Real-time Software Design

이동아

※ This lecture note is based on materials from Ian Sommerville 2006.

Objectives

- To explain the concept of a real-time system and why these systems are usually implemented as concurrent processes
- To describe a design process for real-time systems
- To explain the role of a real-time operating system
- To introduce generic process architectures for monitoring and control and data acquisition systems

Topics covered

- System design
- Real-time operating systems
- Monitoring and control systems
- Data acquisition systems

Real-time systems

- Systems which monitor and control their environment.
- Inevitably associated with hardware devices
  - Sensors: Collect data from the system environment;
  - Actuators: Change (in some way) the system's environment;
- Time is critical
  - Real-time systems MUST respond within specified times.

Definition

- A real-time system is a software system where the correct functioning of the system depends on the results produced by the system and the time at which these results are produced.
- A soft real-time system is a system whose operation is degraded if results are not produced according to the specified timing requirements.
- A hard real-time system is a system whose operation is incorrect if results are not produced according to the timing specification.

Stimulus/Response Systems

- Given a stimulus, the system must produce a response within a specified time.

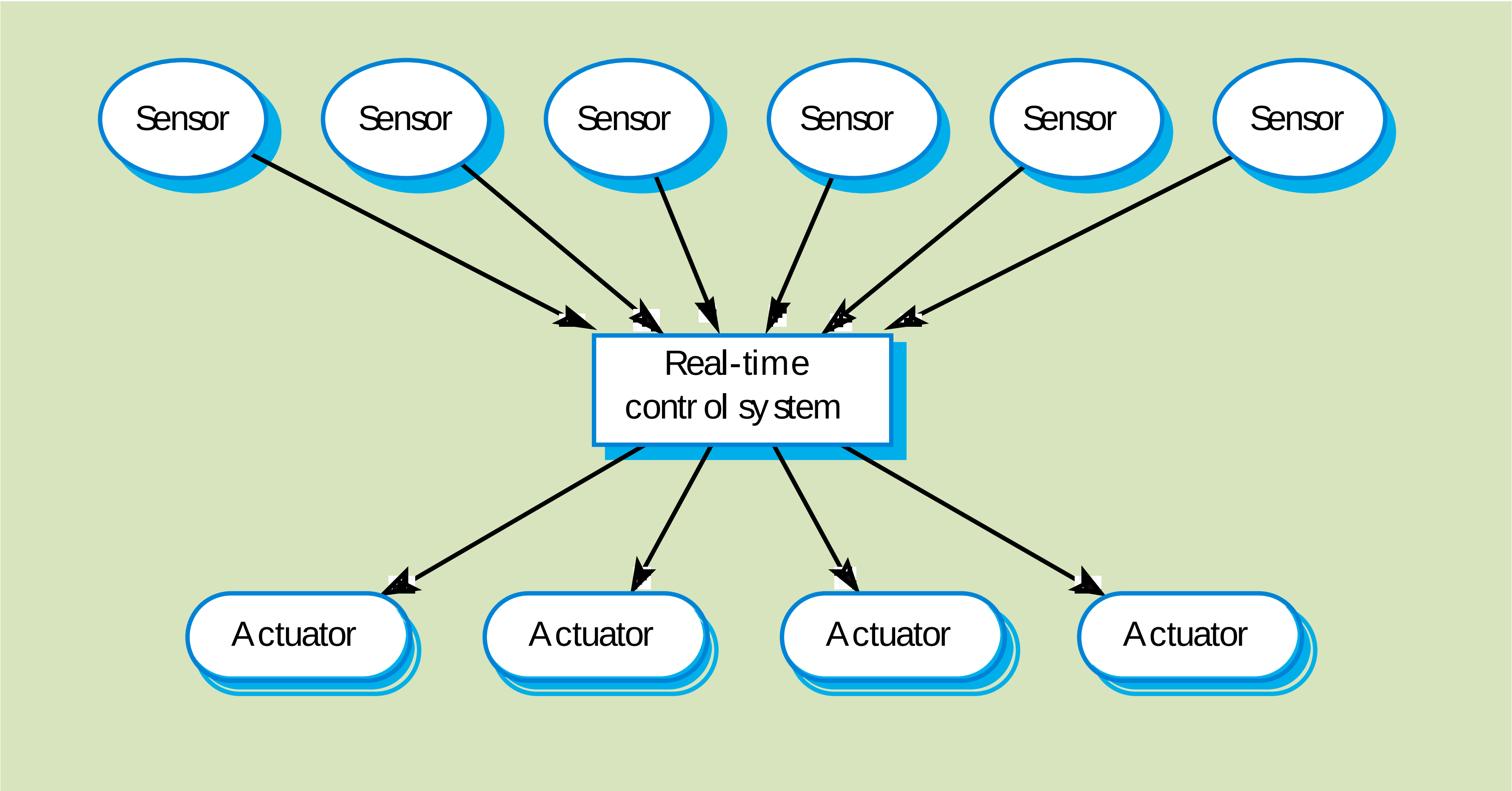
- Periodic stimuli. Stimuli which occur at predictable time intervals
  - For example, a temperature sensor may be polled 10 times per second.

