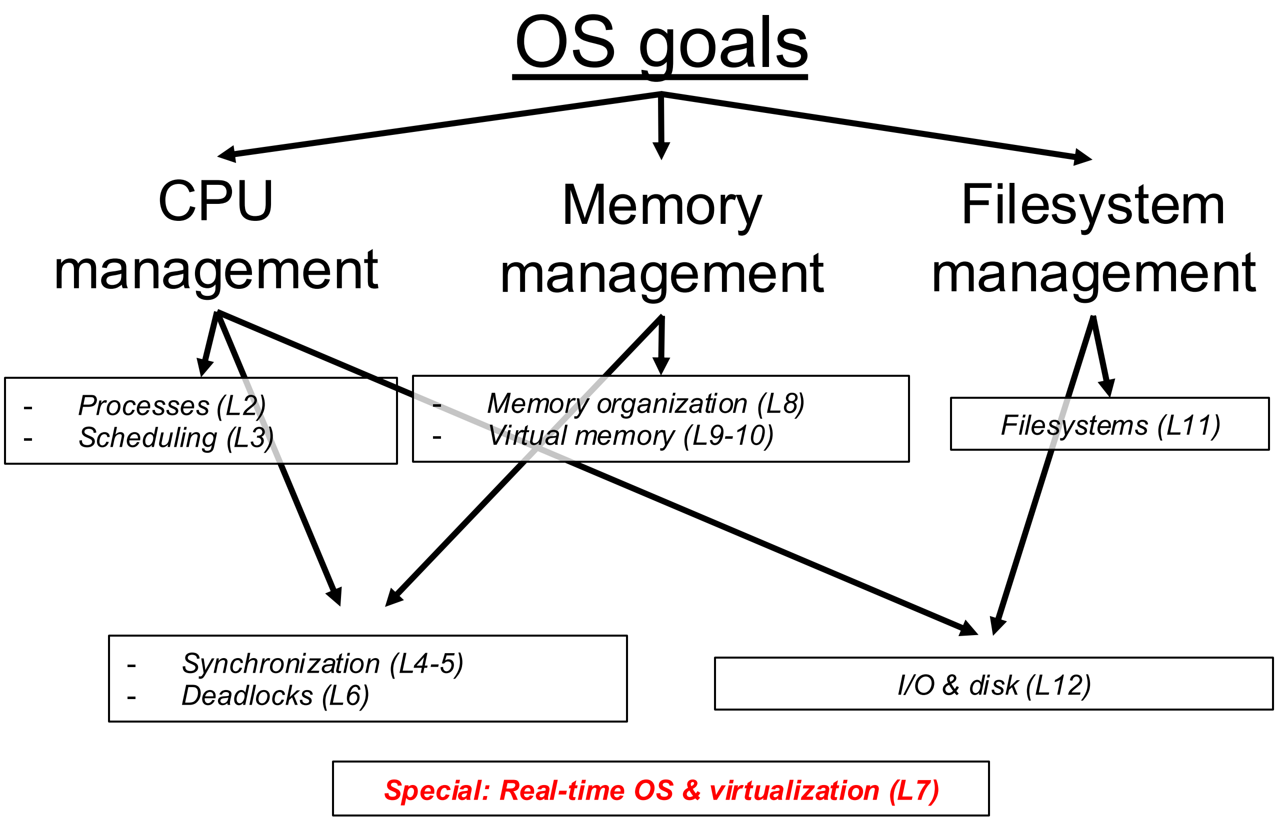
Course Overview



Part 6: Real-Time OS & Virtualization

- **What is a Real-Time OS (RTOS)?**
- Real-Time Process Specification
- Real-Time CPU Scheduling
- Virtualization

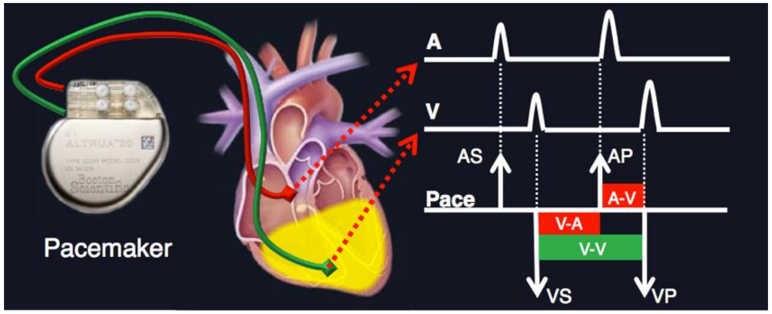
Cyber-Physical Systems

- Physical/Engineered systems whose operations are monitored, coordinated and controlled by a reliable computing and communication core
  - Automotive Systems (Autonomous driving, Parking assist, Airbag controls)
  - Avionics (Flight navigation & control)
  - Manufacturing Systems (Robotics, Process controls)
  - Medical Systems (Robotic surgery, devices)
  - …

