K22 - Operating Systems: Design Principles and Internals

Fall 2025 @dit

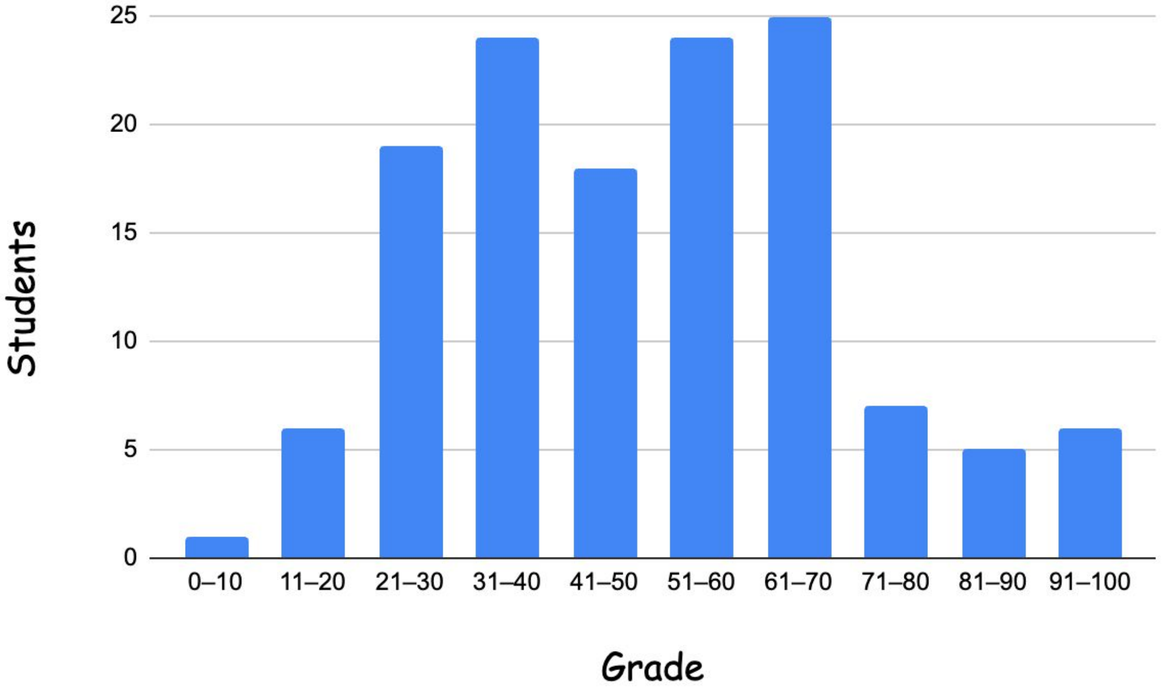
Vaggelis Atlidakis

Lecture 15

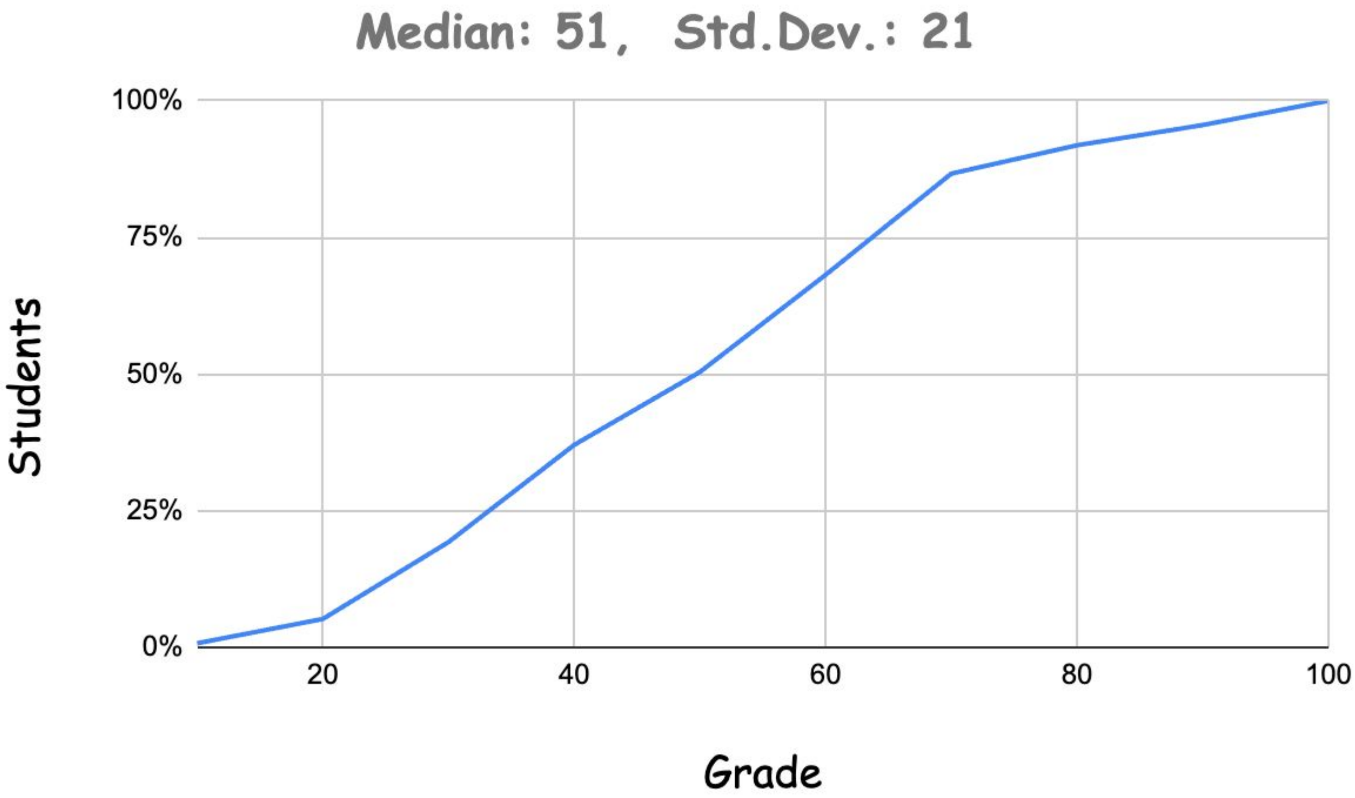
References: Similar OS courses @Columbia, @Stanford, @UC San Diego, @Brown, @di (previous years); and textbooks: Operating Systems: Three Easy Pieces, Operating Systems: Principles and Practice, Operating System Concepts, Linux Kernel Development, Understanding the Linux Kernel

Midterm

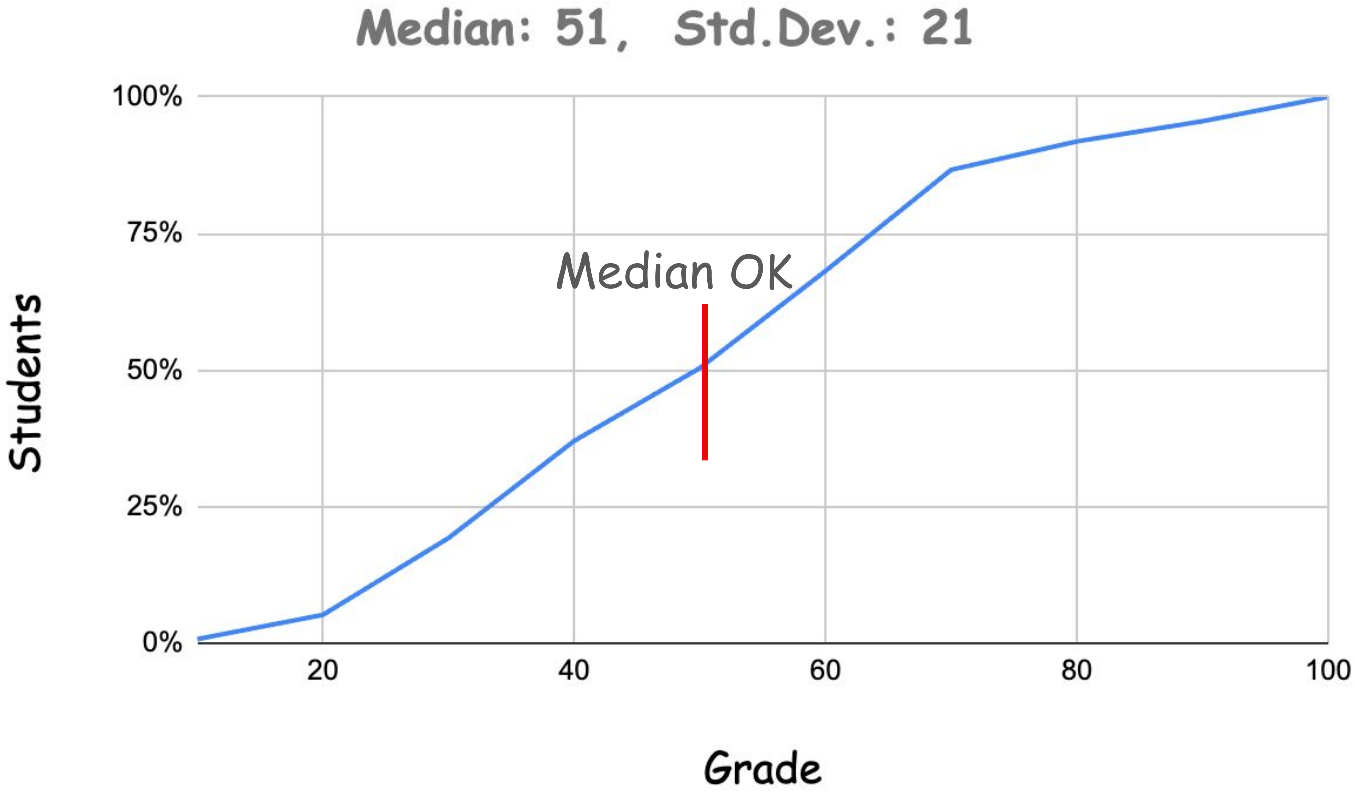
Midterm (median 51 / std.dev.: 21)



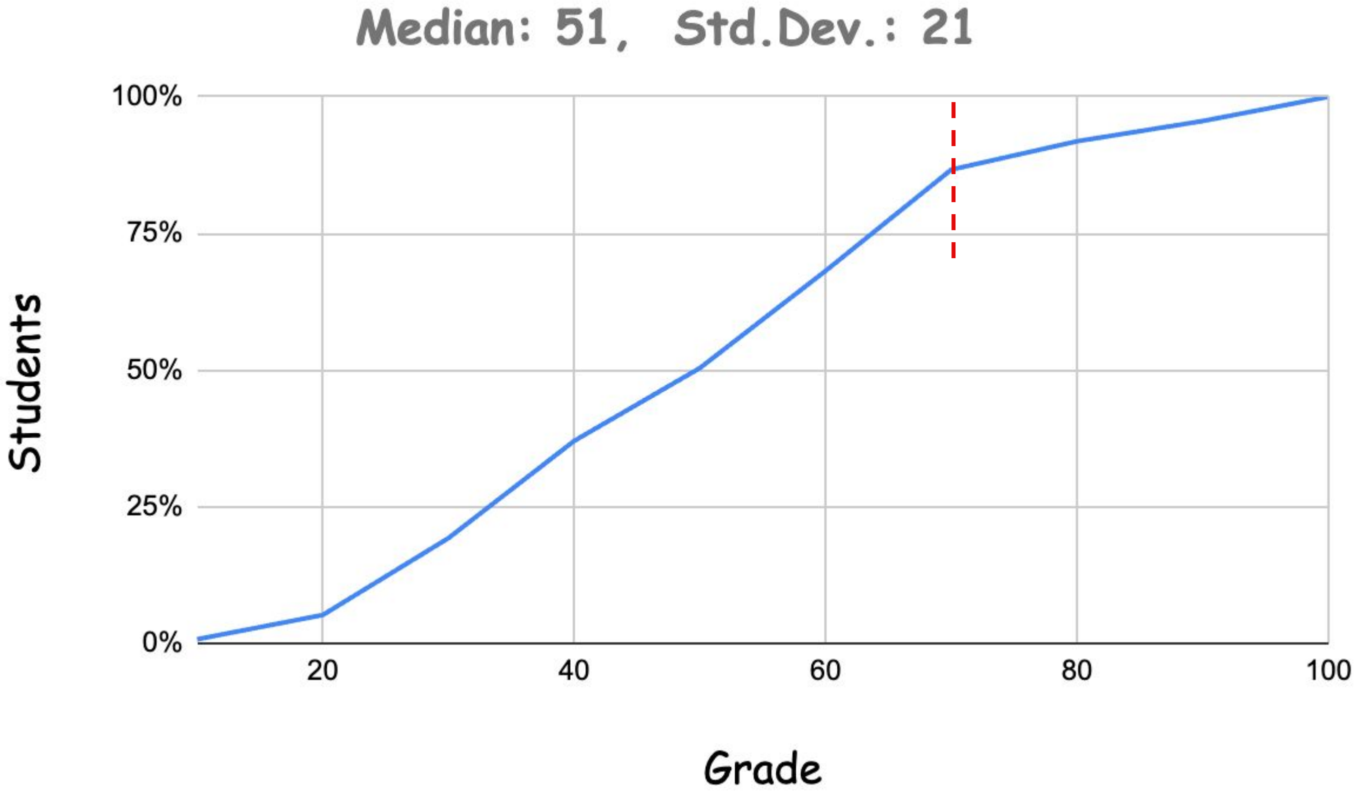
Midterm (median 51 / std.dev.: 21)



