techniques to bound and compute these delays.

Common misunderstandings

- **Are those delays so important?**
  - It depends on the kind of system. If we're talking e.g. about control systems on a plane, a car's X-by-wire or a robot that moves around other people and equipment ... **THEN YES!**
  - On the other hand, if we are talking about multimedia systems or routers in networks, delays in tasks shall not cause death to anyone, but there will be a loss of Quality-of-Service.

Course organization

- **Theoretical component:** presentation and discussion of concepts and techniques
  - Students should read selected parts of the base bibliography
  - Slides are available at the course website
  - Discussions are welcome!
- **Lab component:** learning about RTOS and tools, with focus on its practical use
  - Groups of 2 students
  - Tutorial classes to establish a basic set of practical competences: Linux (GPOS), Xenomai, FreeRTOS, OpenMP.
  - A final project

Grading

Normal period

- Theoretical component: 50%
    - 40% written exam + 10% short quizzes, during classes
- Lab component: 50%:
    - 30% project + 5% for tutorial sessions + 15% practical part of the written exam

Resit period

- Theoretical component: 50%
    - Written exam
- Lab component: 50%
    - Lab grade from the normal period or lab exam.

Class plan

Real-Time Operative Systems plan

Class 1:	Lecture 0+1: Course presentation + Basic concepts about real-time systems
Class 2:	Lecture 2: Computational models and RTOS architecture + Tutorial GPOS (1/2)
Class 3:	Lecture 3: Basic concepts on scheduling + Tutorial GPOS (2/2)
Class 4:	Lecture 4: Fixed-Priority scheduling + Tutorial Xenomai (1/2)
Class 5:	Lecture 5: Dynamic-Priority scheduling + Tutorial Xenomai (2/2)

Class plan

Real-Time Operative Systems plan (cont.)

- Class 6: Lecture 6: Shared resources + Tutorial FreeRTOS (1/2)
- Class 7: Lecture 7: Aperiodic task scheduling + Tutorial FreeRTOS (2/2)
- Class 8: Lecture 8: Additional topics related with RT scheduling + Project presentation
- Class 9: Lecture 9: Optimizations + Project
- Class 10: Lecture 10: Multiprocessor Scheduling + OpenMP Tutorial

Class 11 to Last-1: Project

Last Class: Project Demo and Presentation

Bibliography

- Base
  - Giorgio Buttazzo (2011). HARD REAL-TIME COMPUTING SYSTEMS: Predictable Scheduling Algorithms and Applications, Third Edition, Springer, 2011. (available at the course site
- Complementary
  - Kopetz, H. (2011). Real-Time Systems: Design Principles for Distributed Embedded Applications (Real-Time Systems Series), 2nd Edition , Springer, 2011.
  - Xiaocong Fan (2015). Real-Time Embedded Systems: Design Principles and Engineering Practices, 1st Edition, Springer, 2015
  - Jane W.S. Liu (2000). Real-Time Systems. Prentice Hall.
  - Richard Barry. Using the FreeRTOS Real-Time Kernel - A practical guide, Real-Time Engineers, Ltd. (available at the course site)

And now ....

- It is time to wake up and start working!