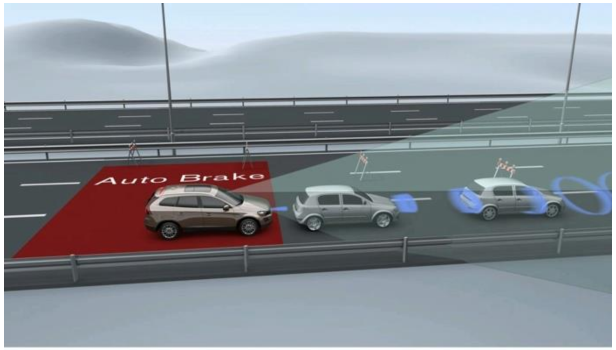
Relevance of Real-Time

- Common to many application examples we saw in the previous slide:
  - Collect data from various sensing devices
  - Execute control law(s) to determine response
  - Send actuator commands in a **reasonable amount of time**

Pacemaker timing diagram



Collision avoidance and braking



Collision Warning with Auto Brake

What is a Reasonable Time?

- What is the functionality?
  - Collision avoidance in automotive (milliseconds)
  - Pacemaker (up to a second)
  - Robotic surgery (varies greatly depending on the target)
- What are the environment constraints?
  - Available computing and communication resources
  - Timing characteristics of sensors/actuators/operations
- Failure-mitigation strategies?
  - Time to detect and recover from failures
  - Example: execution replication for redundancy

Common Misconception

- Real-Time ≠ Fast
- Real-Time = Predictable even in the worst-case

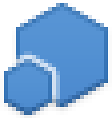
“Man drowned in a river with average depth 20 cms”

Check Your Understanding

Real-Time CPS / Real-Time OS

- Definition: System whose correctness depends not only on the logical/functional aspects, but also on the temporal aspects
  - Application has deadlines that must be met
  - **A real-time OS (RTOS) provides OS services to such systems (e.g., FreeRTOS, MicriumOS, …)**
- Key performance measure for RTOS
  - Timeliness/Predictability on timing constraints (deadlines)
  - Significance of worst-case over average-case
  - Deadlines are a function of application requirements

Check Your Understanding

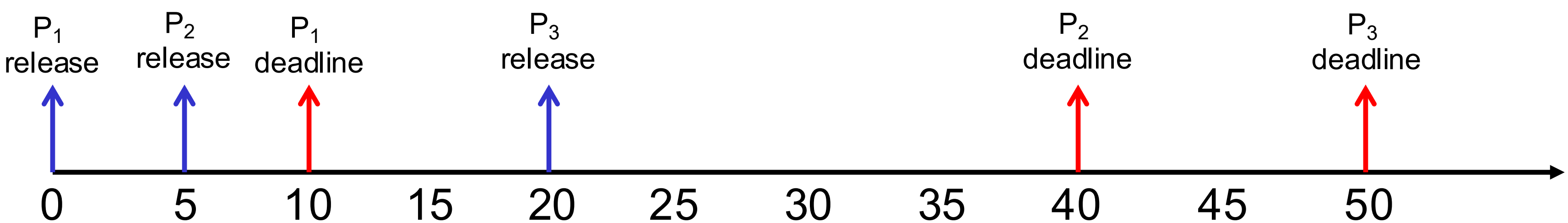


Part 5: Real-Time OS & Virtualization

- What is a Real-Time OS (RTOS)?
- **Real-Time Process Specification**
- Real-Time CPU Scheduling
- Virtualization

RTOS (Real-Time) Process

- **Definition:** A real-time process is specified as <R,C,D>, where R is process release time, C is execution requirement and D is relative deadline
  - Requires C time units of CPU in the interval [R, R+D)
  - **How does one determine these parameters?**
- Example: P_1<0,5,10>, P_2<5,10,35>, P_3<20,10,30>