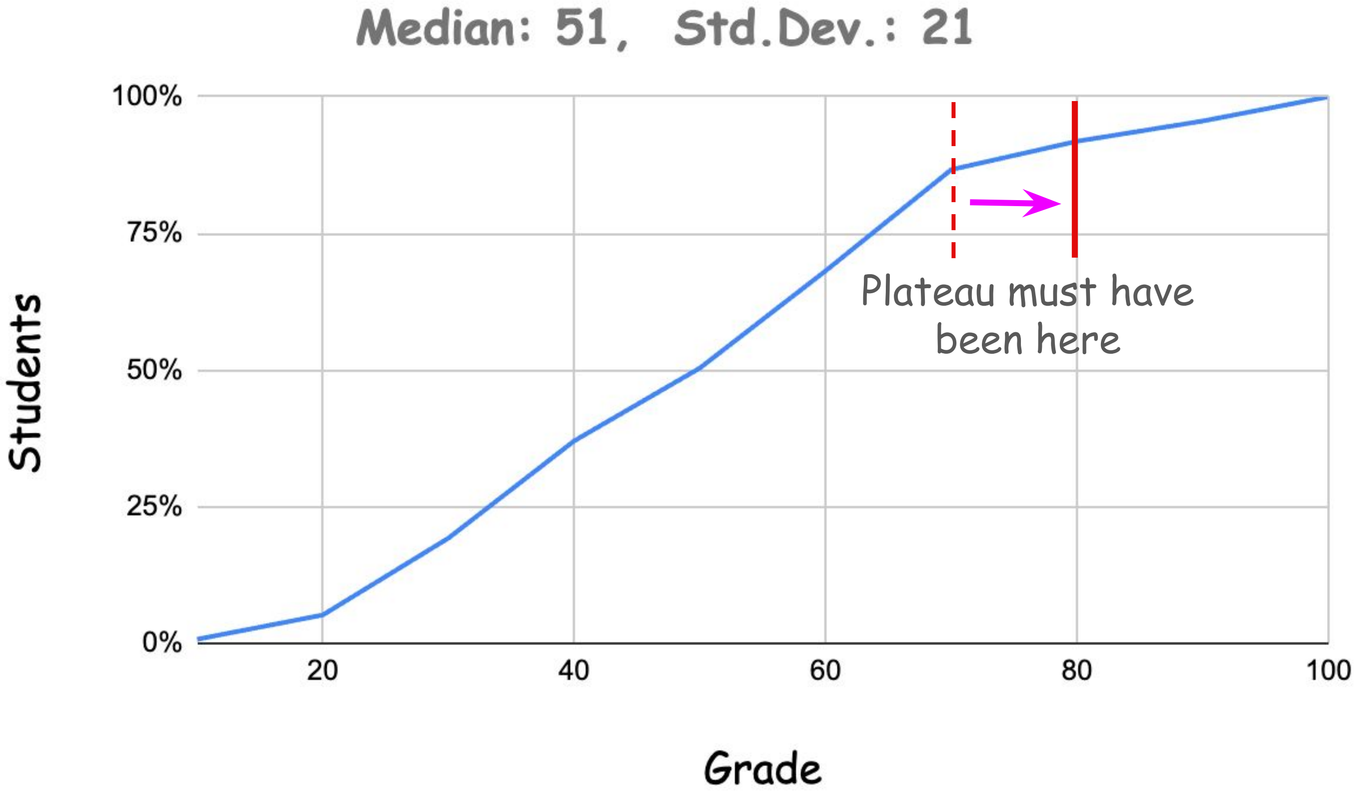
Midterm (median 51 / std.dev.: 21)



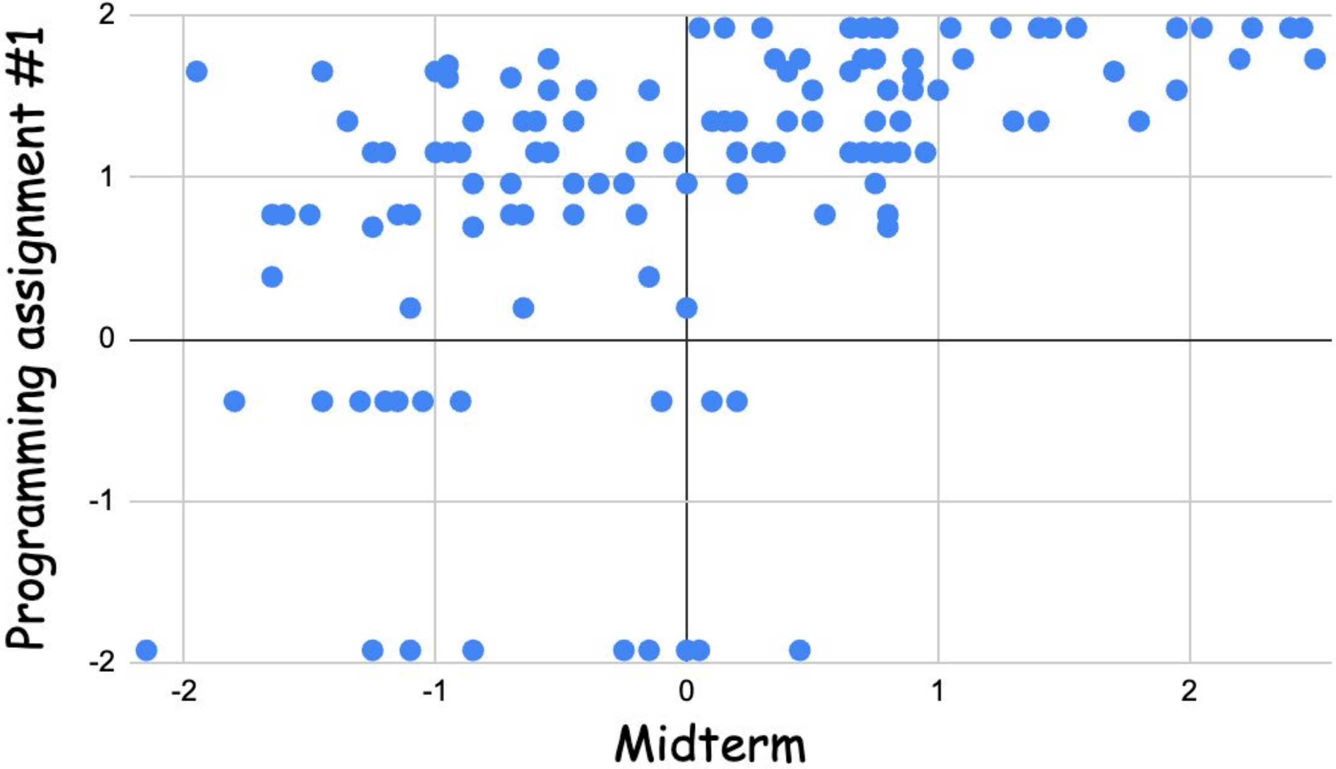
Midterm (median 51 / std.dev.: 21)



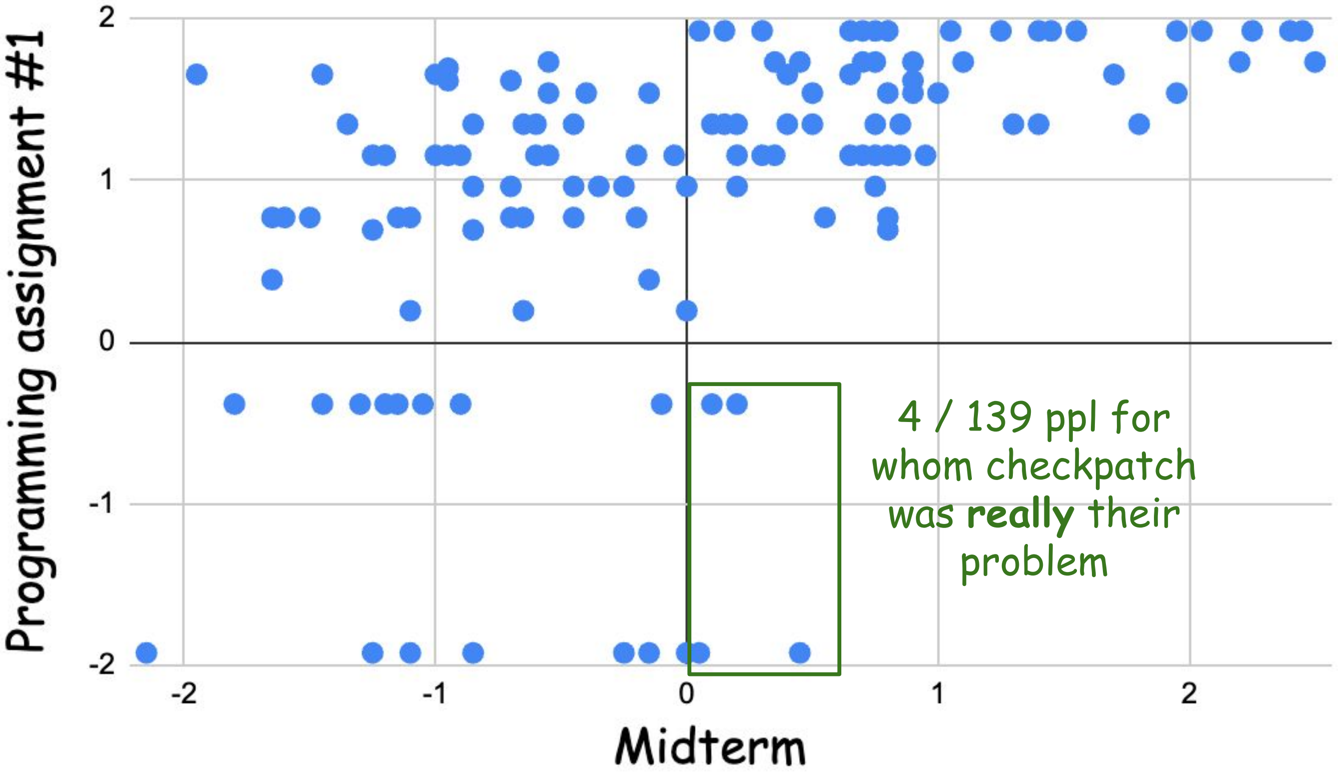
Midterm (median 51 / std.dev.: 21)



Midterm (median 51 / std.dev.: 21)

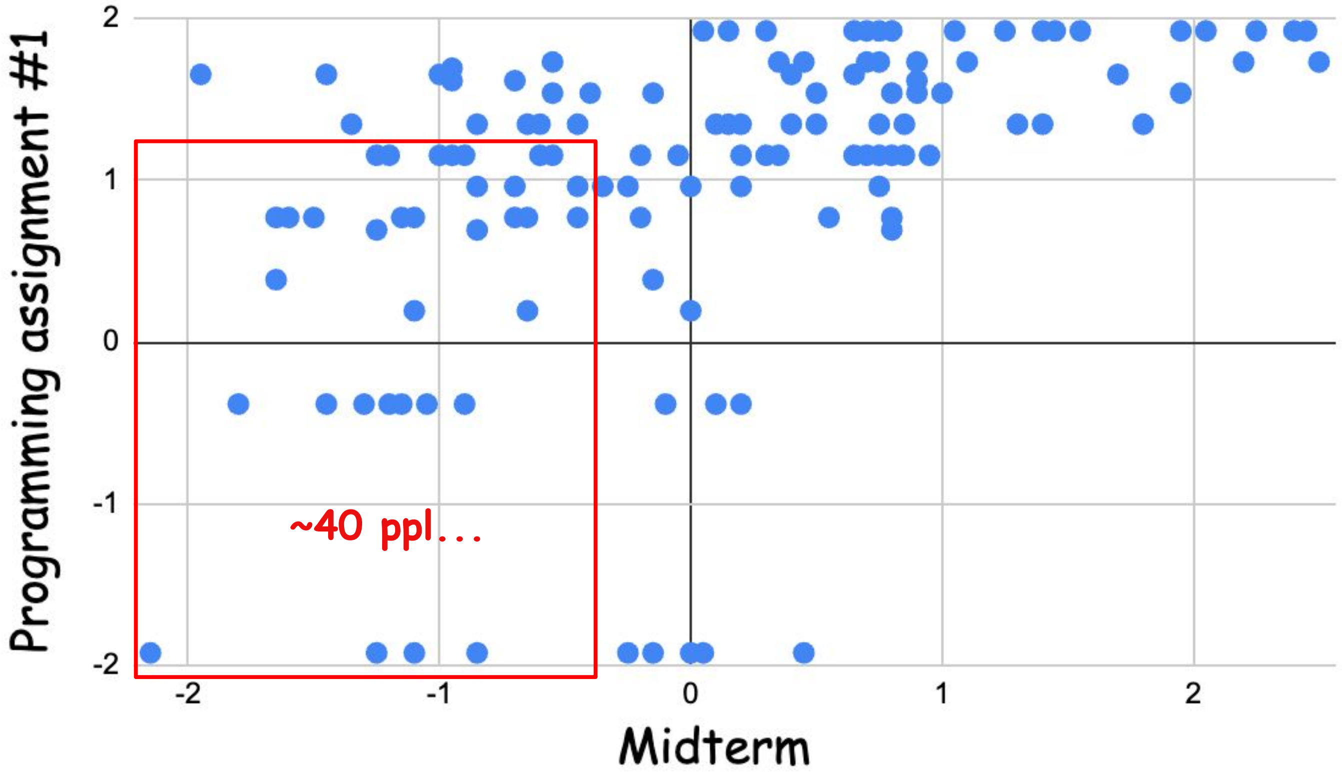


Midterm (median 51 / std.dev.: 21)



4 / 139 ppl for whom checkpatch was **really** their problem

Midterm (median 51 / std.dev.: 21)



~40 ppl...

How to manage stress?

> Personal advices on how to manage your stress

How to manage stress?

> Personal advices on how to manage your stress

- Do not sleep long/well the night before an exam: Being well-rested on a situation that will hyperstimulate you is a bad idea

How to manage stress?

> Personal advices on how to manage your stress

- Do not sleep long/well the night before an exam: Being well-rested on a situation that will hyperstimulate you is a bad idea
- Do not drink coffee for 7-8 hours before an exam: Same reasons, as above. You must minimize all source of hyperstimulation

How to manage stress?

> Personal advices on how to manage your stress

- Do not sleep long/well the night before an exam: Being well-rested on a situation that will hyperstimulate you is a bad idea
- Do not drink coffee for 7-8 hours before an exam: Same reasons, as above. You must minimize all source of hyperstimulation
- Decrease the levels of cortisol in your system: Cortisol is a confidence-killer hormon => Exercise few hours before to decrease it

How to manage stress?

> Personal advices on how to manage your stress

- Do not sleep long/well the night before an exam: Being well-rested on a situation that will hyperstimulate you is a bad idea
- Do not drink coffee for 7-8 hours before an exam: Same reasons, as above. You must minimize all source of hyperstimulation
- Decrease the levels of cortisol in your system: Cortisol is a confidence-killer hormon => Exercise few hours before to decrease it
- Disassociate yourself from the environment **during** the exam: Imaging that you are explaining the exam to someone who would easily get 100% ⇒ This someone is helping you silently in your head

Overall

> Thank you for helping having a smooth exam

- This was a difficult exam: Look around, very few OS courses worldwide cover such breadth of topics

Overall

- **Thank you for helping having a smooth exam**
  - This was a difficult exam: Look around, very few OS courses worldwide cover such breadth of topics
  - The vast majority of people seem to be following along: You need to level up, and practice more, but we are getting there

Overall

> **Thank you for helping having a smooth exam**

- This was a difficult exam: Look around, very few OS courses worldwide cover such breadth of topics
- The vast majority of people seem to be following along: You need to level up, and practice more, but we are getting there
- Final exam will be more difficult: But you will be an even better version of your k22-self until them. Keep going...

Overall

> **Thank you for helping having a smooth exam**

- This was a difficult exam: Look around, very few OS courses worldwide cover such breadth of topics
- The vast majority of people seem to be following along: You need to level up, and practice more, but we are getting there
- Final exam will be more difficult: But you will be an even better version of your k22-self until them. Keep going...
- **What's next?** Put your focus on programming assignment #2

Overall

> **Thank you for helping having a smooth exam**

- This was a difficult exam: Look around, very few OS courses worldwide cover such breadth of topics
- The vast majority of people seem to be following along: You need to level up, and practice more, but we are getting there
- Final exam will be more difficult: But you will be an even better version of your k22-self until them. Keep going...
- **What's next?** Put your focus on programming assignment #2
- Noone will be excluded... but be considerate of how you invest your time

Overall

> **Thank you for helping having a smooth exam**

- Anyone who scored >90-95% has our attention!