- Aperiodic stimuli. Stimuli which occur at unpredictable times
  - For example, a system power failure may trigger an interrupt which must be processed by the system.

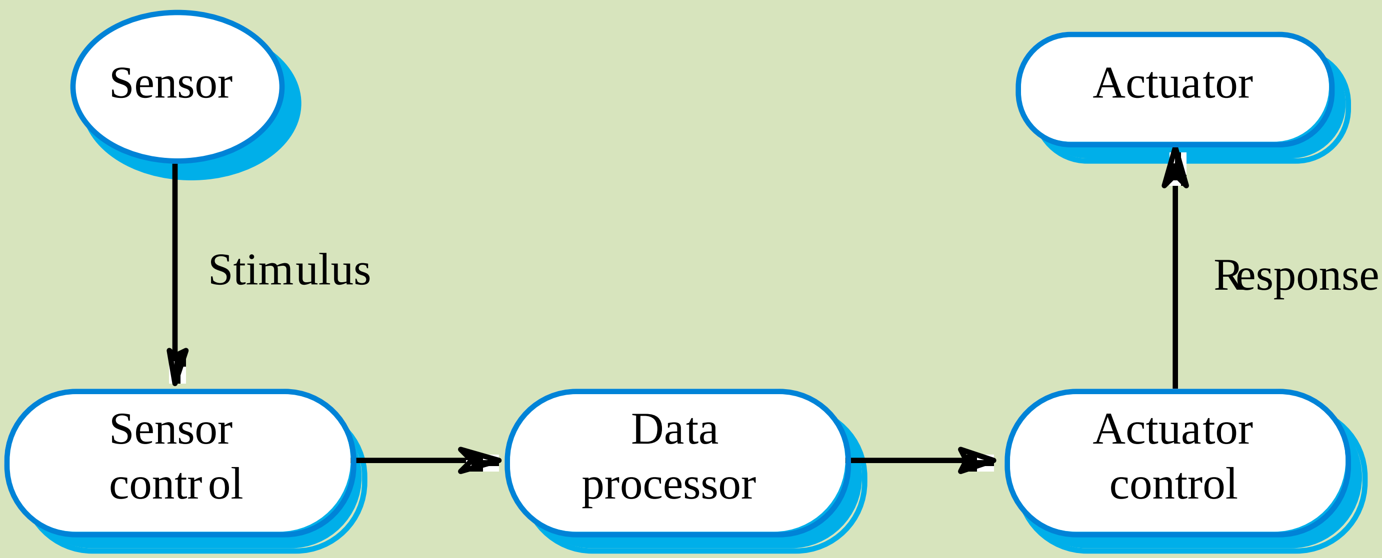
Architectural considerations

- Because of the need to respond to timing demands made by different stimuli/responses, the system architecture must allow for fast switching between stimulus handlers.
- Timing demands of different stimuli are different so a simple sequential loop is not usually adequate.
- Real-time systems are therefore usually designed as cooperating processes with a real-time executive controlling these processes.

A real-time system model



Sensor/actuator processes



System elements

- Sensor control processes
  - Collect information from sensors. May buffer information collected in response to a sensor stimulus.
- Data processor
  - Carries out processing of collected information and computes the system response.
- Actuator control processes
  - Generates control signals for the actuators.

