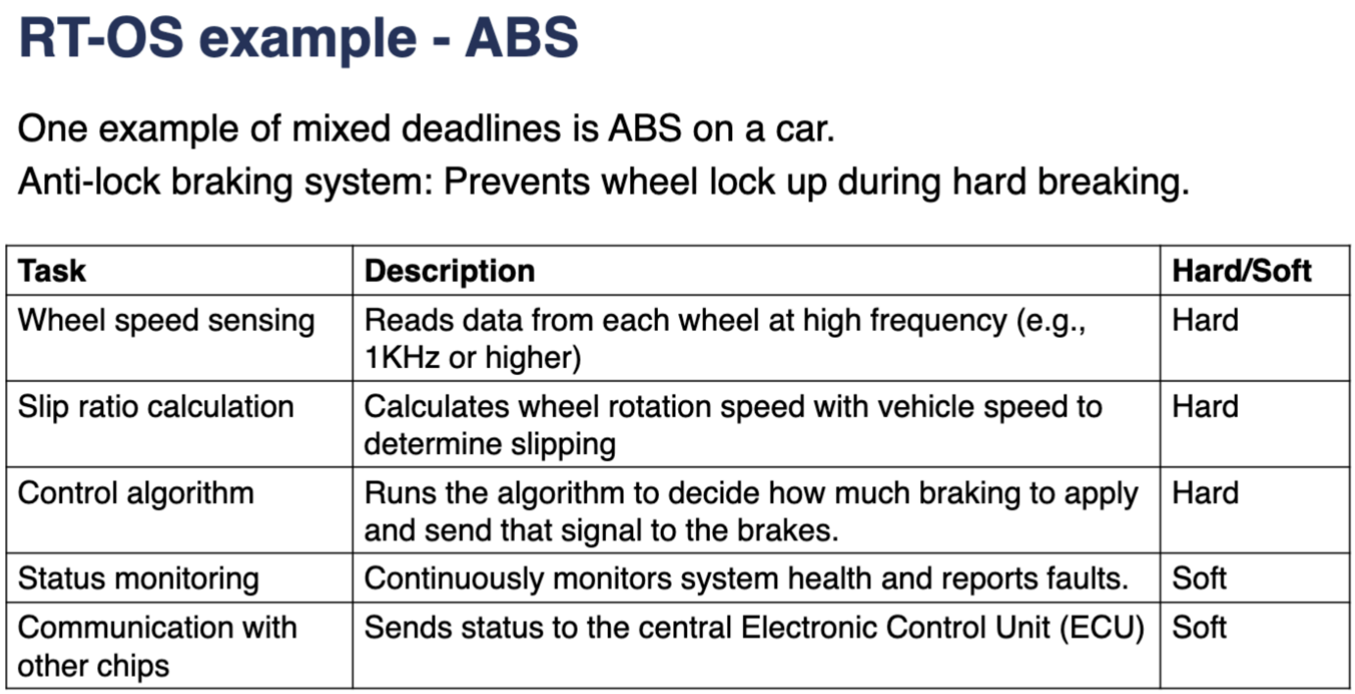
- time critical embedded systems
- pacemakers
    - micro-second level → deliver electric impulses to heard
- fight control systems
- power systems

Real Time OS

- RT → meeting strict deadlines
- NOT fast computing
    - precise timing and high degree of reliability
    - handle interrupts and sys exceptions timely
    - ensure critical sections of code are done within allocated time

Hard V.S. Soft RT Deadline

- hard → air bags, fly-by-wire
- soft → video streaming, home temperature monitor