Overall

> **Thank you for helping having a smooth exam**

- Anyone who scored >90-95% has our attention!

  >> Your systems understanding is solid

  >> Next question: How to turn you to computer **scientists**

Overall

> **Thank you for helping having a smooth exam**

- Anyone who scored >90-95% has our attention!

  >> Your systems understanding is solid

  >> Next question: How to turn you to computer **scientists**

- The **new** deadline for prog. assignment #2 is Dec. 19th
- There will be an additional 5% bonus — stay tuned!

Overall

> **Thank you for helping having a smooth exam**

- Anyone who scored >90-95% has our attention!

  >> Your systems understanding is solid

  >> Next question: How to turn you to computer **scientists**

- The **new** deadline for prog. assignment #2 is Dec. 19th

- There will be an additional 5% bonus — stay tuned!

- **New** requirement for exams: must get the base on at least one of the exams (midterm or final)

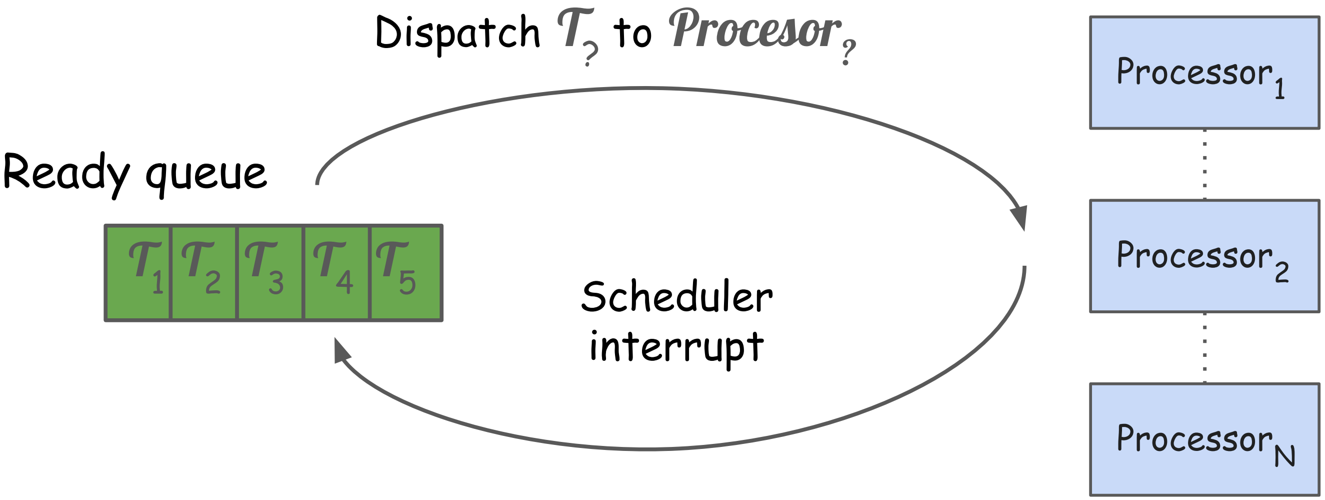
Overview

- We'll start from hardware and follow a question-oriented approach
    - ~~Intro [Q: What is an OS?]~~
    - ~~Events [Q: When does the OS run?]~~
    - ~~Runtime [Q: How does a program look like in memory?]~~
    - ~~Processes [Q: What is a process?]~~
    - ~~IPC [Q: How do processes communicate?]~~
    - ~~Threads [Q: What is a thread?]~~
    - ~~Synchronization [Q: What goes wrong w/o synchronization?]~~
    - **Time Management [Q: What is scheduling?]**
    - Memory Management [Q: What is virtual memory?]
    - Files [Q: What is a file descriptor?]
    - Storage Management [Q: How do we allocate disk space to files?]

* Basic (H/W & S/W)
* Abstractions
* Primitives
* **Mechanisms**

The scheduling problem

> Given k tasks ready to run in a system with N available processors, which task should be dispatched to which processor at any given point in time?



The scheduling problem

> Given k tasks ready to run in a system with N available processors, which task should be dispatched to which processor at any given point in time?

> Quantitative goals

- Minimize avg. completion time of all jobs
- Minimize the avg. response time of all jobs (latency)
- Maximize #jobs completed per unit of time (throughput)

The scheduling problem

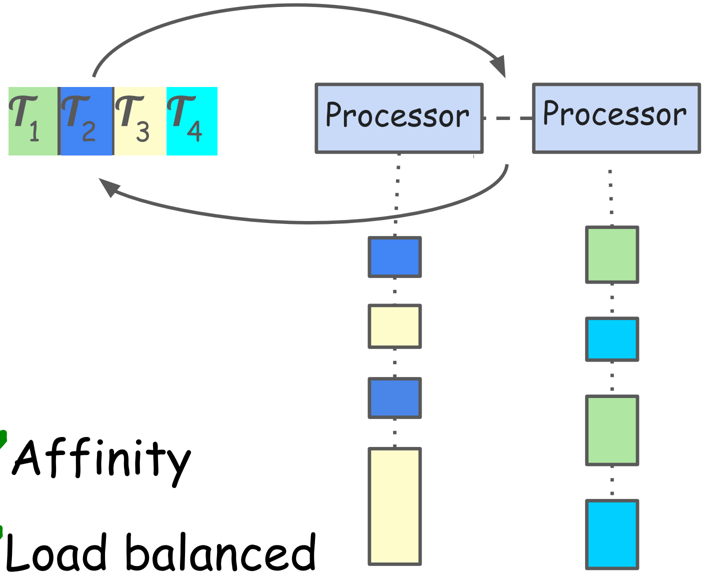
> Given k tasks ready to run in a system with N available processors, which task should be dispatched to which processor at any given point in time?

> Qualitative goals

- Jobs receive a similar share of available processors' time
- Upper bound on the maximum latency of jobs
- Uniform load across all available processors

SMP load balancing and processor affinity

Processor affinity?



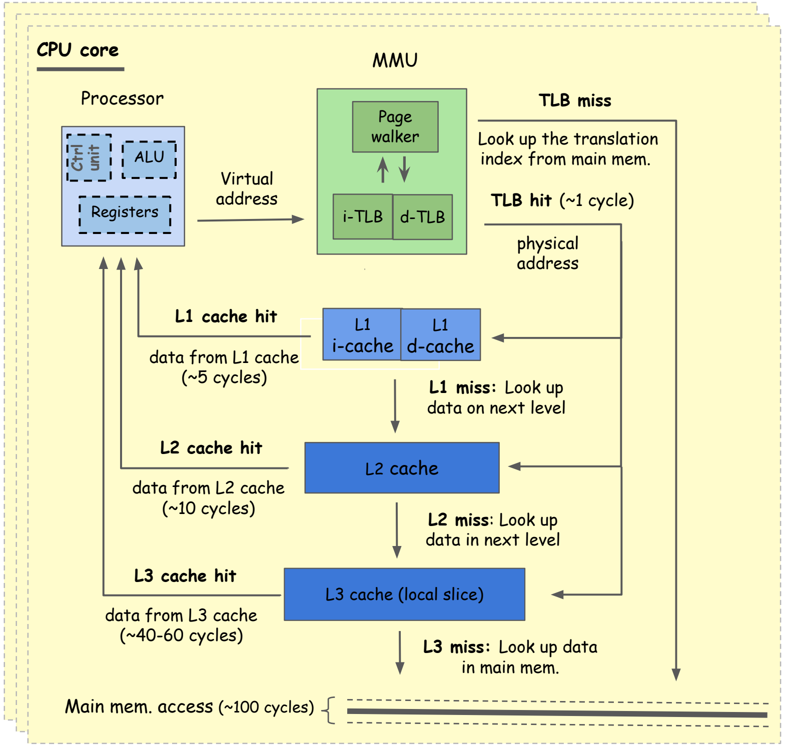
T_1 T_2 T_3 T_4

Processor

Processor

✔ Affinity

✔ Load balanced



CPU core

MMU

Processor

Ctrl unit

ALU

Registers

Page walker

i-TLB

d-TLB

Virtual address

TLB miss

Look up the translation index from main mem.

TLB hit (~1 cycle)

physical address

L1 cache hit

L1 i-cache

L1 d-cache

data from L1 cache (~5 cycles)

L1 miss: Look up data on next level

L2 cache hit

L2 cache

data from L2 cache (~10 cycles)

L2 miss: Look up data in next level

L3 cache hit

L3 cache (local slice)

data from L3 cache (~40-60 cycles)

L3 miss: Look up data in main mem.

Main mem. access (~100 cycles)

Workloads and scheduling requirements

Real time workloads: Hard real time and soft real time

- Hard real time
  - Their tasks must finish within specific deadlines
  - Example: Pacemakers, Airbag deployment systems, Autopilots
  - Sched. goals: **Zero miss rate; Guarantees every time**
  - Sched. algorithms: Earliest Deadline First (EDF)
- Soft real time
  - Their tasks must receive priority over lower-priority tasks
  - Example: Video Streaming / Multimedia applications
  - Sched. goals: **Bounded latency**
  - Sched. algorithms: Priority-based scheduling

Workloads and scheduling requirements

CPU- vs I/O-bound workloads

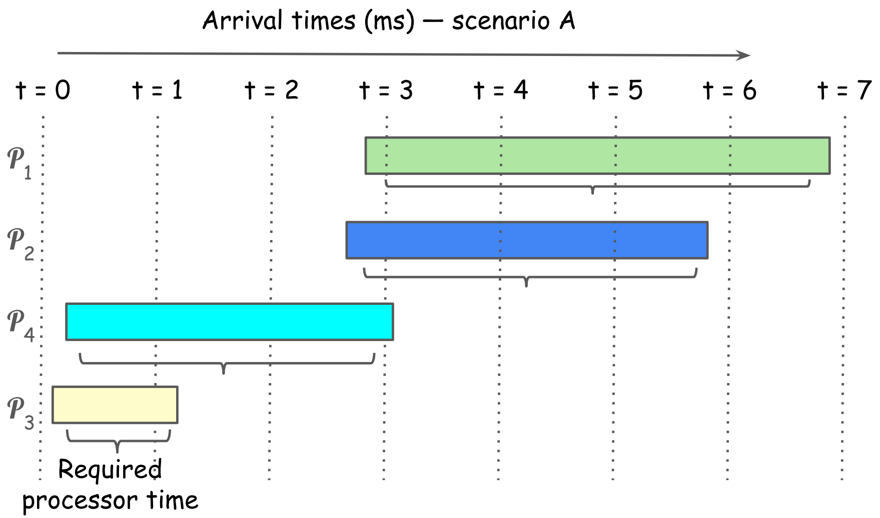
- CPU-bound
  - Their tasks spend most time doing intensive computation
  - Rarely yield voluntarily and rarely need to perform I/O
  - Example: Scientific simulations / computations
  - Sched. goals: **Balanced processor time / avoid starvation**
  - Sched. algorithms: RR with large quanta

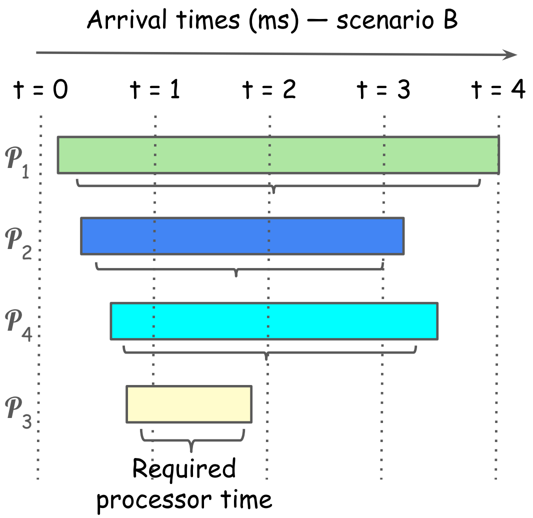
Workloads and scheduling requirements

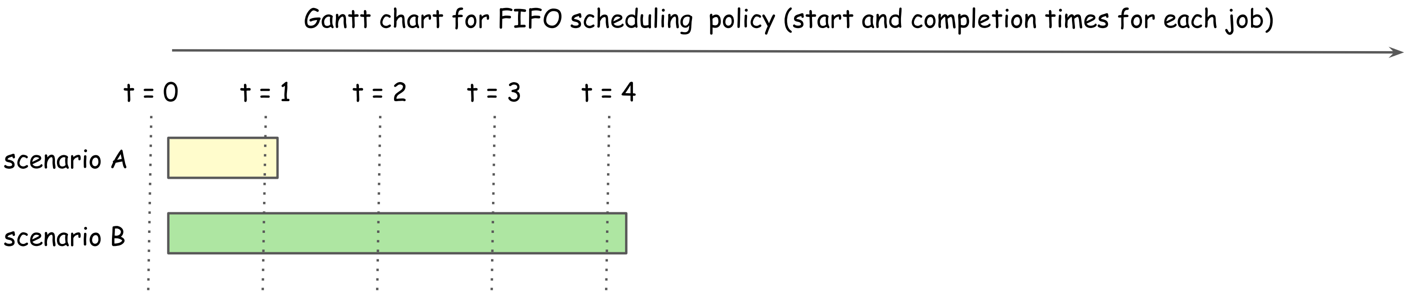
CPU- vs I/O-bound workloads

- I/O-bound workloads
  - Their tasks spend most of their time waiting for I/O
  - Short processor bursts and then block again
  - Example: Downloading a file / fetching data from disk
  - Sched. goals: Minimize I/O device idle periods by promptly allocating the processor for the brief time needed to initiate I/O requests (usually via DMA)
  - Sched. algorithms: Priority-based favoring I/O-bound tasks

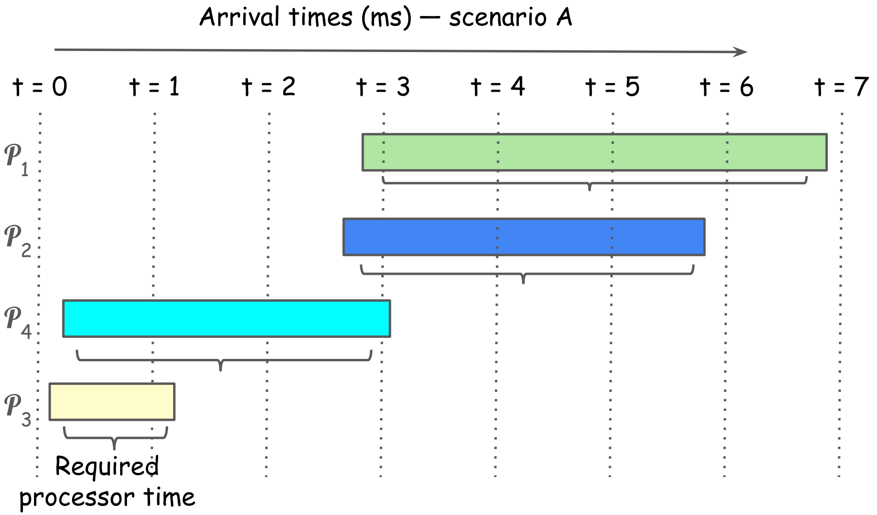
First in First Out scheduling policy (SCHED_FIFO)