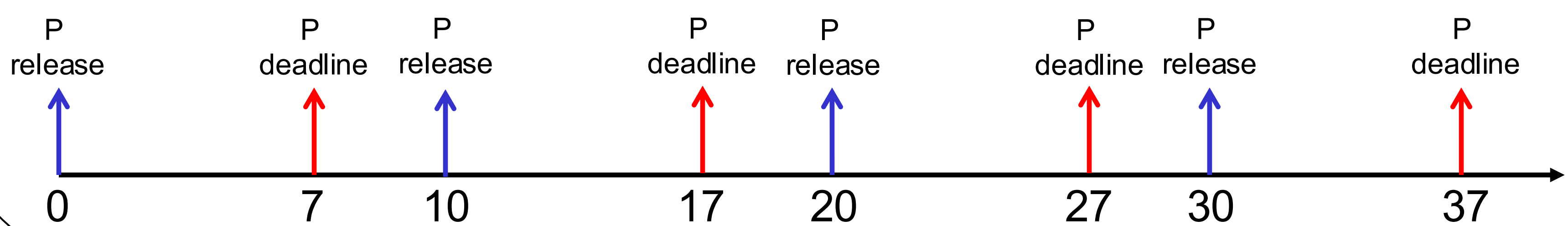
In this lecture, we assume processes only have a single CPU burst; C is the duration of this burst

Recurrent Real-Time Process

- Nature of real-time processes
  - Collect data from sensing devices, execute control laws to determine responses, and send actuator commands in reasonable time
  - **Repeat the above steps regularly**
    - Examples: airbag control, flight control, collision avoidance, pacemaker, etc.
- A recurrent real-time process
  - **Executes some function repeatedly over time**
  - Each instance of execution is a real-time process <R,C,D>

Periodic Real-Time Process

- Definition: A process that **repeats periodically**
  - Processes generated by a time-triggered phenomena (sensor sending data periodically)
  - Example: Perception function for collision detection
- A periodic process is specified as <T,C,D>, where T is process period, C & D are as defined earlier
  - Real-time processes are released at R=0, T, 2T, …
  - Example: Periodic process P<10, 5, 7>

