SE350 – Project Overview
Second Part

Outline

1. Requirements and Assumption
2. Processor Management
3. Scheduling
4. Initialization

Basic Requirements & Assumptions

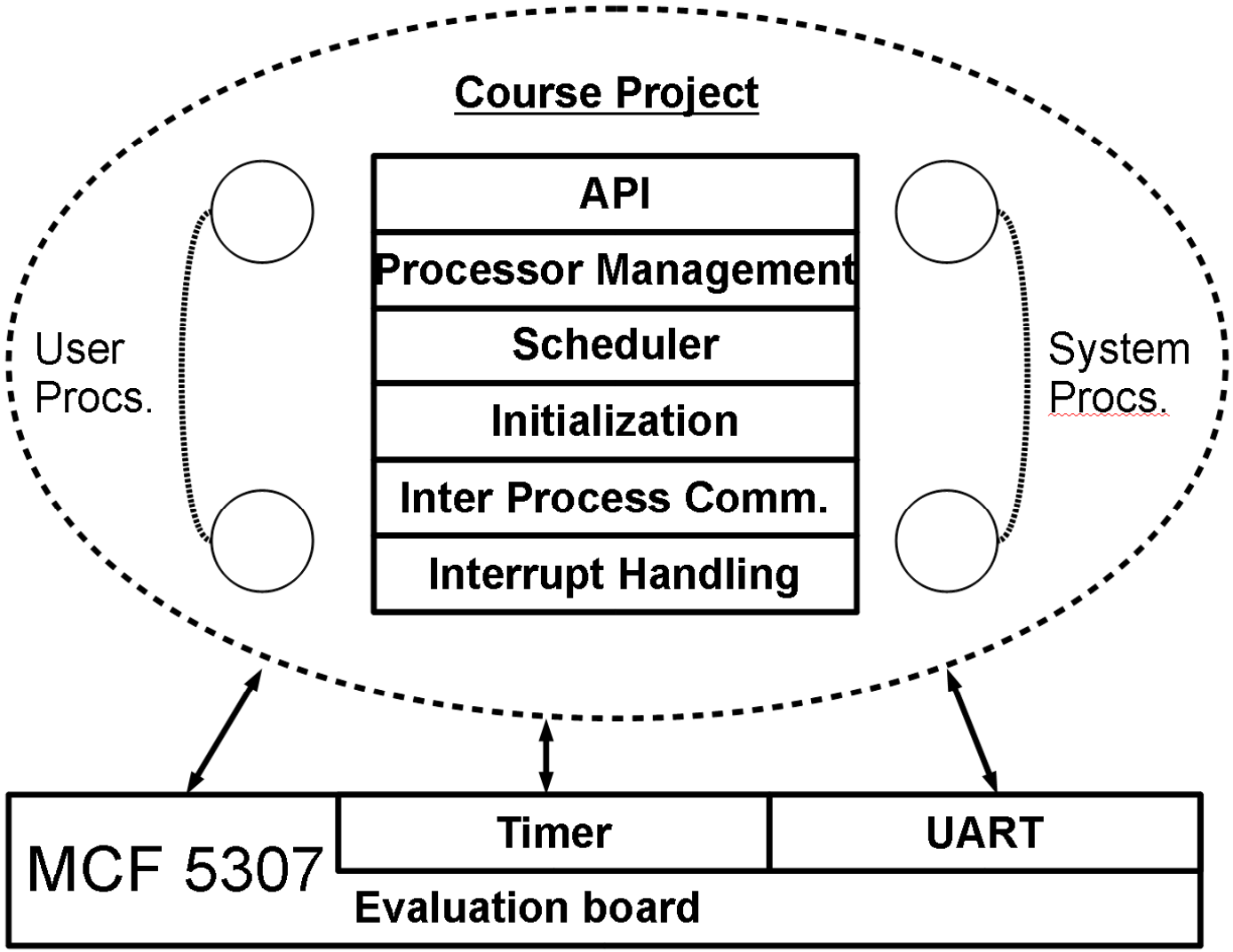
Basic Requirements:

- Preemptive scheduling
- Managing processes that are only created once
- Fixed priority scheduling
- Message-based inter-process Communication (i.e., send , receive messages)
- Memory management (i.e., request, release of memory blocks)

Simplifying Assumptions:

- All processes are known (at start-up; know each other)
- Processes are non-malicious

Functional Overview

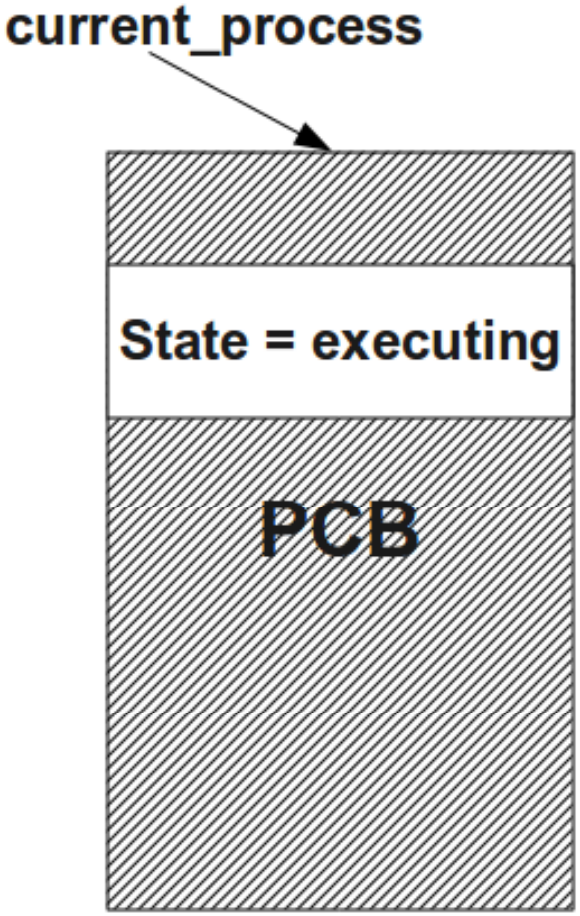


Processor Management: Process Data

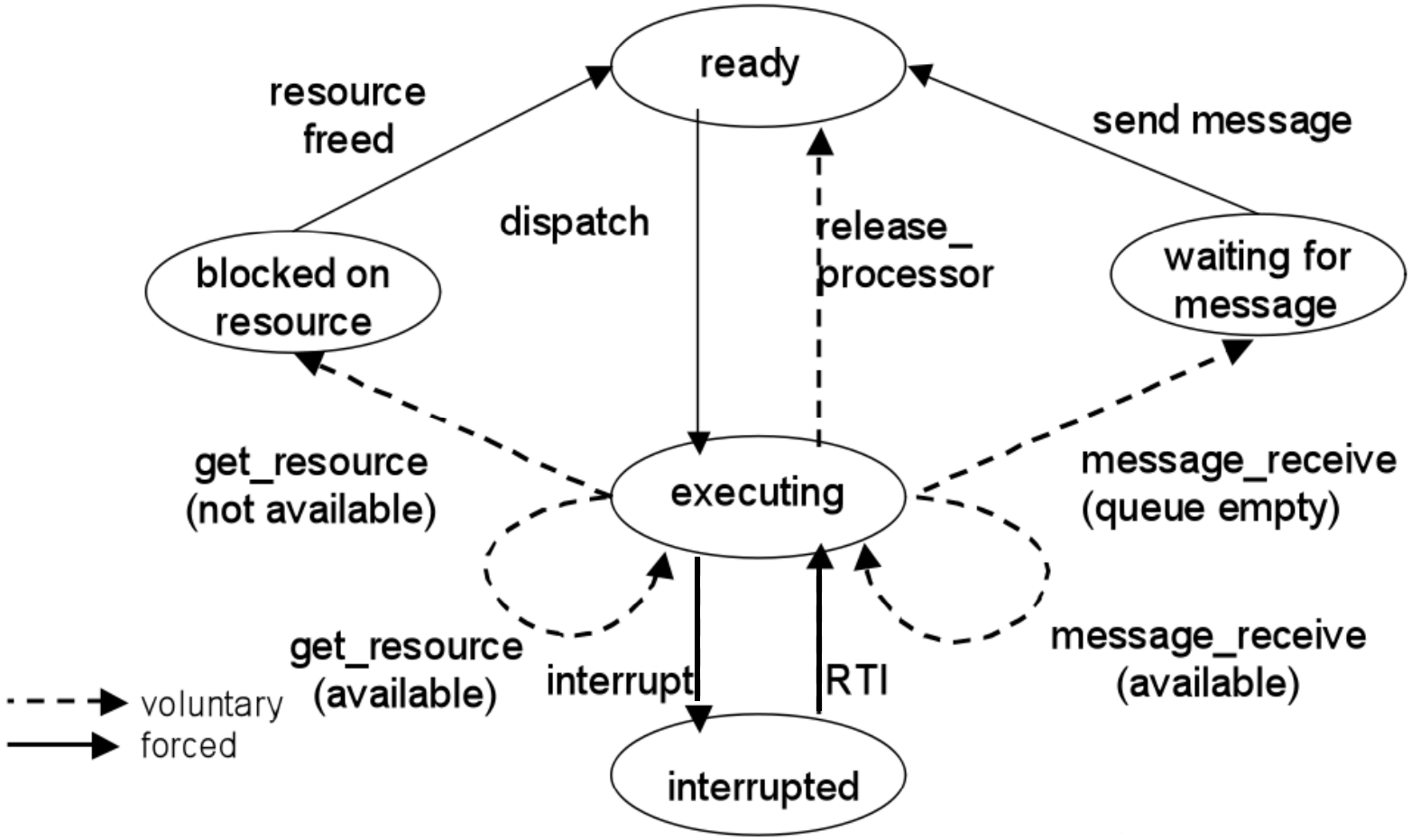
- Process control block (PCB):
- Needed for each process
- Describes status and context of process

current_process variable:

- OS must know, which process currently executes
- … always refers to PCB of currently executing process



Processor Management: Process States



Processor Management: Process Switching

Process switching:

1. Remove currently executing process from CPU
2. Select next process to execute using scheduler
3. Invoke context switch to new process

Context switching:

1. Save context of currently executing process
2. Update `current_process` to new process
3. Set state of new process to executing
4. Restore context of `current_process`
5. Execute `current_process`