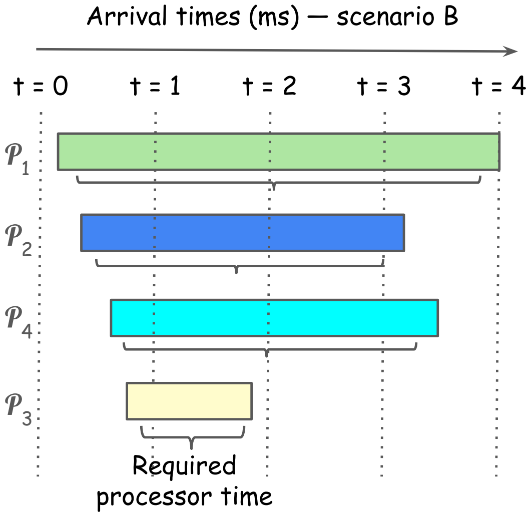


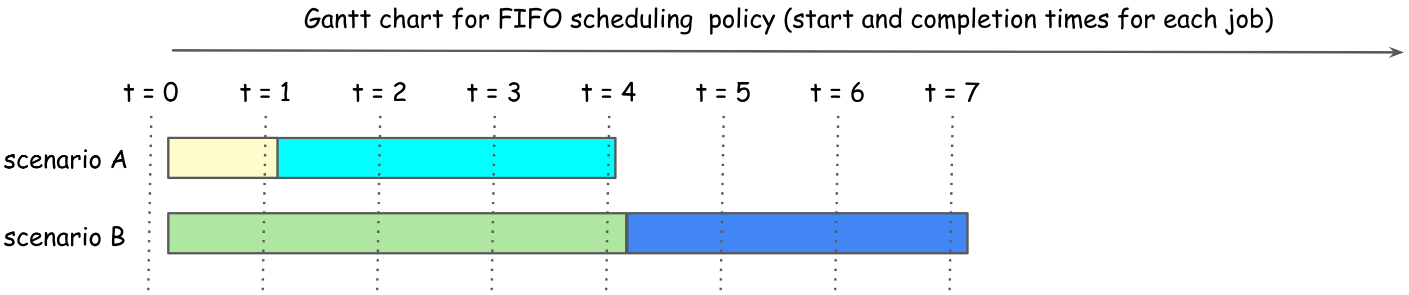




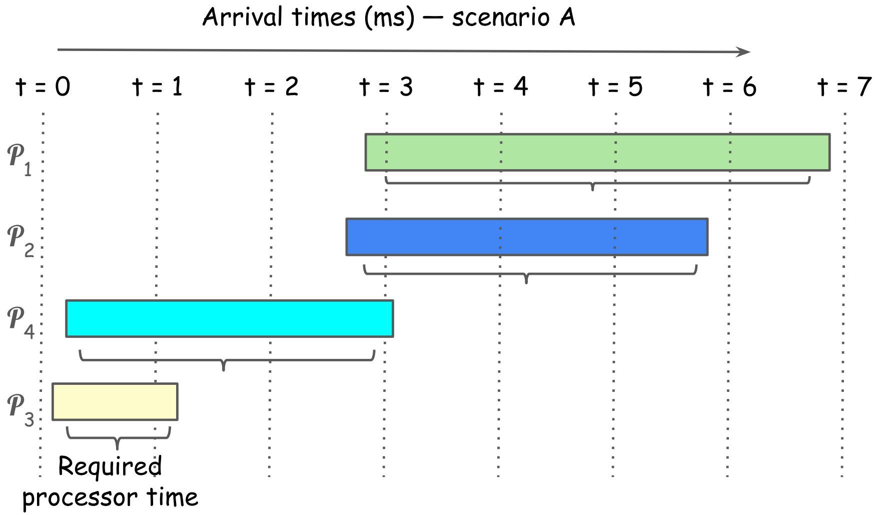
First in First Out scheduling policy (SCHED_FIFO)

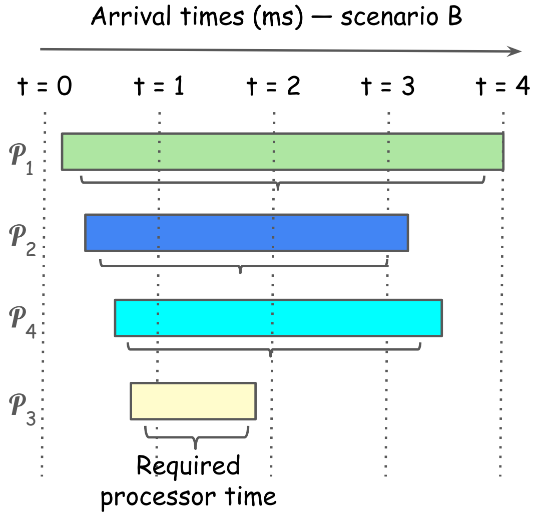


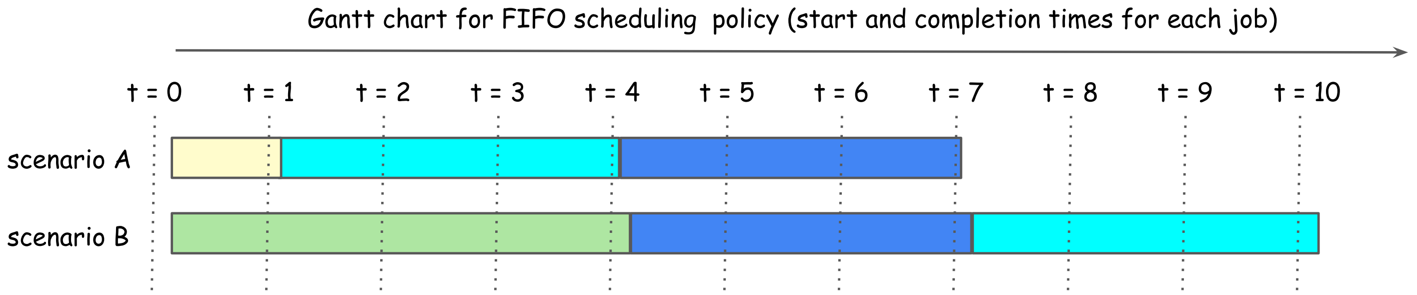




First in First Out scheduling policy (SCHED_FIFO)

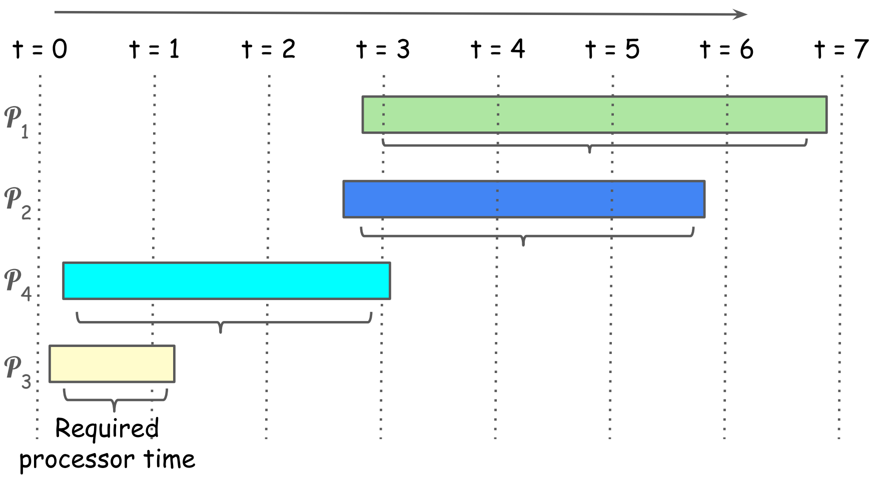




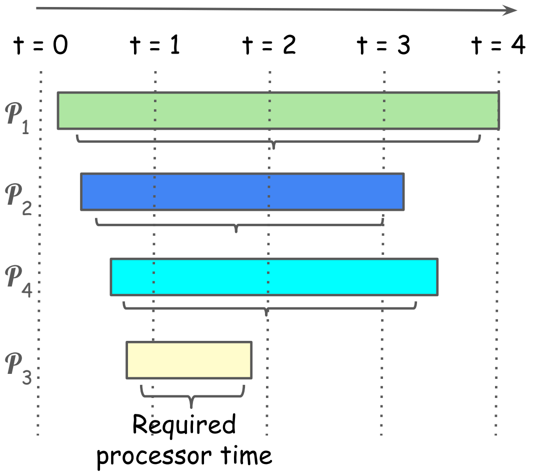


First in First Out scheduling policy (SCHED_FIFO)

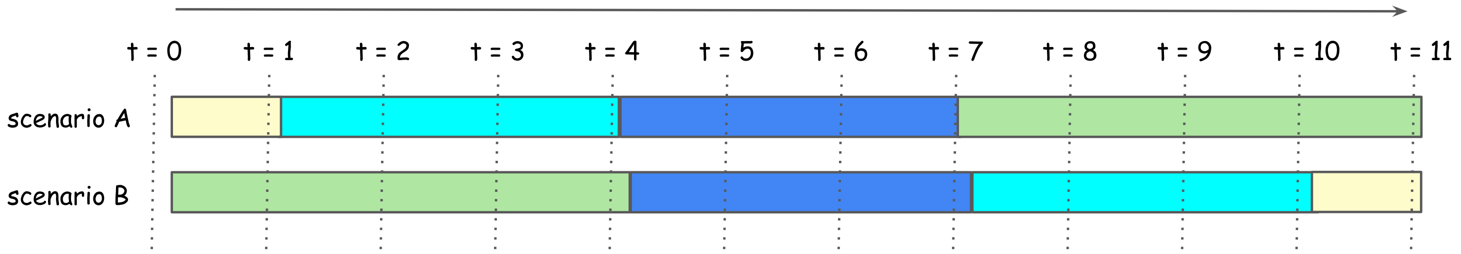
Arrival times (ms) — scenario A



Arrival times (ms) — scenario B

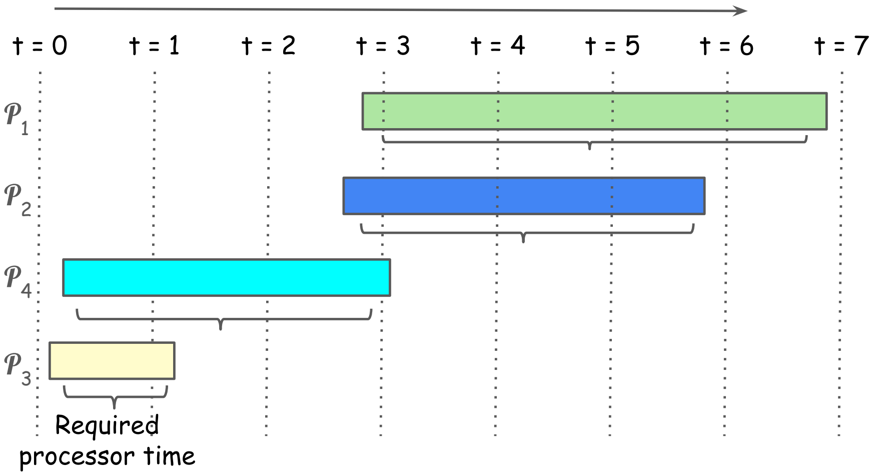


Gantt chart for FIFO scheduling policy (start and completion times for each job)

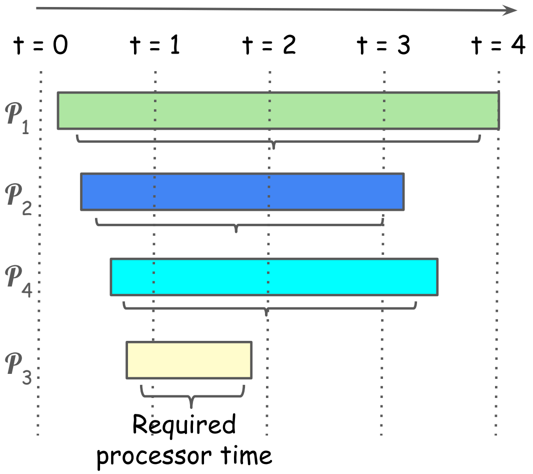


SCHED_FIFO: Avg.completion and response time?

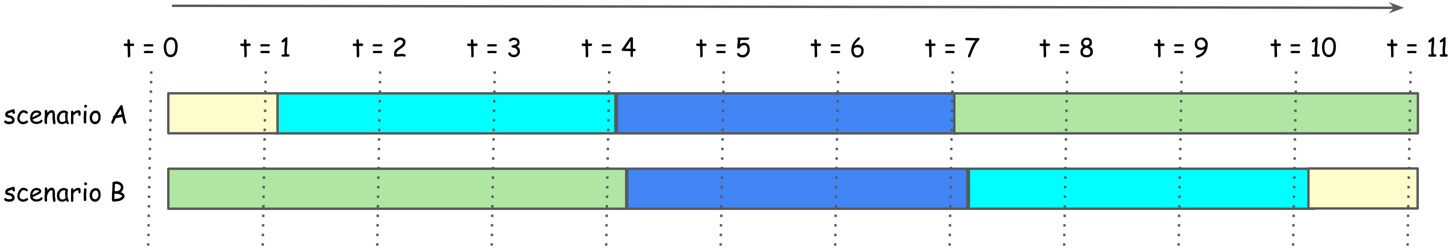
Arrival times (ms) — scenario A



Arrival times (ms) — scenario B

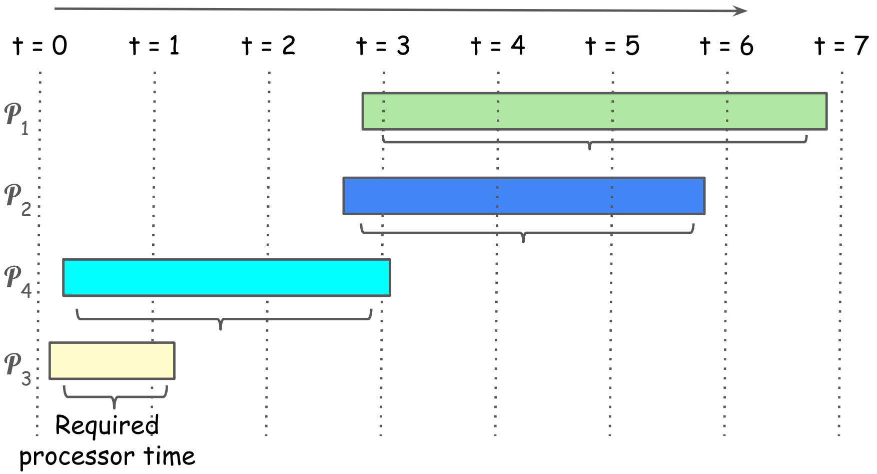


Gantt chart for FIFO scheduling policy (start and completion times for each job)