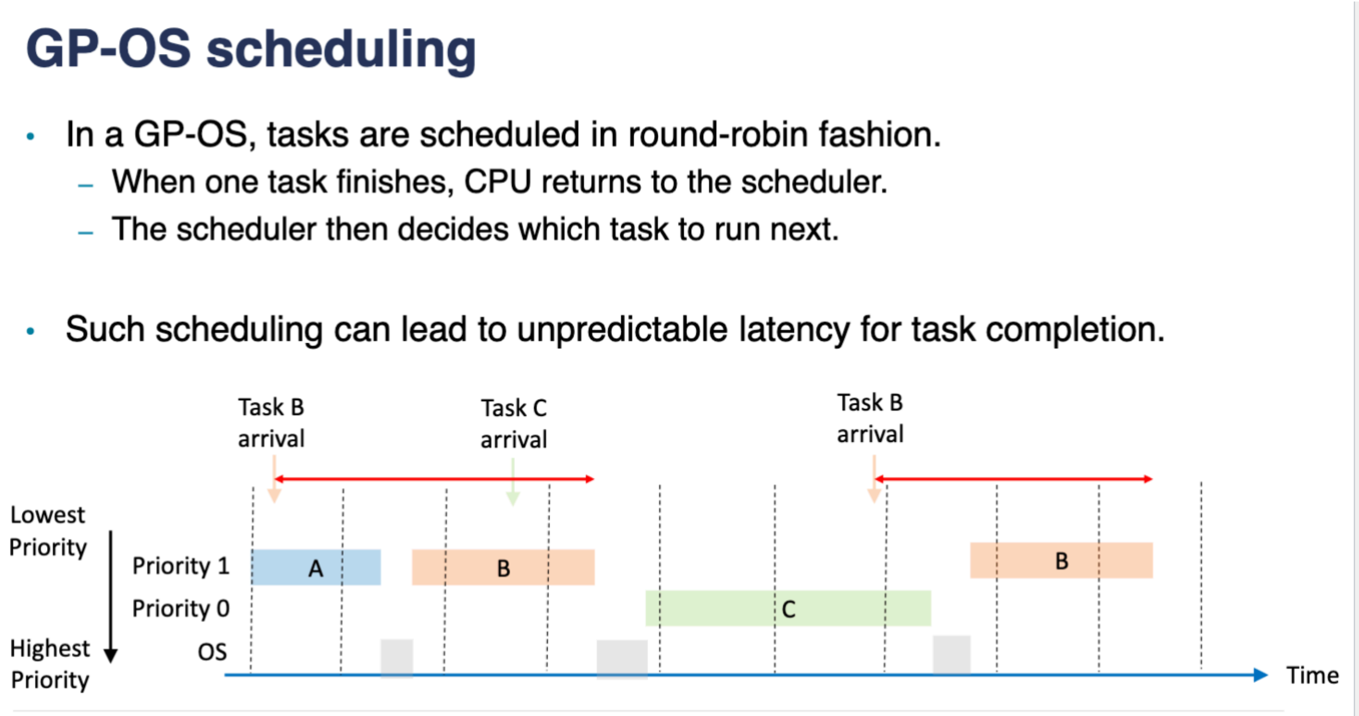
RT-OS example - ABS

One example of mixed deadlines is ABS on a car.
Anti-lock braking system: Prevents wheel lock up during hard breaking.

Task	Description	Hard/Soft
Wheel speed sensing	Reads data from each wheel at high frequency (e.g., 1KHz or higher)	Hard
Slip ratio calculation	Calculates wheel rotation speed with vehicle speed to determine slipping	Hard
Control algorithm	Runs the algorithm to decide how much braking to apply and send that signal to the brakes.	Hard
Status monitoring	Continuously monitors system health and reports faults.	Soft
Communication with other chips	Sends status to the central Electronic Control Unit (ECU)	Soft