Scheduling

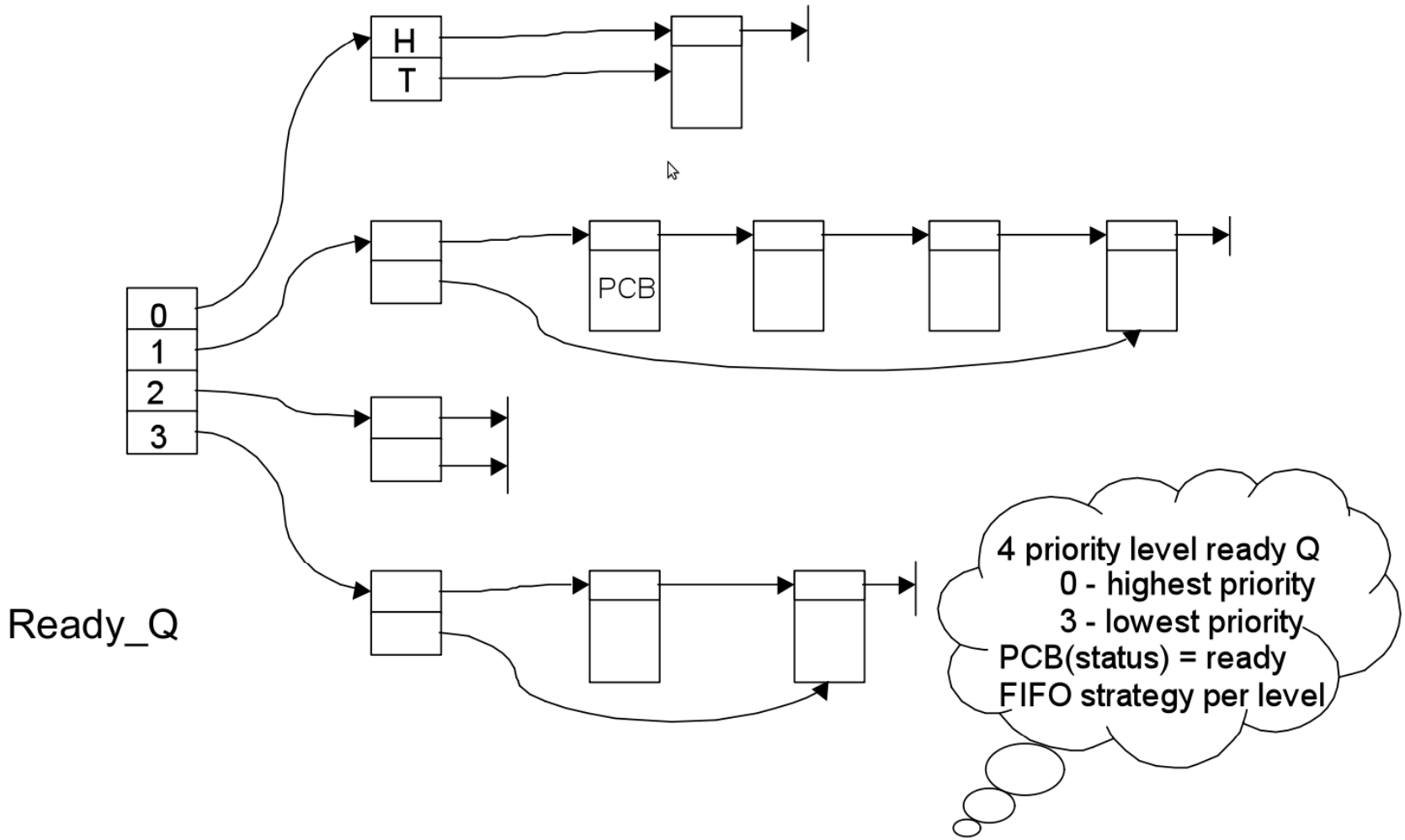
Requirements:

- Fixed, priority-based scheduling
- Each process has assigned priority
  - Highest priority process executes first
  - First-come-first-serve for processes of same priority

Procedure:

1. `process_switch()` invokes scheduler
2. Scheduler selects highest-priority ready process
3. `context_switch(new_proc)` lets the selected process execute

Scheduling: Ready Queues for Four Process Priorities



Scheduling: Null Process

- CPU must execute something
- What to do when ready queues are empty?
  - Possible solution: NULL process
  - Make sure that the ready queue is never empty
  - NULL has lowest priority and is always ready to run
- Basic example

```
void null_process() {
    while(1) {
            release_processor();
    }
}
```

Scheduling: release_processor()

- `release_processor()`

Basic Procedure:

1. Set `current_process` to state READY
2. `rpq_enqueue(current_process)`
   put current_process in ready queues
3. `process_switch()`
   invokes scheduler and context-switches to the new process

Initialization

- What operations need to be carried out at start-up?
- Initialize all hardware, incl.
  - Serial port(s) and timer(s)
  - Memory mapping (dynamic memory allocation for mem-blocks and stacks…)
  - Interrupts (hardware and software: vector table & traps)
- Create all kernel data structures
  - PCBs (status=ready), queues…
  - Place PCBs into respective queues
  - Start first ready process (i.e. `process_switch()`)

Initialization: Initialization Table

- How does RTX know which processes to create?

- Initialization Table:
- Array of records
- Each record contains spec of its process and additional data structures