REAL TIME SYSTEMS

1. Real Time Operating systems: an overview

Brief overview of Real-Time Systems

Objective

Hide the particularities of the hardware from the application

=> more or less complex virtual machine

OS Classification :

Generalist (UNIX...)

Real-time extended generalists (Linux, POSIX...)

Original real time

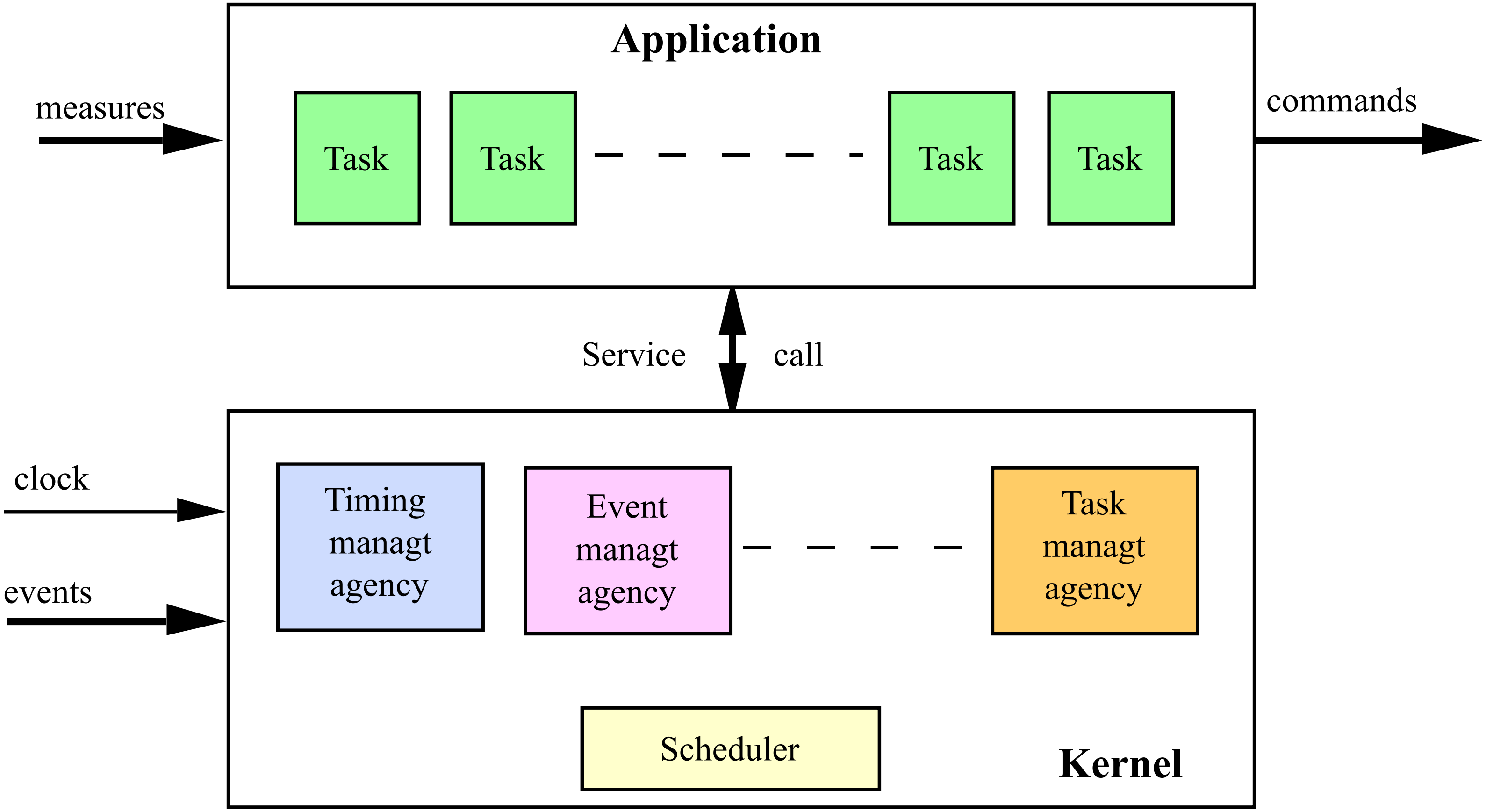
Small kernels for limited embedded applications

=> APEX

=> OSEK

Brief overview of Real-Time Systems

General structure:



Generalities about Real-Time OS

The main characteristics of real-time kernels

- **Conformity to a standard or pseudo-standard** (POSIX, Sceptre project)
- **Compactness** (for embedded applications)
- **Target environment** (microprocessors, architecture, …)
- **Host environment** (OS type)
- **Development tools** (debug, online analysis, …)
- **Real-time functions** (list of all services provided)
- **Characteristics of the scheduler** (scheduling policies)
- **Temporal characteristics :**
  - **interrupt latency:** time during which interrupts are masked and therefore cannot be taken into account (execution of atomic primitives, manipulation of critical structures, ...)
  - **preemptive latency:** the maximum amount of time the kernel can delay the scheduler.
  - **task response time:** time between the occurrence of an interruption and the execution of the woken up task.