Real-time programming

- Hard-real time systems may have to programmed in assembly language to ensure that deadlines are met.
- Languages such as C allow efficient programs to be written but do not have constructs to support concurrency or shared resource management.
- Java 2.0 is not suitable for hard RT programming but real-time versions of Java are now available that address problems such as
  - Not possible to specify thread execution time;
  - Different timing in different virtual machines;
  - Uncontrollable garbage collection;
  - Not possible to discover queue sizes for shared resources;
  - Not possible to access system hardware;
  - Not possible to do space or timing analysis.

System design

- Design both the hardware and the software associated with system. Partition functions to either hardware or software.
- Design decisions should be made on the basis on non-functional system requirements.
- Hardware delivers better performance but potentially longer development and less scope for change.

R-T systems design process

1. Identify the stimuli to be processed and the required responses to these stimuli.
2. For each stimulus and response, identify the timing constraints.
3. Aggregate the stimulus and response processing into concurrent processes. A process may be associated with each class of stimulus and response.
4. Design algorithms to process each class of stimulus and response. These must meet the given timing requirements.
5. Design a scheduling system which will ensure that processes are started in time to meet their deadlines.
6. Integrate using a real-time operating system.

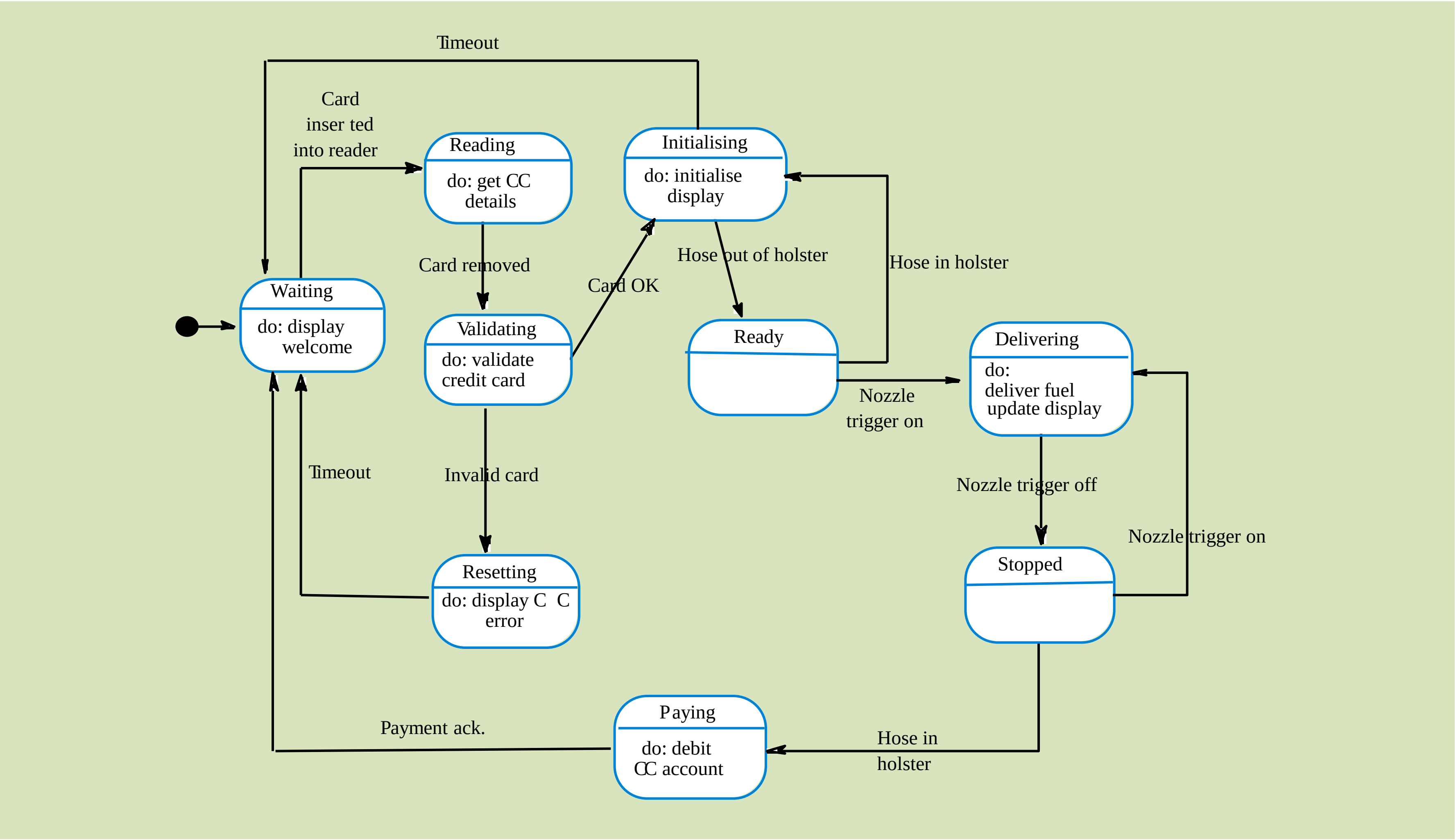
Timing constraints

- May require extensive simulation and experiment to ensure that these are met by the system.
- May mean that certain design strategies such as object-oriented design cannot be used because of the additional overhead involved.
- May mean that low-level programming language features have to be used for performance reasons.

Real-time system modelling

- The effect of a stimulus in a real-time system may trigger a transition from one state to another.
- Finite state machines can be used for modelling real-time systems.
- However, FSM models lack structure. Even simple systems can have a complex model.
- The UML includes notations for defining state machine models
- See Chapter 8 for further examples of state machine models.

Petrol pump state model



Real-time operating systems

- Real-time operating systems are specialised operating systems which manage the processes in the RTS.
- Responsible for process management and resource (processor and memory) allocation.
- May be based on a standard kernel which is used unchanged or modified for a particular application.
- Do not normally include facilities such as file management.