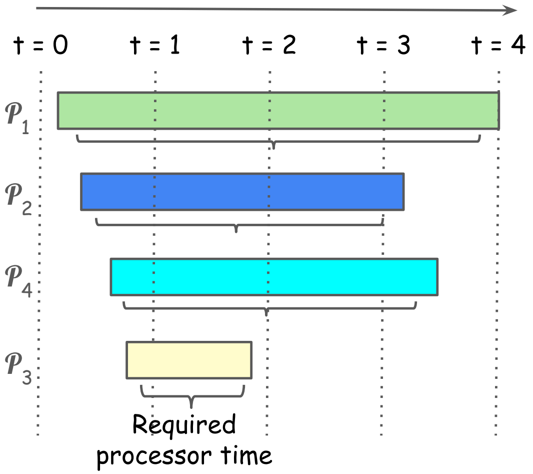


Round Robin scheduling policy (SCHED_RR)

Arrival times (ms) — scenario A



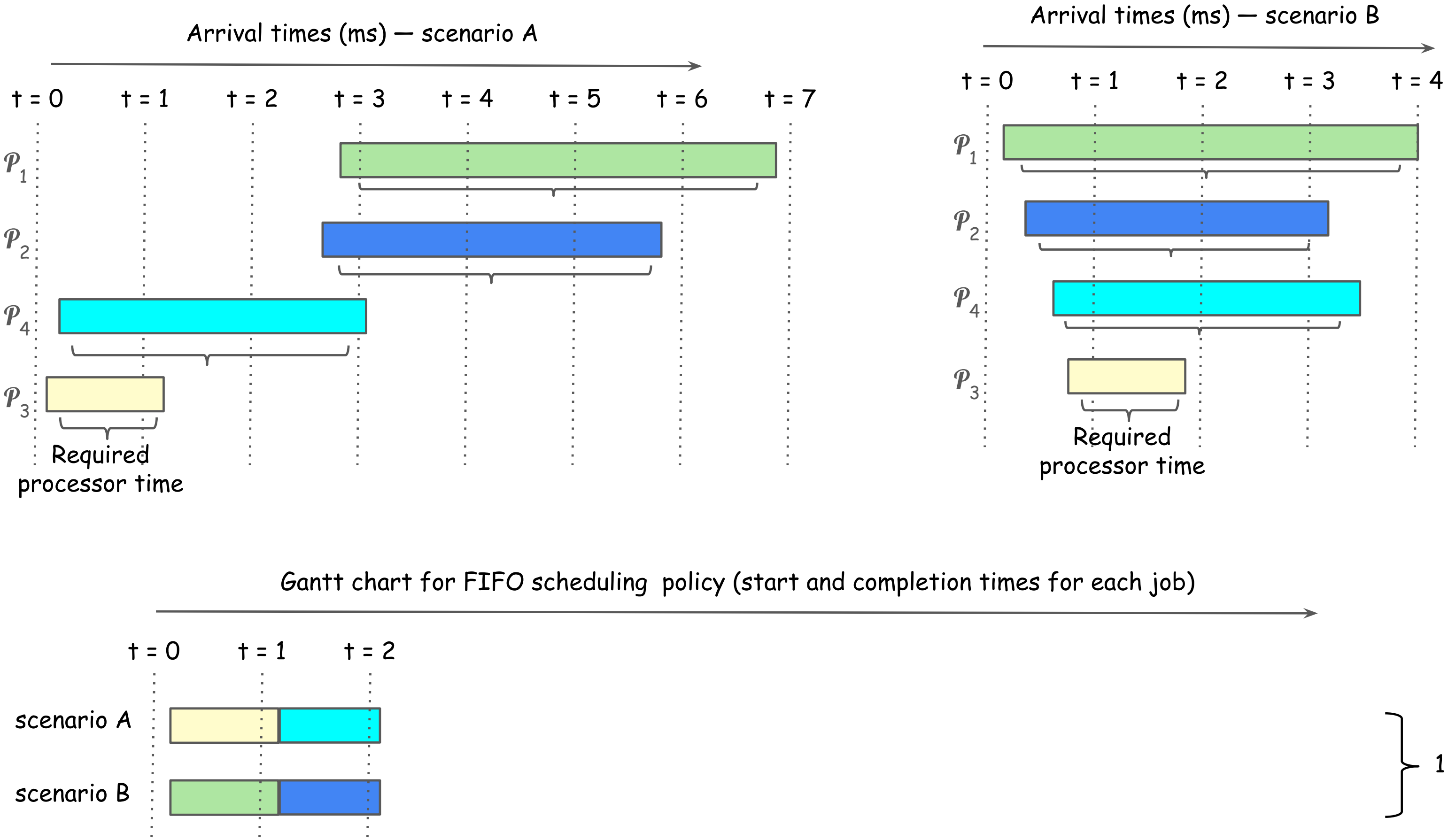
Arrival times (ms) — scenario B



Gantt chart for FIFO scheduling policy (start and completion times for each job)

t = 0
t = 1
scenario A
scenario B
1 ms timeslice

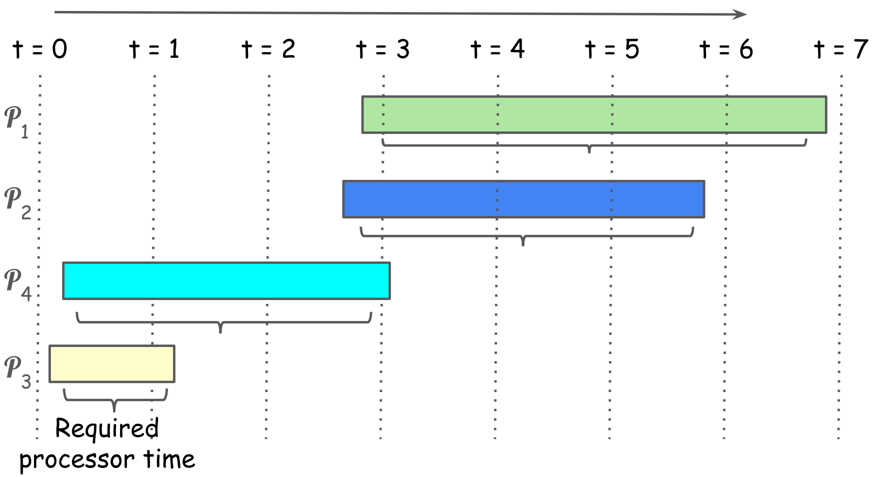
Round Robin scheduling policy (SCHED_RR)



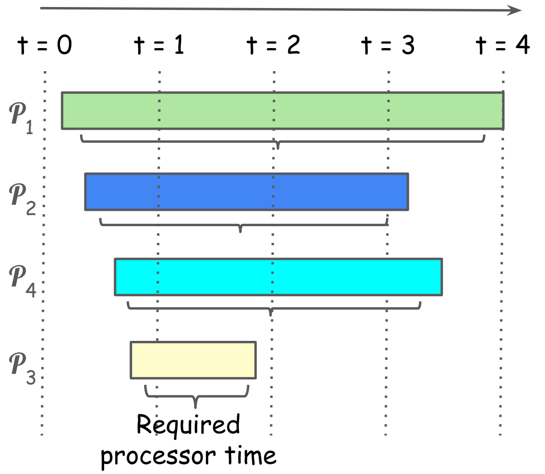
1 ms timeslice

Round Robin scheduling policy (SCHED_RR)

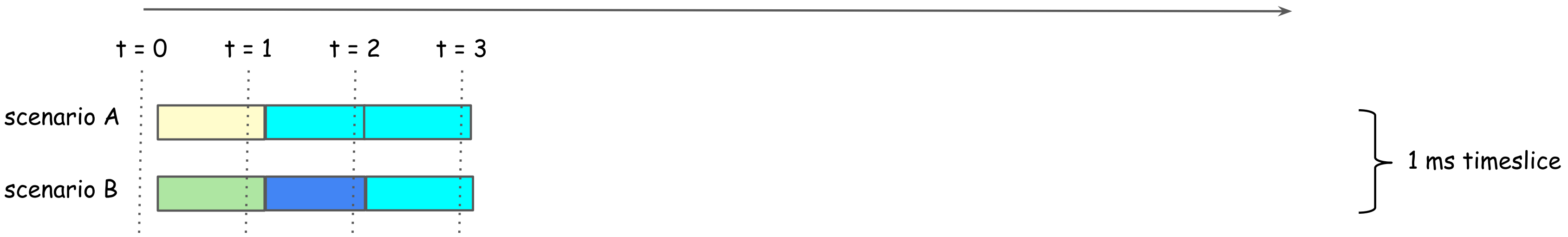
Arrival times (ms) — scenario A



Arrival times (ms) — scenario B

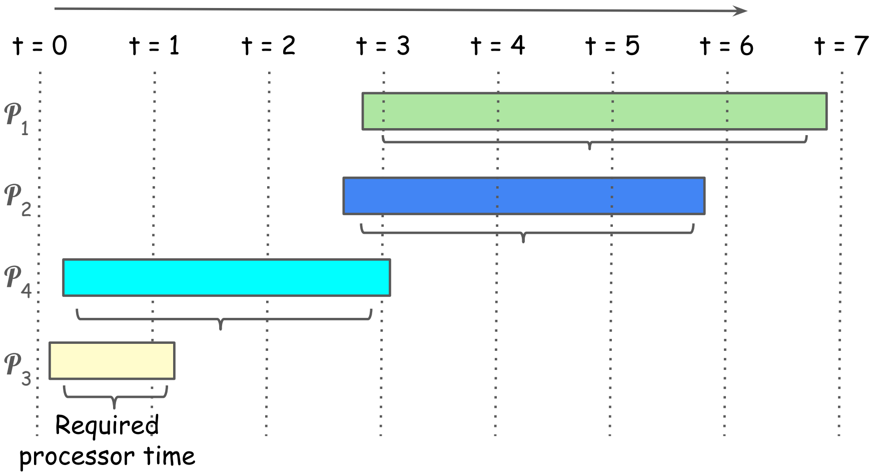


Gantt chart for FIFO scheduling policy (start and completion times for each job)

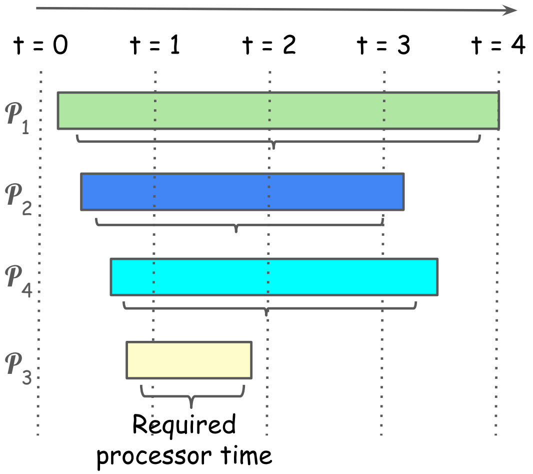


Round Robin scheduling policy (SCHED_RR)

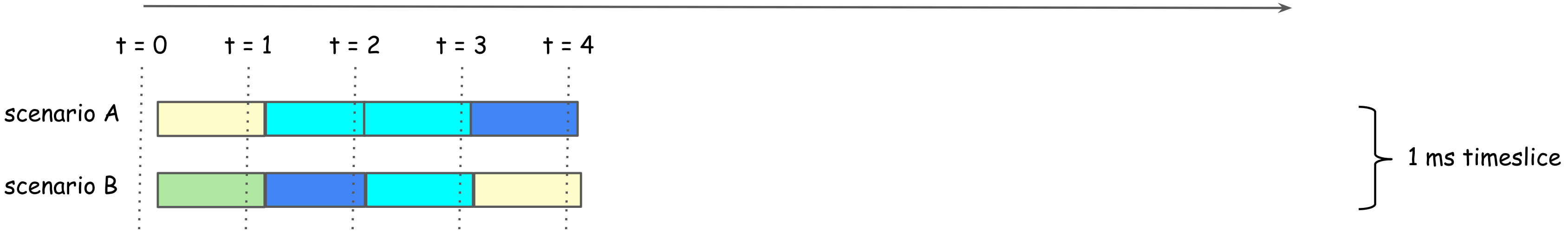
Arrival times (ms) — scenario A



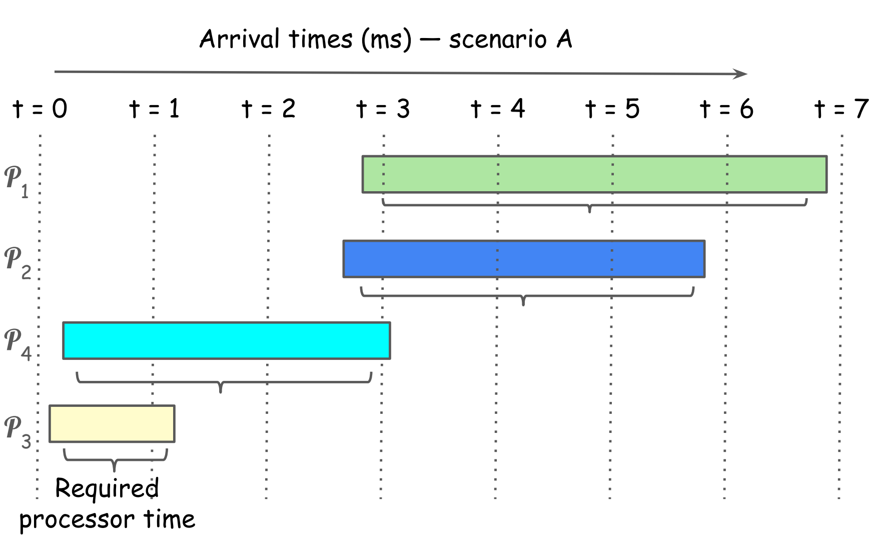
Arrival times (ms) — scenario B

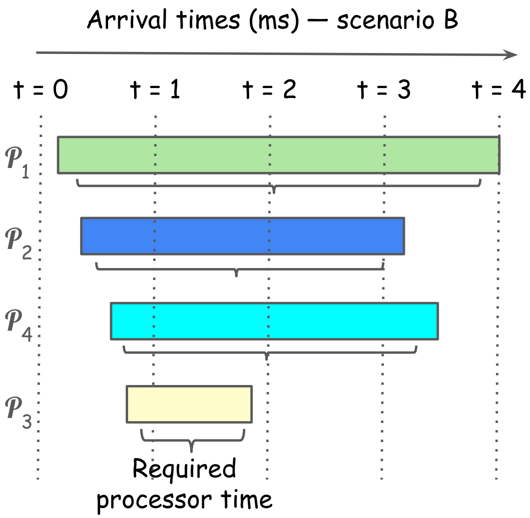


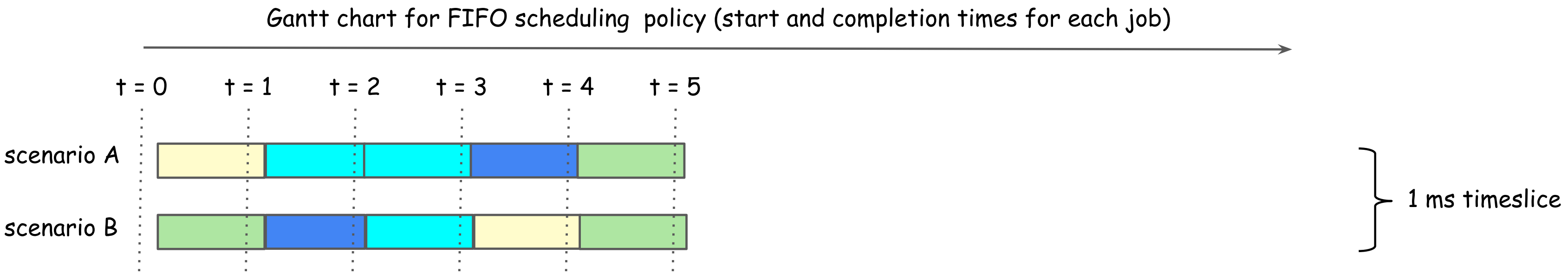
Gantt chart for FIFO scheduling policy (start and completion times for each job)