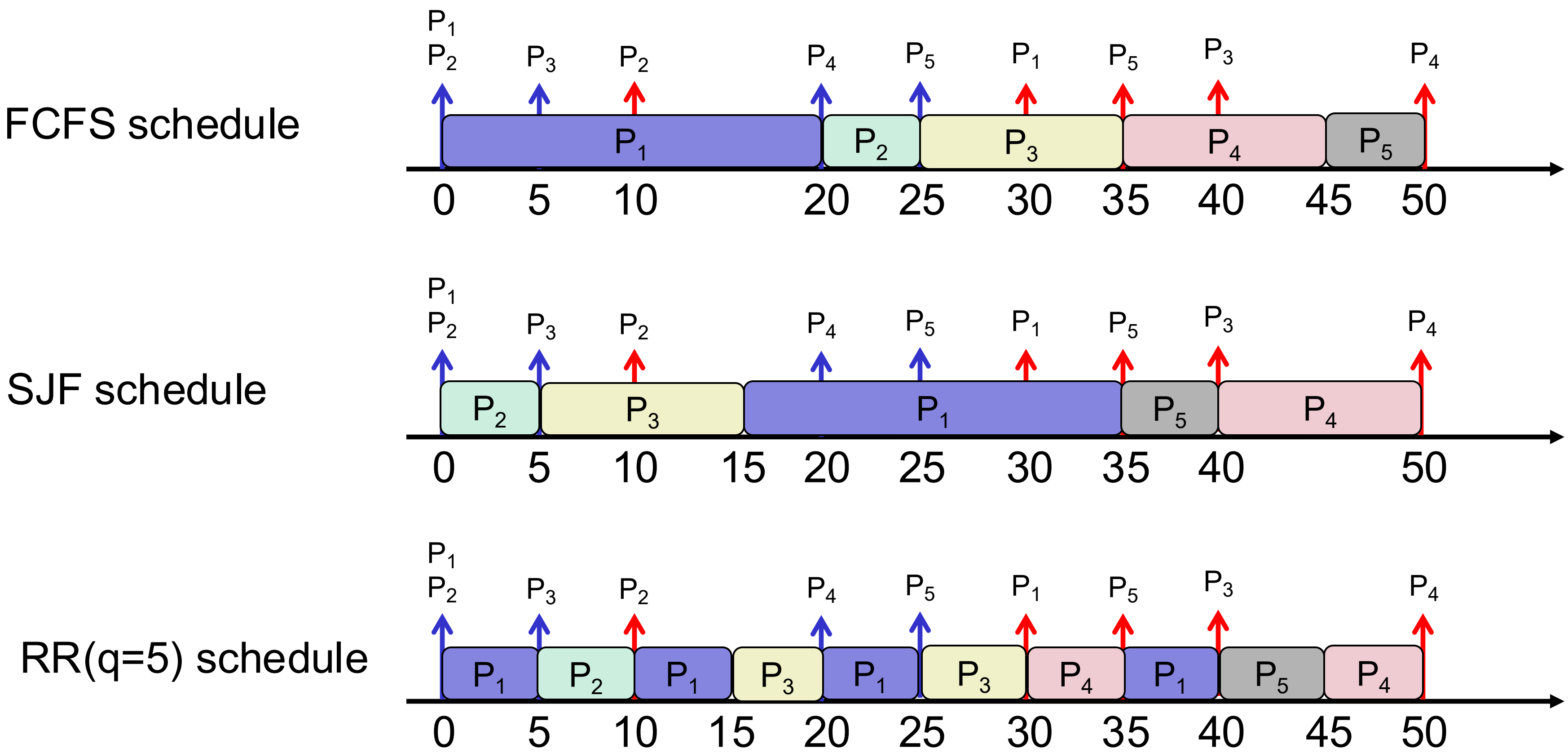


Part 5: Real-Time OS & Virtualization

- What is a Real-Time OS (RTOS)?
- Real-Time Process Specification
- **Real-Time CPU Scheduling (short-term scheduler)**
  - **Fixed-priority scheduling**
  - **Dynamic-priority scheduling**
- Virtualization

Why Classic Algorithms Fail?

- Consider real-time processes (non-recurring): $P_1<0,20,30>$, $P_2<0,5,10>$, $P_3<5,10,35>$, $P_4<20,10,30>$, $P_5<25,5,10>$