Generalities about Real-Time OS

Two main types of real-time OS:

- the original Real-Time OS:
  - Domain-oriented OS (aeronautics, automotive…)
  - General real-time OS (Tornado, QNX, …)

  - allow a fine management of priorities
  - offer fast system primitives, in limited time (management of interrupts, semaphores...)
  - no virtual memory, but locking pages in main memory
  - minimizing overhead (the time taken by the system to run and manage itself)

=> the solution to Hard Real-Time

Generalities about Real-Time OS

Two main types of real-time OS:

=> the classical O.S. (Unix...) extended for real time

Enable concurrent development of real-time and non-real-time applications in a standard and comfortable environment.

- But it took:
  - review the scheduling policies
  - reinforce the notion of preemption
  - define reentrant system primitives
  - define the notion of thread (to facilitate preemption with context saving and then recovery with context restitution)

=> important and complex modifications

RTAI

RTLinux

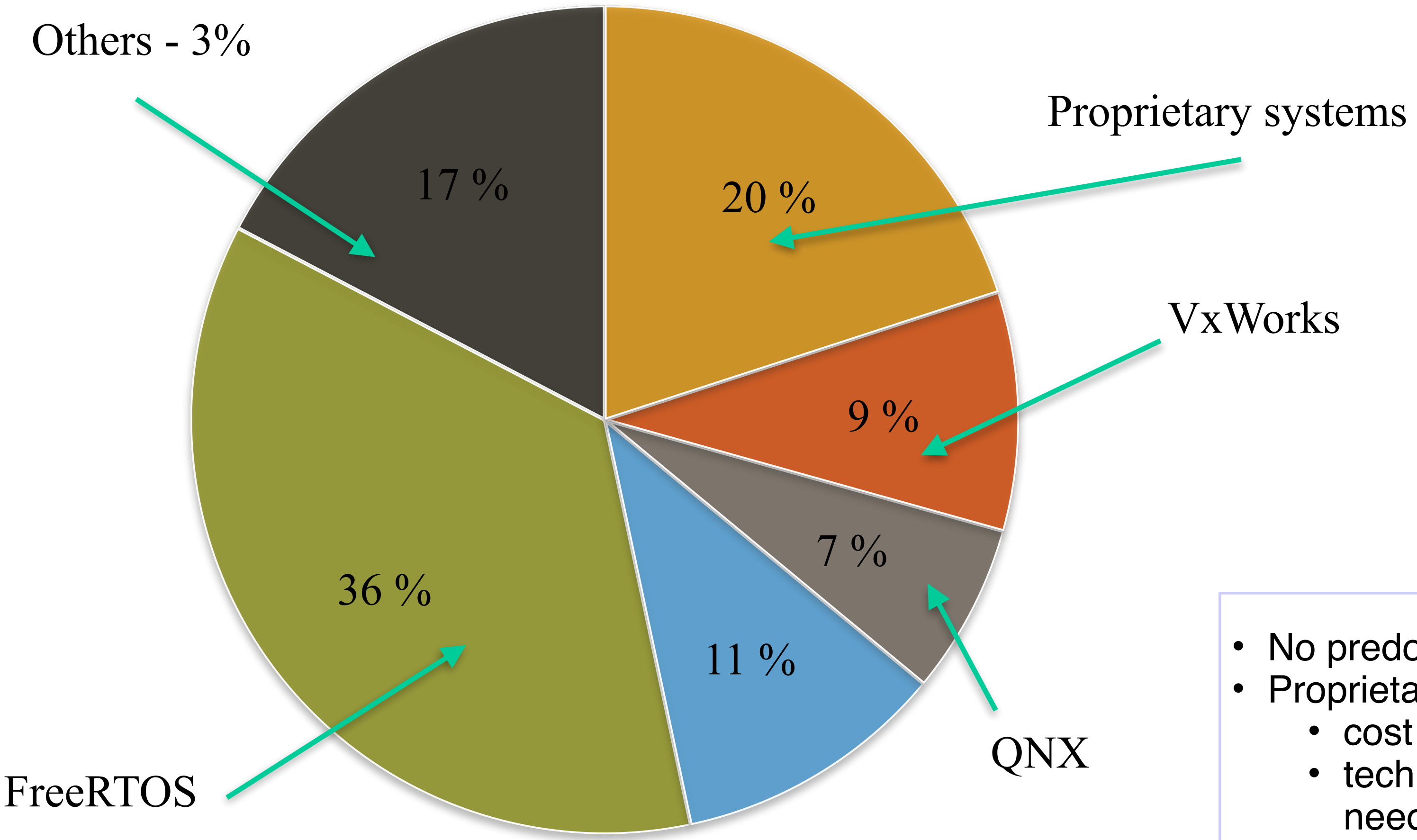
Windows CE

=> for Soft Real-Time

Generalities about Real-Time OS

Situation of the industrial supply of real-time kernels

Embedded Market Study (USA - 2014)



- No predominance of one system
- Proprietary systems :
  - cost (licence)
  - techniques (adaptation to needs)
  - strategy (control)