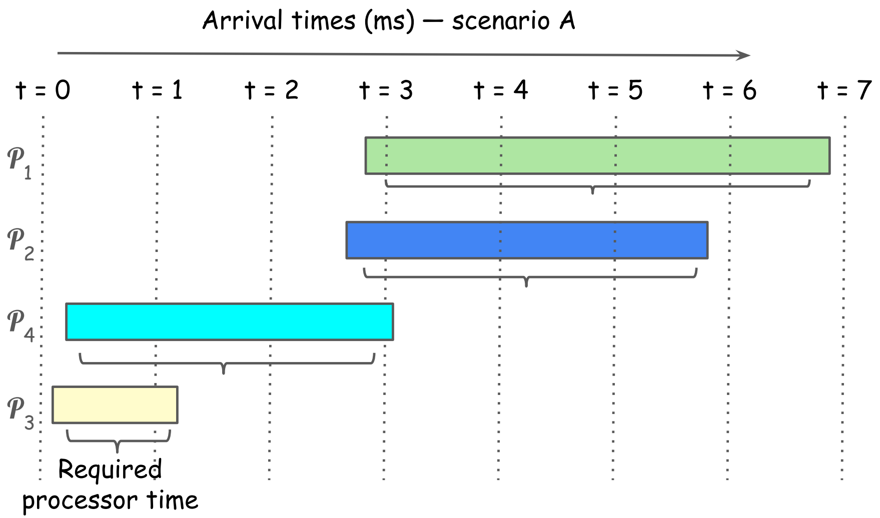
Round Robin scheduling policy (SCHED_RR)



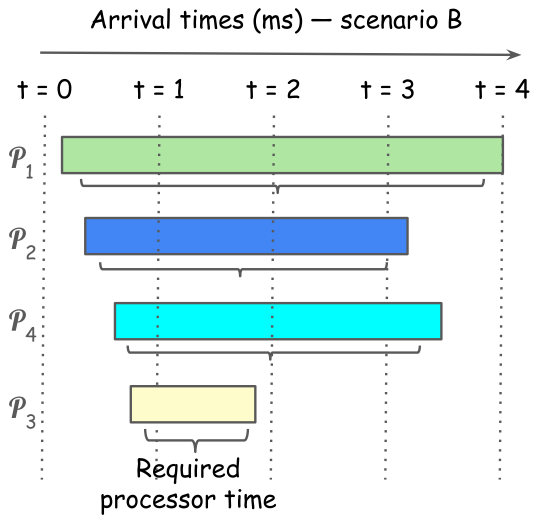



Gantt chart for FIFO scheduling policy (start and completion times for each job)
t = 0
t = 1
t = 2
t = 3
t = 4
t = 5
scenario A
scenario B
1 ms timeslice

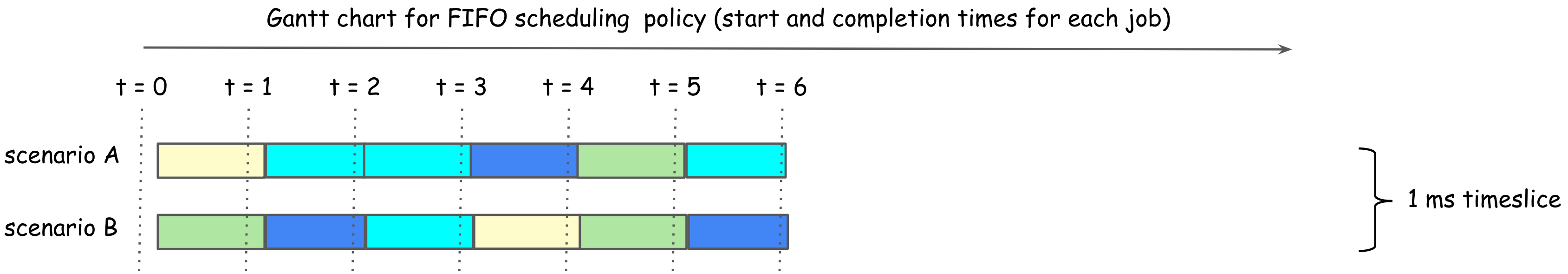
Round Robin scheduling policy (SCHED_RR)



Arrival times (ms) — scenario A



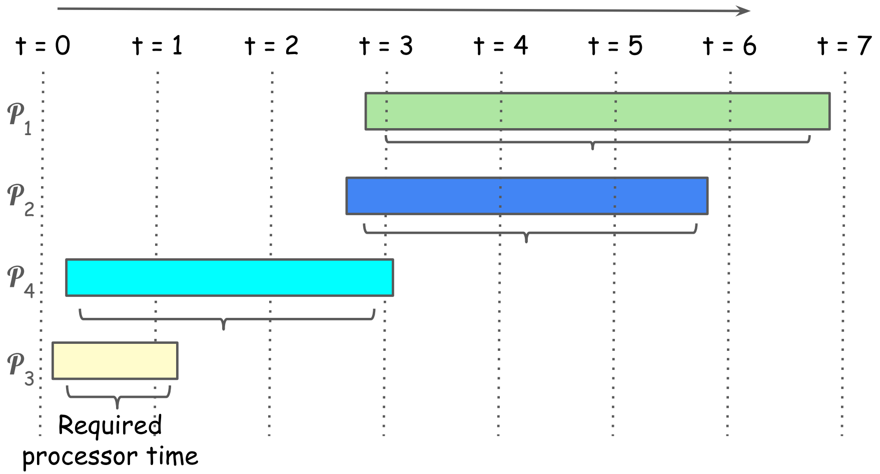
Arrival times (ms) — scenario B



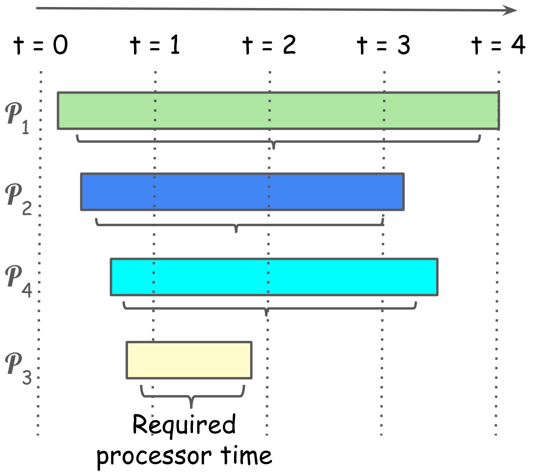
Gantt chart for FIFO scheduling policy (start and completion times for each job)

Round Robin scheduling policy (SCHED_RR)

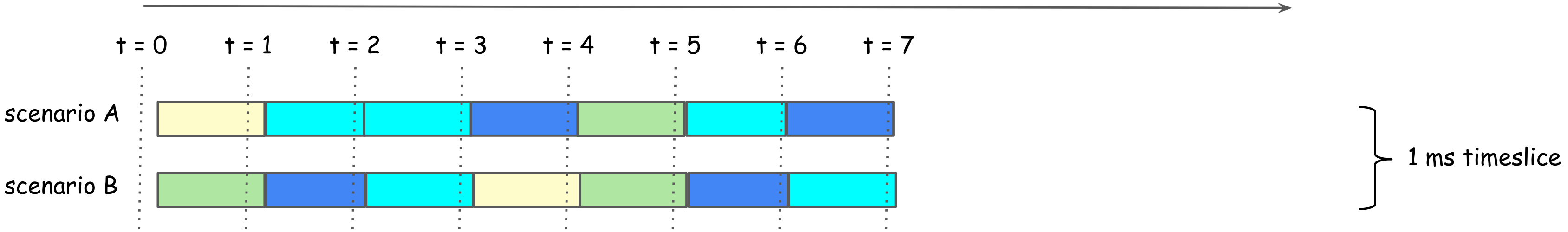
Arrival times (ms) — scenario A



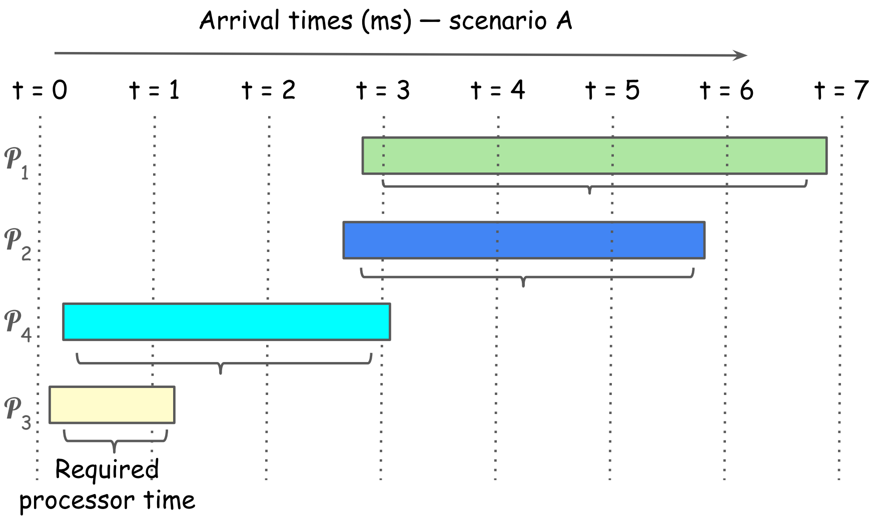
Arrival times (ms) — scenario B

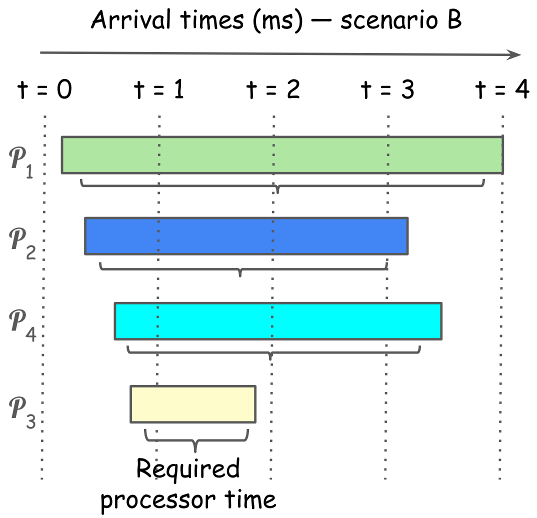


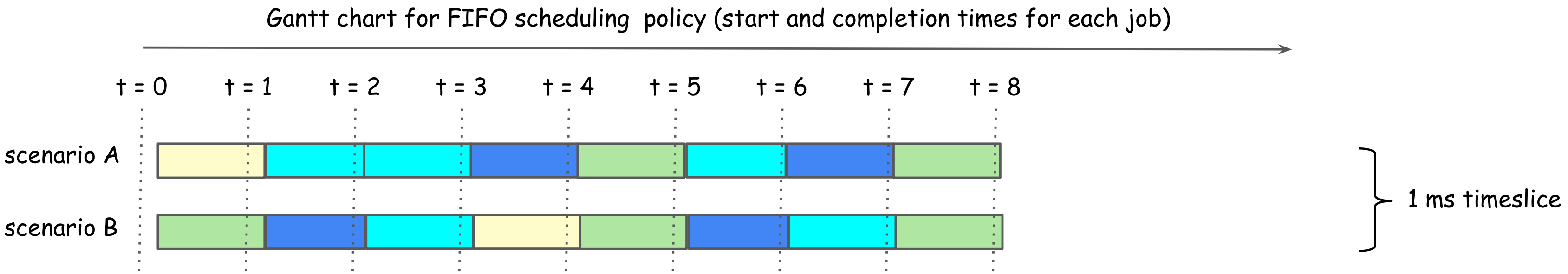
Gantt chart for FIFO scheduling policy (start and completion times for each job)



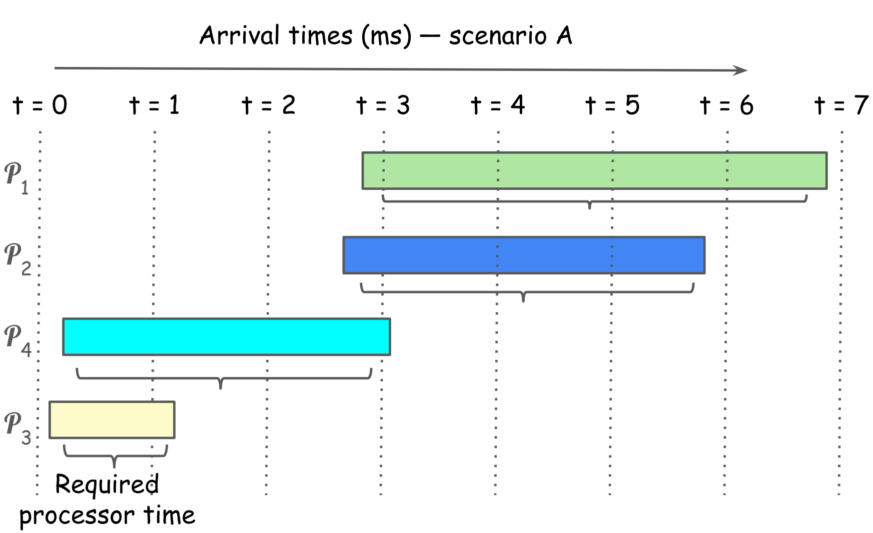
Round Robin scheduling policy (SCHED_RR)

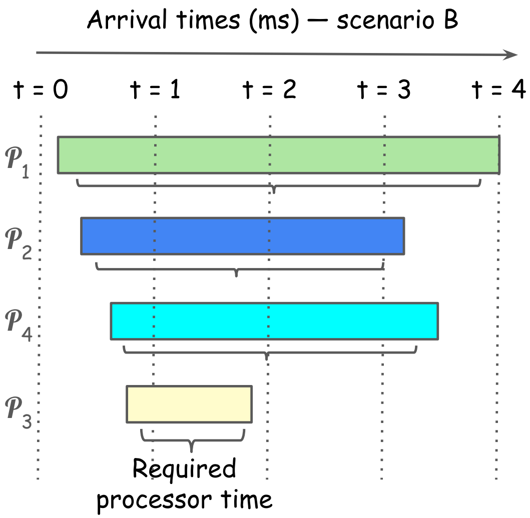


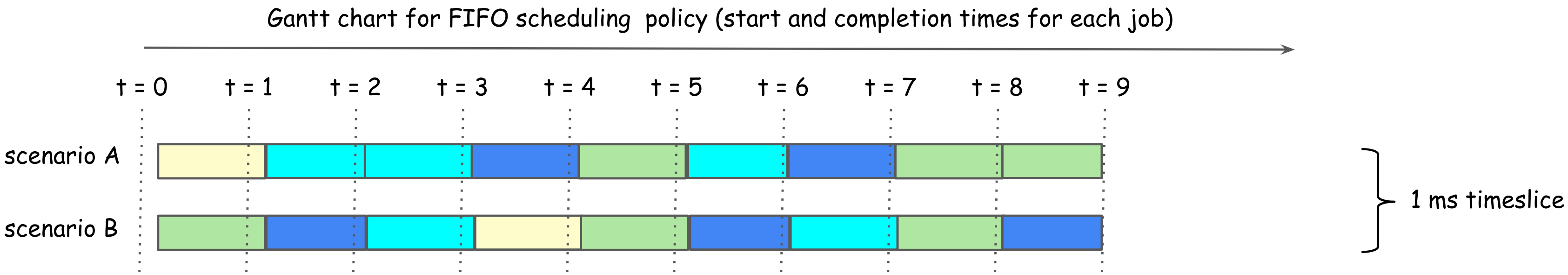




Round Robin scheduling policy (SCHED_RR)