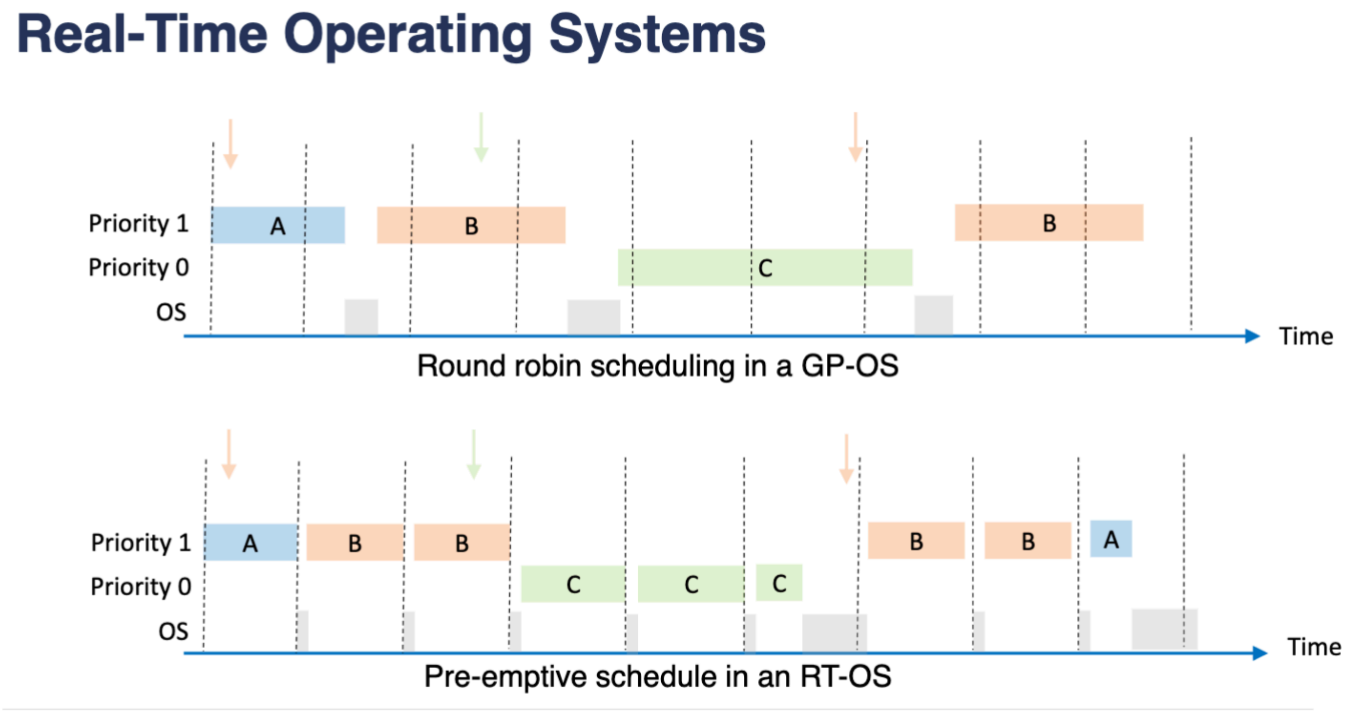
Comparison with GP-OS

- GP-OS focuses on high throughput (many applications running at the same time)
    - RR, runs when task finishes
- RT-OS → predictable latency (each task has a strict deadline)
    - priority based pre-emptive scheduling where higher priority tasks take precedence
    - runs at a fixed time interval, set with a hardware timer
- **scheduler difference**



GP-OS scheduling

- In a GP-OS, tasks are scheduled in round-robin fashion.
  - When one task finishes, CPU returns to the scheduler.
  - The scheduler then decides which task to run next.
- Such scheduling can lead to unpredictable latency for task completion.