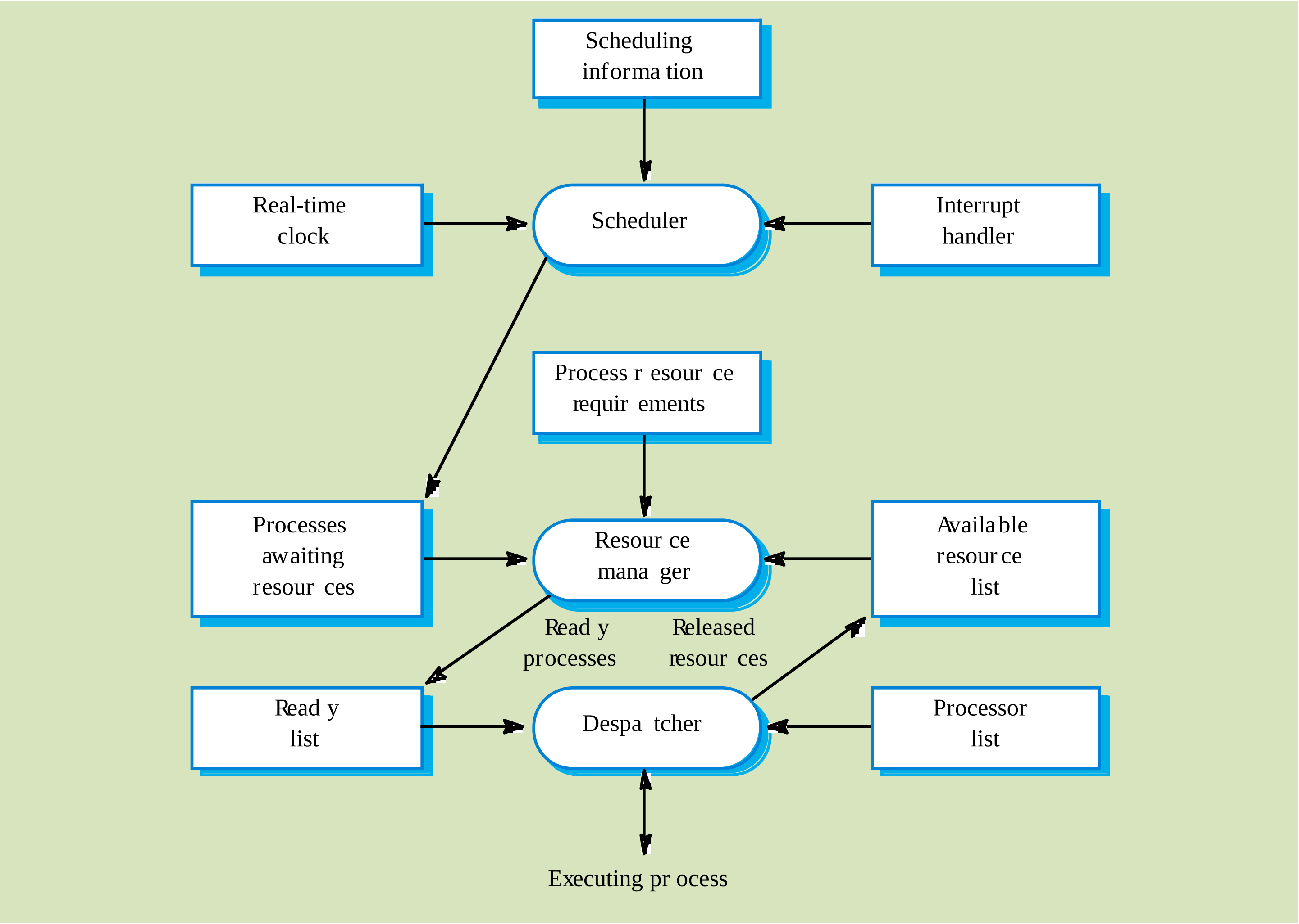
Operating system components

- Real-time clock
  - Provides information for process scheduling.
- Interrupt handler
  - Manages aperiodic requests for service.
- Scheduler
  - Chooses the next process to be run.
- Resource manager
  - Allocates memory and processor resources.
- Dispatcher
  - Starts process execution.

Non-stop system components

- Configuration manager
  - Responsible for the dynamic reconfiguration of the system software and hardware. Hardware modules may be replaced and software upgraded without stopping the systems.
- Fault manager
  - Responsible for detecting software and hardware faults and taking appropriate actions (e.g. switching to backup disks) to ensure that the system continues in operation.