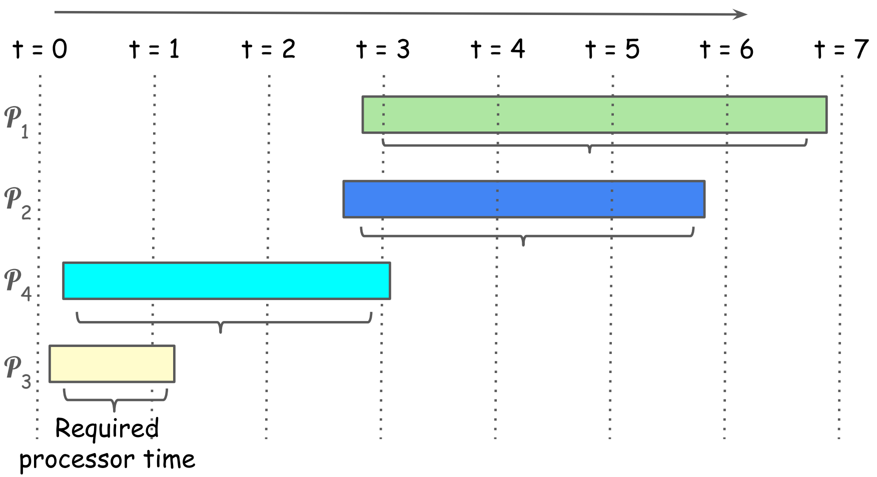




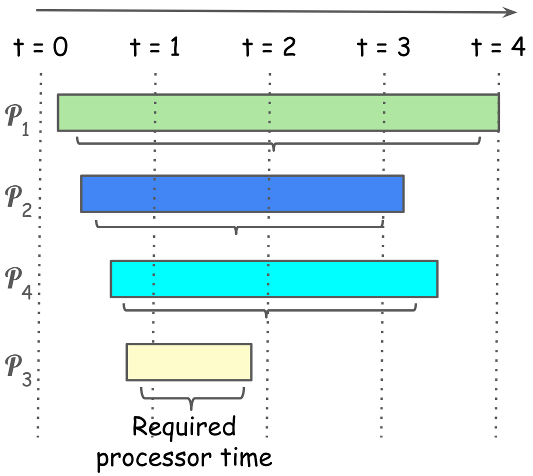


Round Robin scheduling policy (SCHED_RR)

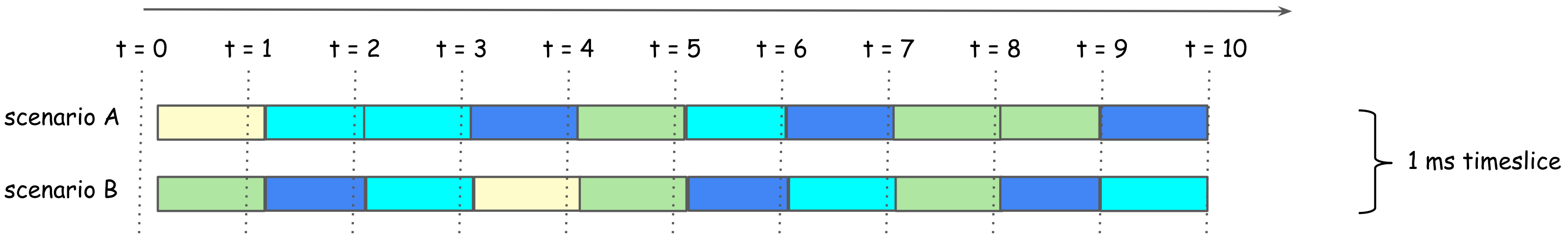
Arrival times (ms) — scenario A



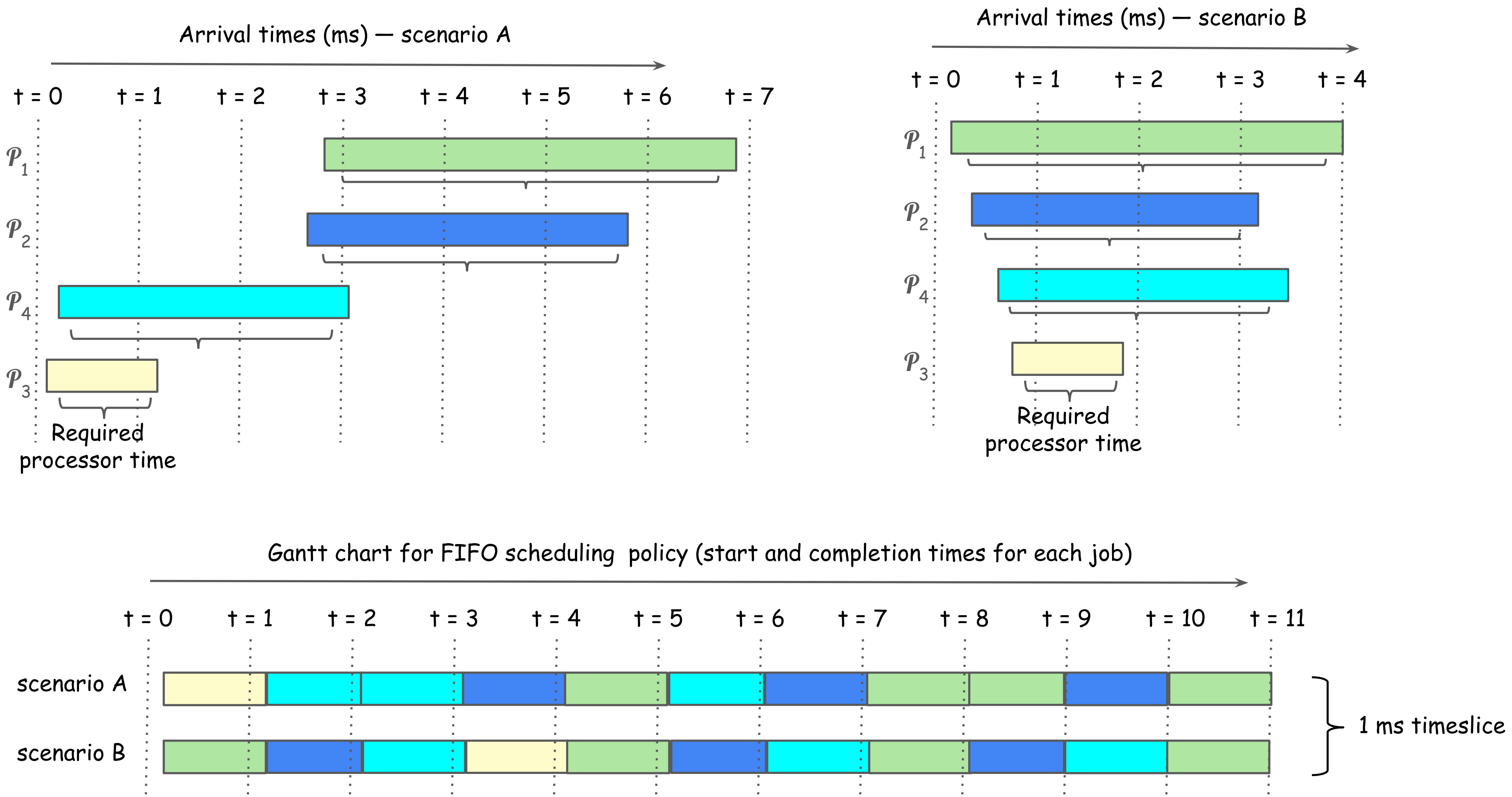
Arrival times (ms) — scenario B



Gantt chart for FIFO scheduling policy (start and completion times for each job)

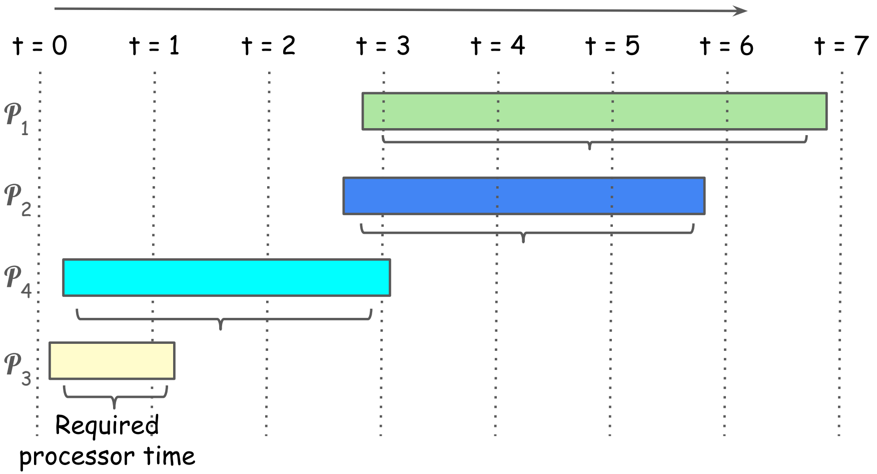


Round Robin scheduling policy (SCHED_RR)

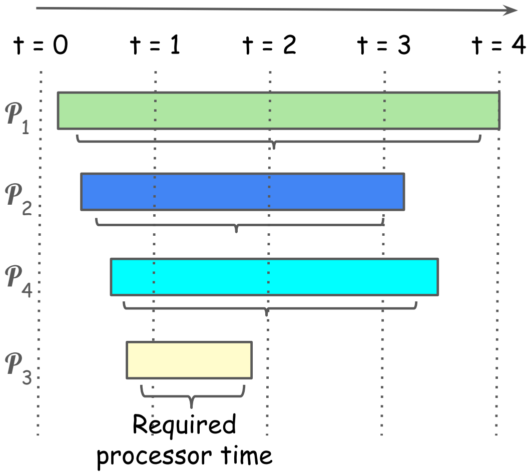


SCHED_RR: Avg.completion and response time?

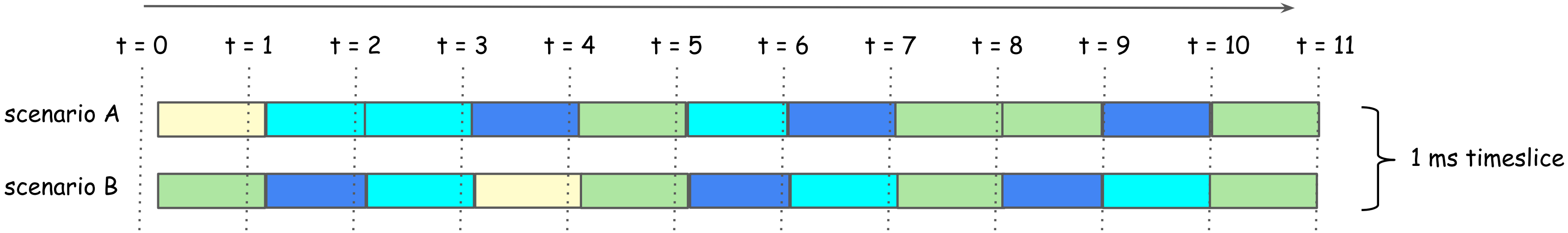
Arrival times (ms) — scenario A



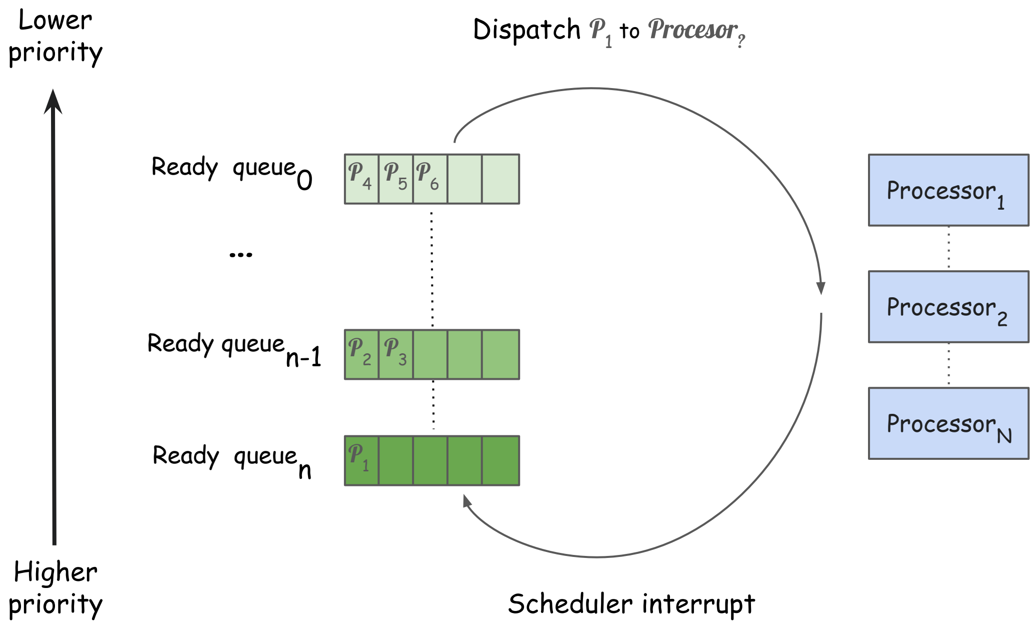
Arrival times (ms) — scenario B



Gantt chart for FIFO scheduling policy (start and completion times for each job)



Hierarchical priority-based scheduling



Ready for your first bug in the outer universe

Ready for your first bug in the outer universe

Mars Rover: software and hardware

- Works Real-Time Operating System (RTOS)

Mars Rover: software and hardware

- Works Real-Time Operating System (RTOS)
  - Tasks must meet strict timing constraints

Mars Rover: software and hardware

> Works Real-Time Operating System (RTOS)