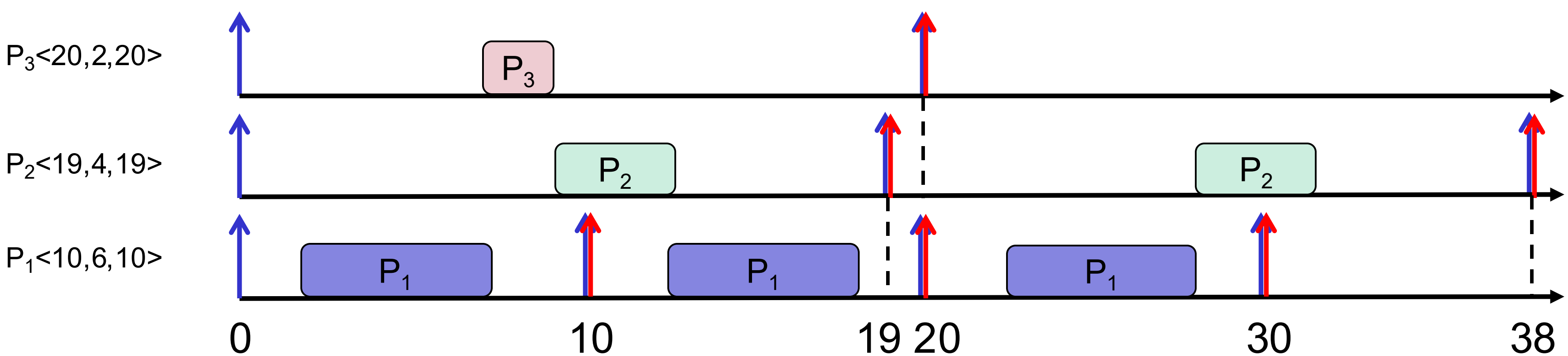


They don't prioritize deadlines and hence perform poorly

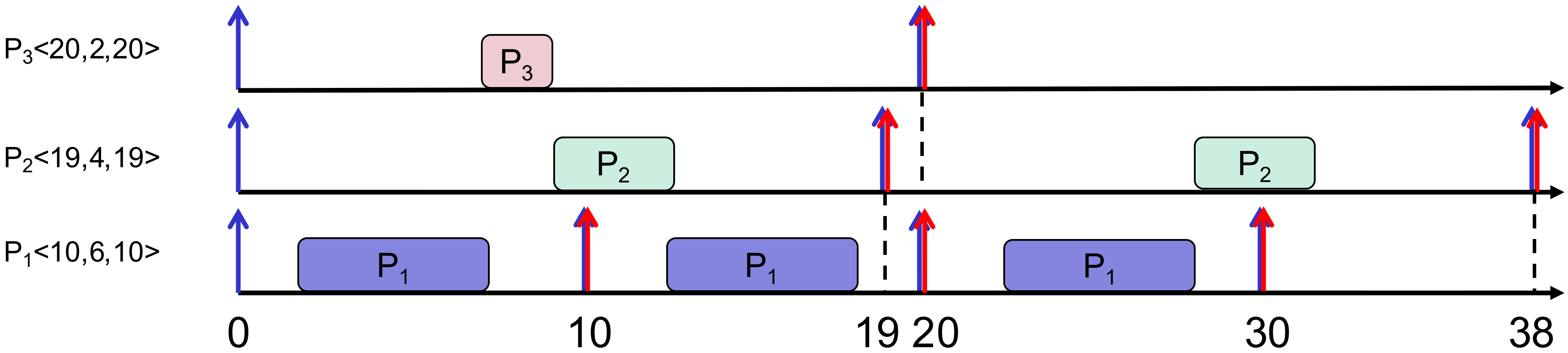
Real-Time CPU Scheduling Problem

Given a set of periodic/sporadic real-time processes, find a uni-processor CPU scheduling algorithm that can meet process deadlines

- We will use a running example (periodic real-time process set): P_1<10,6,10>, P_2<19,4,19>, P_3<20,2,20>



Fixed-Priority CPU Scheduling

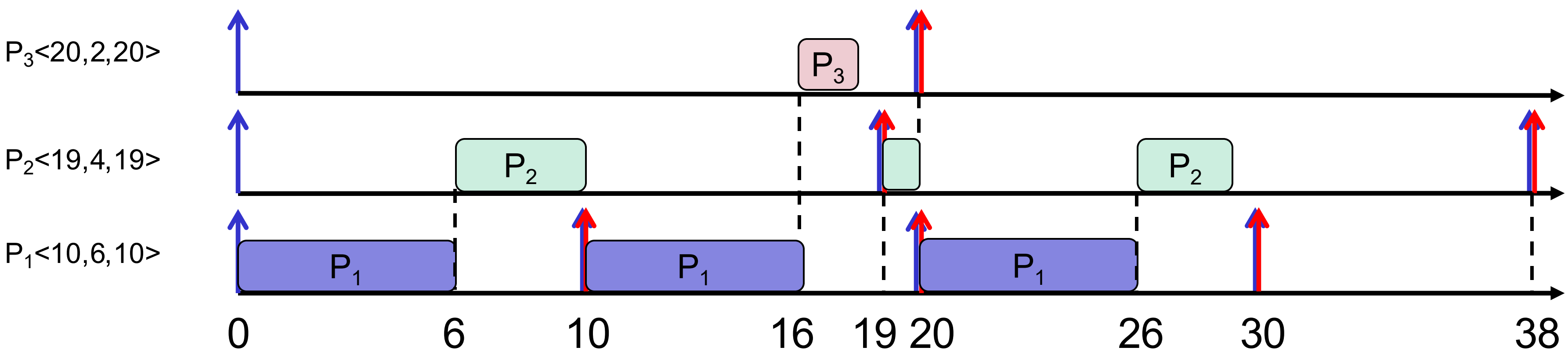


Priorities are fixed for each recurrent process

- **Priorities are fixed across instances of recurrent processes**
- Suppose instance of $P_1(R=0)$ has higher priority than instance of $P_2(R=0)$. Then,
  - $P_1(R=10)$ has higher priority than $P_2(R=0)$
  - $P_1(R=10)$ has higher priority than $P_2(R=19)$
  - $P_1(R=20)$ has higher priority than $P_2(R=19)$ …