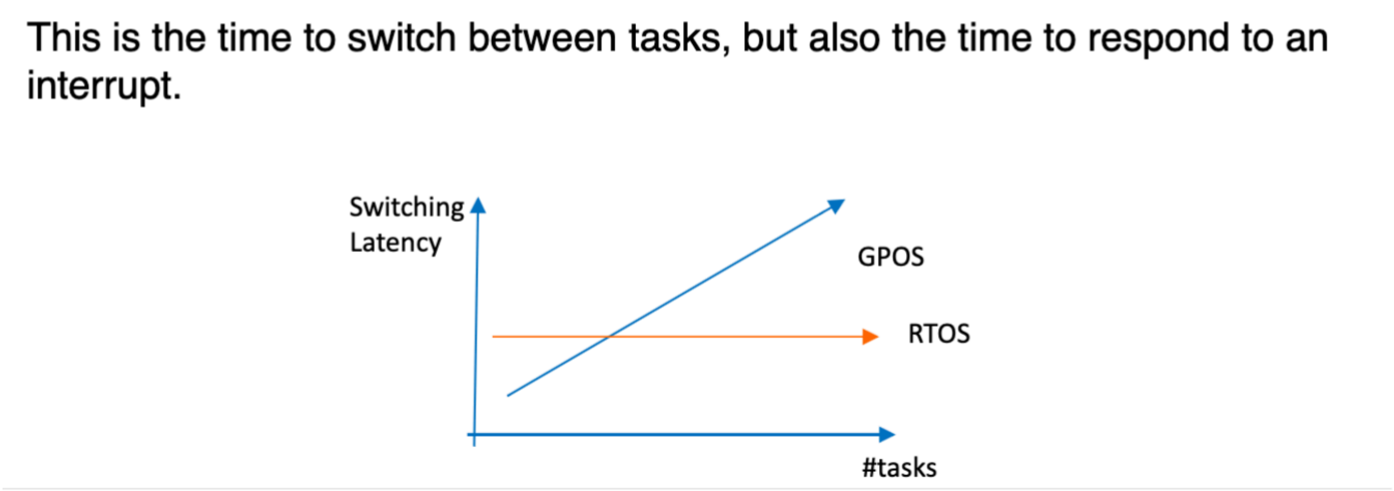


Real-Time Operating Systems

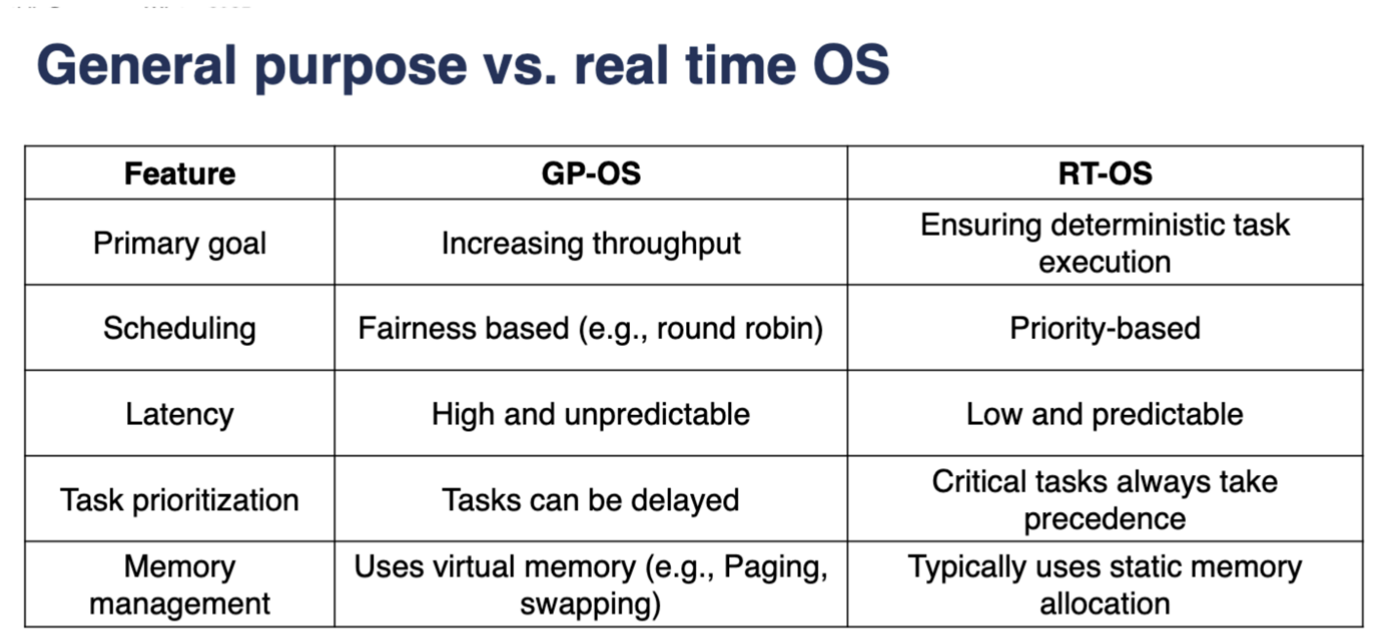
Round robin scheduling in a GP-OS

Pre-emptive schedule in an RT-OS

Context Switching



This is the time to switch between tasks, but also the time to respond to an interrupt.



General purpose vs. real time OS

Feature	GP-OS	RT-OS
Primary goal	Increasing throughput	Ensuring deterministic task execution
Scheduling	Fairness based (e.g., round robin)	Priority-based
Latency	High and unpredictable	Low and predictable
Task prioritization	Tasks can be delayed	Critical tasks always take precedence
Memory management	Uses virtual memory (e.g., Paging, swapping)	Typically uses static memory allocation

Bare Metal V.S. RTOS

- bare metal
  - super-loop approach, every task is run one after the other
  - each task can take different amounts of time to finish, no way to guarantee a fixed latency for any of them
- RT-OS
  - create **tasks** and hand over duties to OS scheduler
  - task priorities and task deadlines are specified at time of task creation
  - programmer must make sure the hardware is capable of completing all the required tasks in the specified time