Real-time OS components



Process priority

- The processing of some types of stimuli must sometimes take priority.
- Interrupt level priority. Highest priority which is allocated to processes requiring a very fast response.
- Clock level priority. Allocated to periodic processes.
- Within these, further levels of priority may be assigned.

Interrupt servicing

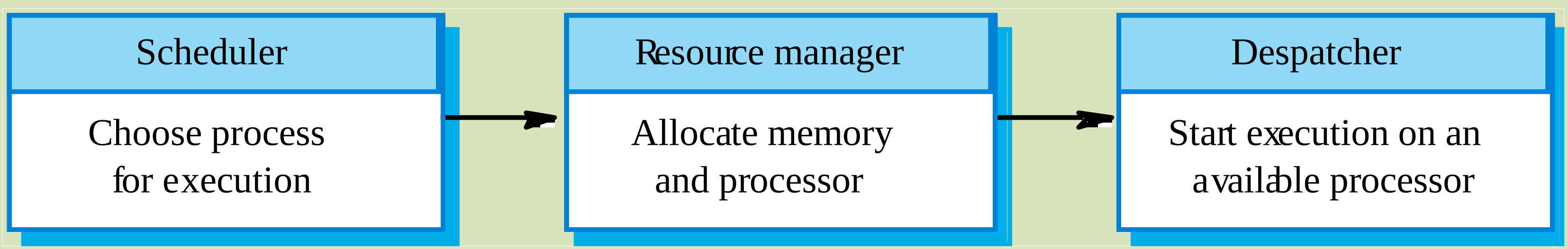
- Control is transferred automatically to a pre-determined memory location.
- This location contains an instruction to jump to an interrupt service routine.
- Further interrupts are disabled, the interrupt serviced and control returned to the interrupted process.
- Interrupt service routines MUST be short, simple and fast.