Rate Monotonic (RM) Scheduler

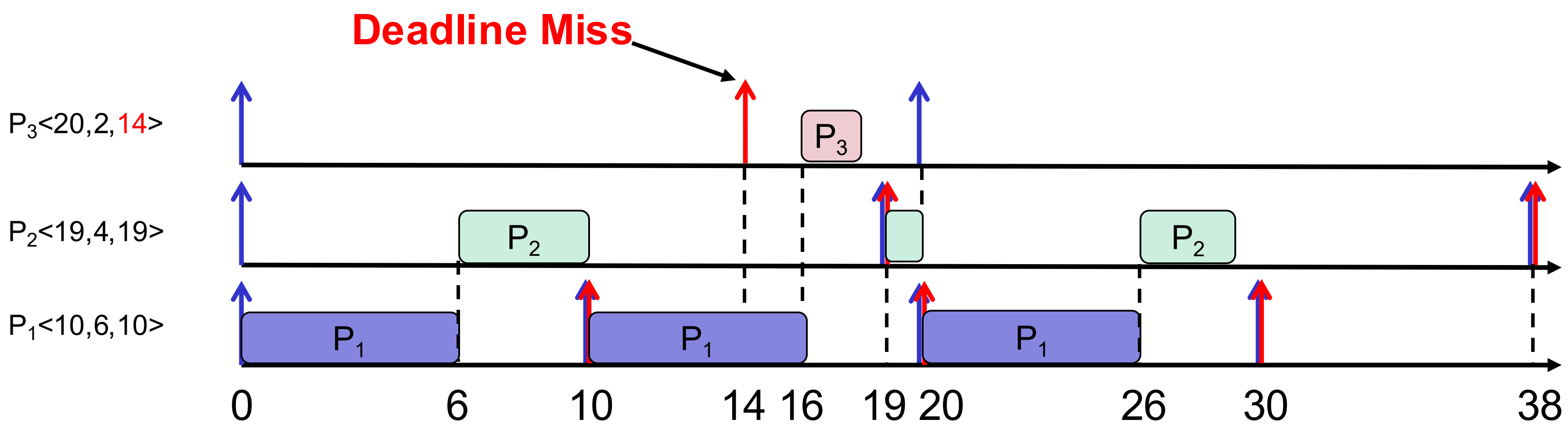
- Assign priorities based on process periods / minimum release-separation time (T)
  - Shorter T implies higher priority
  - Ties are broken arbitrarily



RM is a very popular short-term CPU scheduler in the real-time CPS industry. Why?

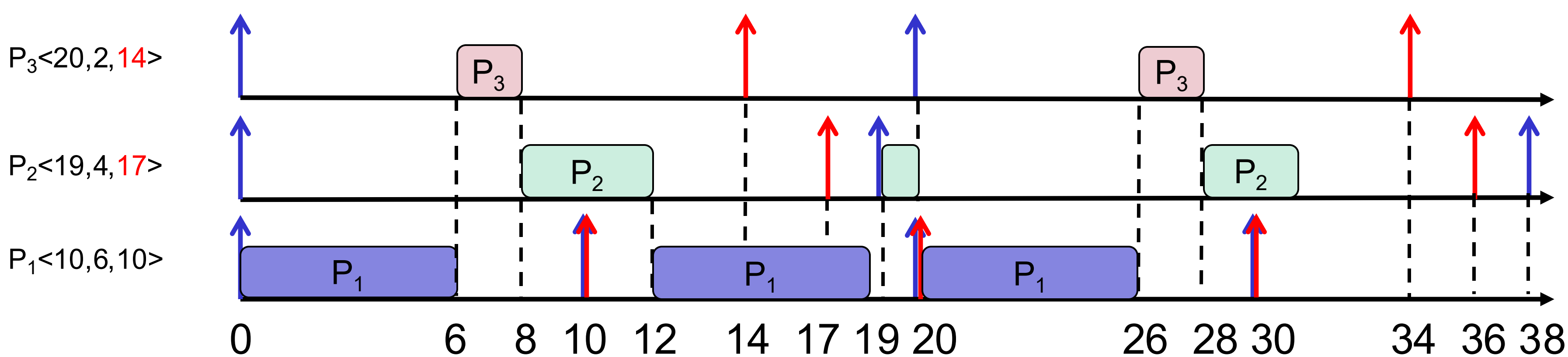
RM and Process Deadlines

- RM is good, but still does not always prioritize urgent processes
  - Suppose we modify the process set as follows: P_1<10,6,10>, P_2<19,4,19>, P_3<20,2,14>



Earliest Deadline First (EDF) Scheduler

- Dynamic-priority scheduler that assigns priorities based on process **instance deadlines**
  - Instances with shorter deadline are given higher priority
    - **NOT the same as parameter D**
  - Ties are broken arbitrarily



EDF is a dynamic-priority scheduler, hence more powerful than RM

Check Your Understanding

RM/DM versus EDF

RM

- Simpler implementation
  (separate queue for each recurrent process)
- Predictability for high priority processes, even under high load

EDF

- Harder implementation
  (online sorting of queue based on instance deadlines)
- Can be unpredictable under high load

Part 5: Real-Time OS & Virtualization

- What is a Real-Time OS (RTOS)?
- Real-Time Process Specification
- Real-Time CPU Scheduling
- **Virtualization**