Periodic process servicing

- In most real-time systems, there will be several classes of periodic process, each with different periods (the time between executions), execution times and deadlines (the time by which processing must be completed).
- The real-time clock ticks periodically and each tick causes an interrupt which schedules the process manager for periodic processes.
- The process manager selects a process which is ready for execution.

Process management

- Concerned with managing the set of concurrent processes.
- Periodic processes are executed at pre-specified time intervals.
- The RTOS uses the real-time clock to determine when to execute a process taking into account:
  - Process period - time between executions.
  - Process deadline - the time by which processing must be complete.



Process switching

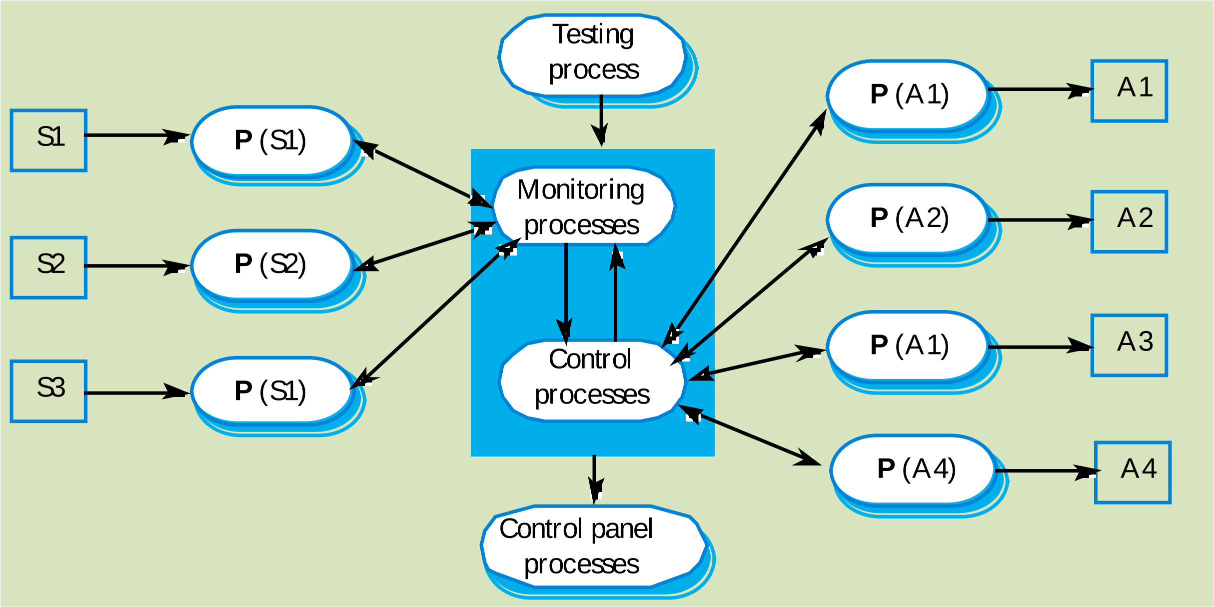
- The scheduler chooses the next process to be executed by the processor. This depends on a scheduling strategy which may take the process priority into account.
- The resource manager allocates memory and a processor for the process to be executed.
- The dispatcher takes the process from ready list, loads it onto a processor and starts execution.

Scheduling strategies

- Non pre-emptive scheduling
  - Once a process has been scheduled for execution, it runs to completion or until it is blocked for some reason (e.g. waiting for I/O).
- Pre-emptive scheduling
  - The execution of an executing processes may be stopped if a higher priority process requires service.
- Scheduling algorithms
  - Round-robin;
  - Rate monotonic;
  - Shortest deadline first.

Monitoring and control systems

- Important class of real-time systems.
- Continuously check sensors and take actions depending on sensor values.
- Monitoring systems examine sensors and report their results.
- Control systems take sensor values and control hardware actuators.



Key points

- Real-time system correctness depends not just on what the system does but also on how fast it reacts.
- A general RT system model involves associating processes with sensors and actuators.
- Real-time systems architectures are usually designed as a number of concurrent processes.
- Real-time operating systems are responsible for process and resource management.
- Monitoring and control systems poll sensors and send control signal to actuators.