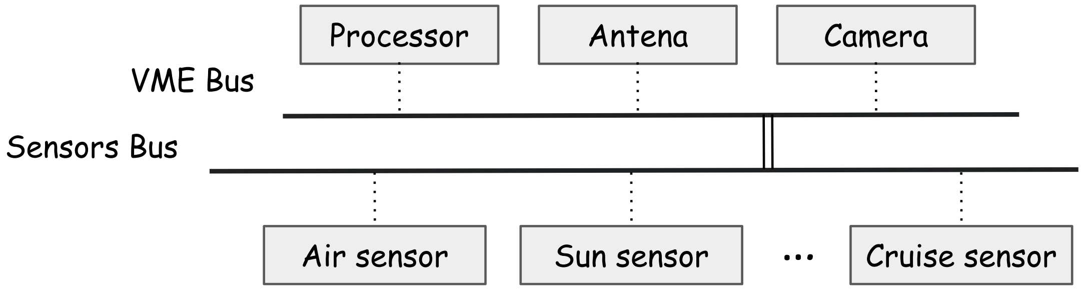
- Tasks must meet strict timing constraints
- Preemptive with priority-based scheduling

Mars Rover: software and hardware

- Works Real-Time Operating System (RTOS)
  - Tasks must meet strict timing constraints
  - Preemptive with priority-based scheduling
  - Scheduler ticks at 8 Hz (i.e., every 125ms)

Mars Rover: software and hardware

- Works Real-Time Operating System (RTOS)
  - Tasks must meet strict timing constraints
  - Preemptive with priority-based scheduling
  - Scheduler ticks at 8 Hz (i.e., every 125ms)
- Hardware overview



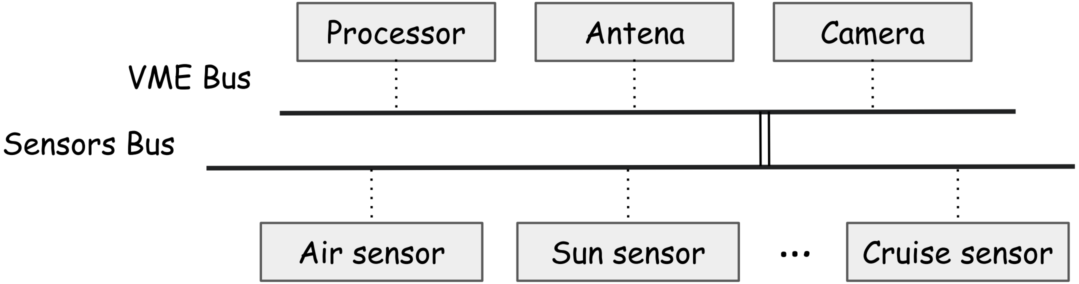
Mars Rover: software and hardware

> **Works Real-Time Operating System (RTOS)**
- Tasks must meet strict timing constraints
- Preemptive with priority-based scheduling
- Scheduler ticks at 8 Hz (i.e., every 125ms)

> **Hardware overview**
- Data from sensor bus to the VMA bus (to antenna)
- Processor signal from VMA bus to sensor bus (cruise)

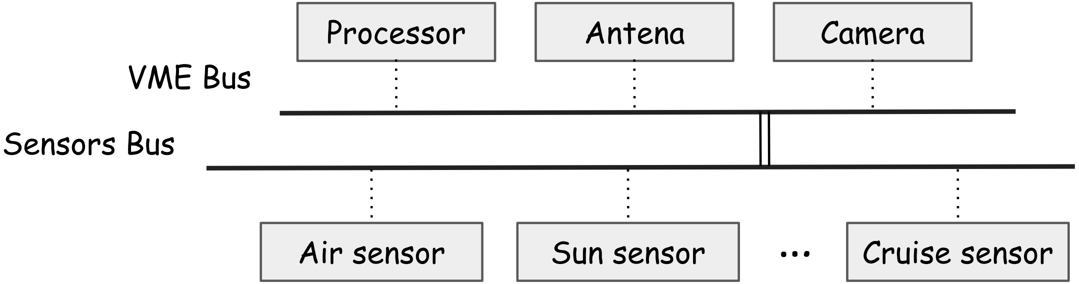
Mars Rover: software and hardware

- Synchronization



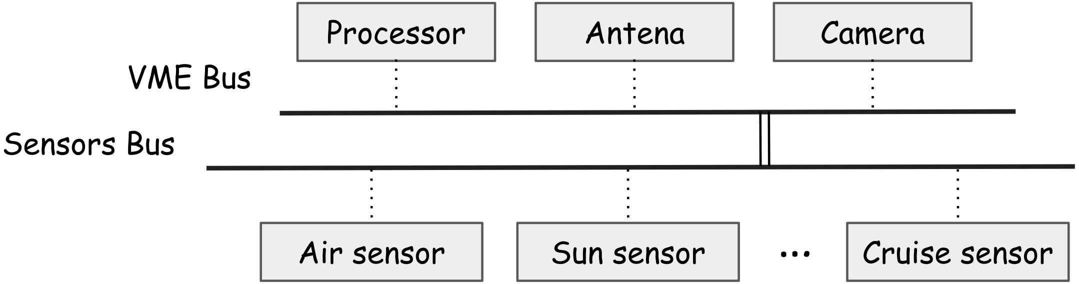
Mars Rover: software and hardware

- **Synchronization**
  - sched_tsk: Decides who transmits data next



Mars Rover: software and hardware

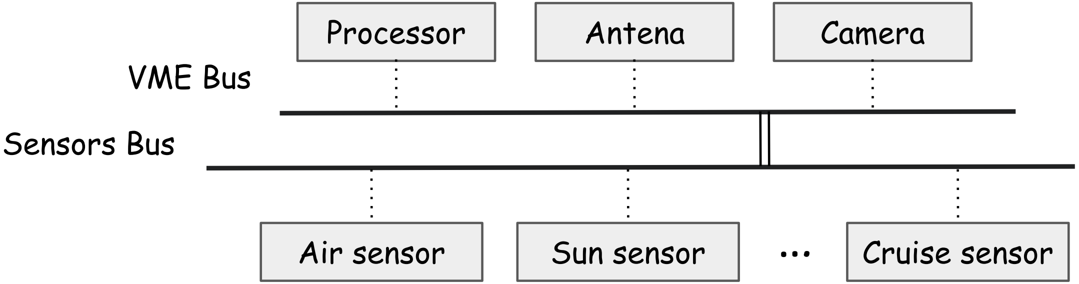
- Synchronization
  - sched_tsk: Decides who transmits data next
  - dist_tsk: Decides who receives data next



Mars Rover: software and hardware

> **Synchronization**

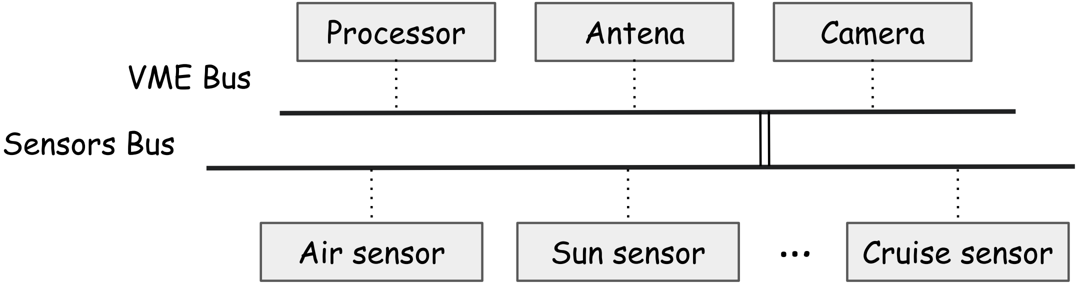
- sched_tsk: Decides who transmits data next
- dist_tsk: Decides who receives data next
- comm_tsk: Uses the antenna to transmit data to Earth



Mars Rover: software and hardware

> **Synchronization**

- sched_tsk: Decides who transmits data next
- dist_tsk: Decides who receives data next
- comm_tsk: Uses the antenna to transmit data to Earth
- asi_tsk: Uses the air sensor for scientific computations

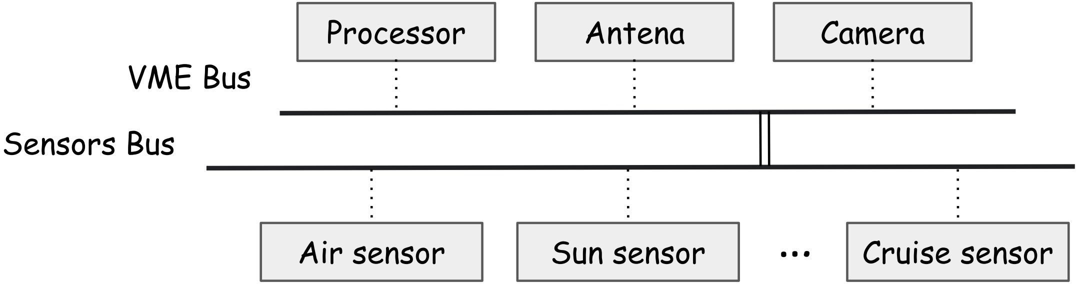


Mars Rover: software and hardware

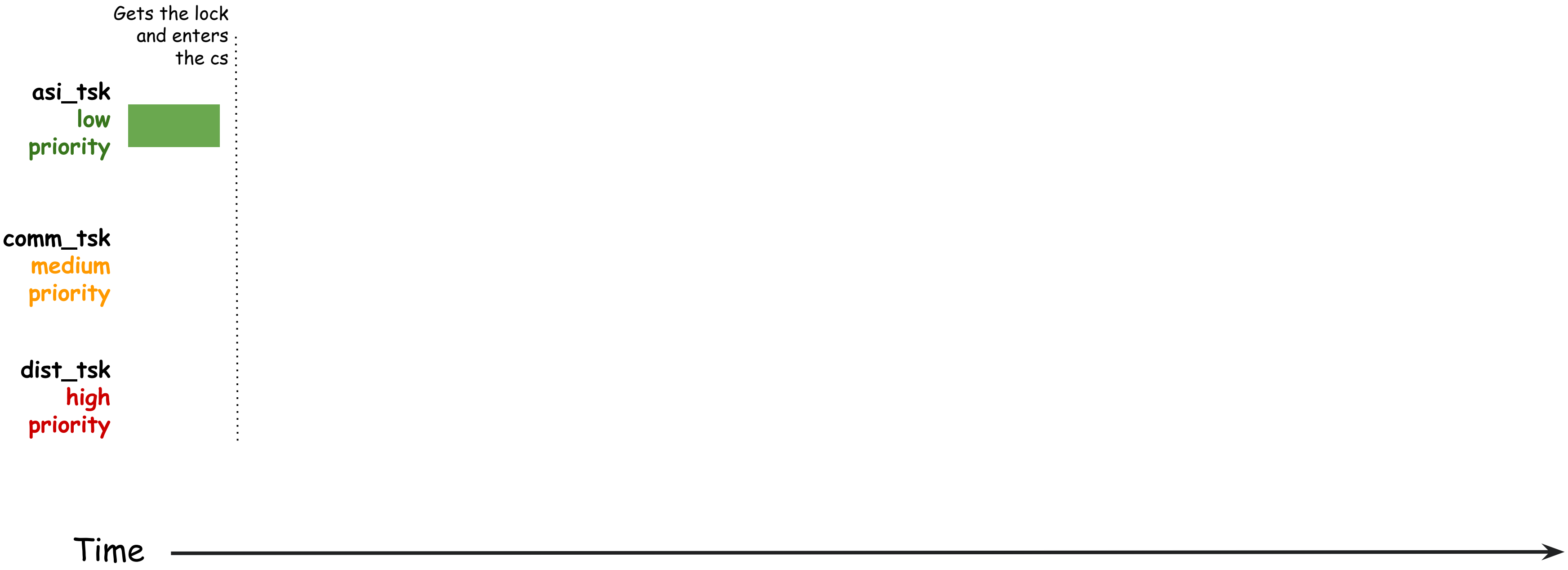
> **Synchronization**

- sched_tsk: Decides who transmits data next
- dist_tsk: Decides who receives data next
- comm_tsk: Uses the antenna to transmit data to Earth
- asi_tsk: Uses the air sensor for scientific computations

Priorities: sched_tsk > dist_tsk (high) > comm_tsk (medium) > asi_tsk (low)



Ready for your first bug in the outer universe?



Gets the lock and enters the cs

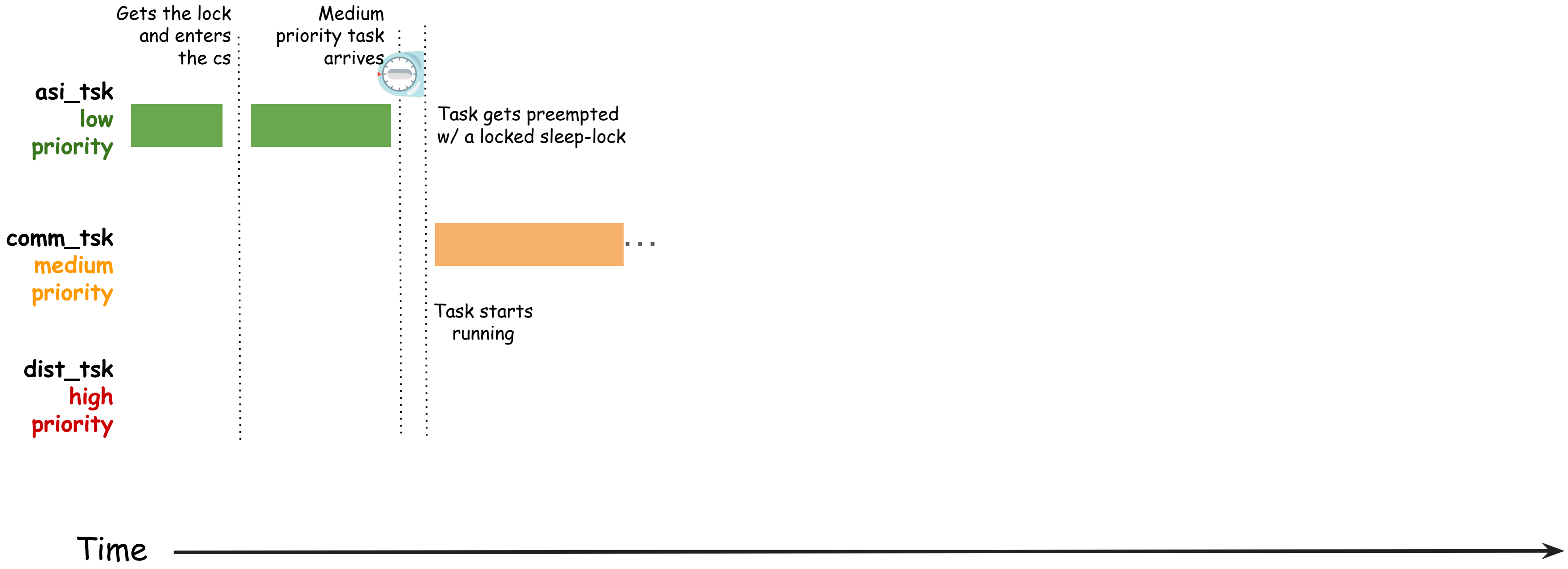
asi_tsk low priority

comm_tsk medium priority

dist_tsk high priority

Time

Ready for your first bug in the outer universe?



Gets the lock and enters the cs

Medium priority task arrives

Task gets preempted w/ a locked sleep-lock

Task starts running

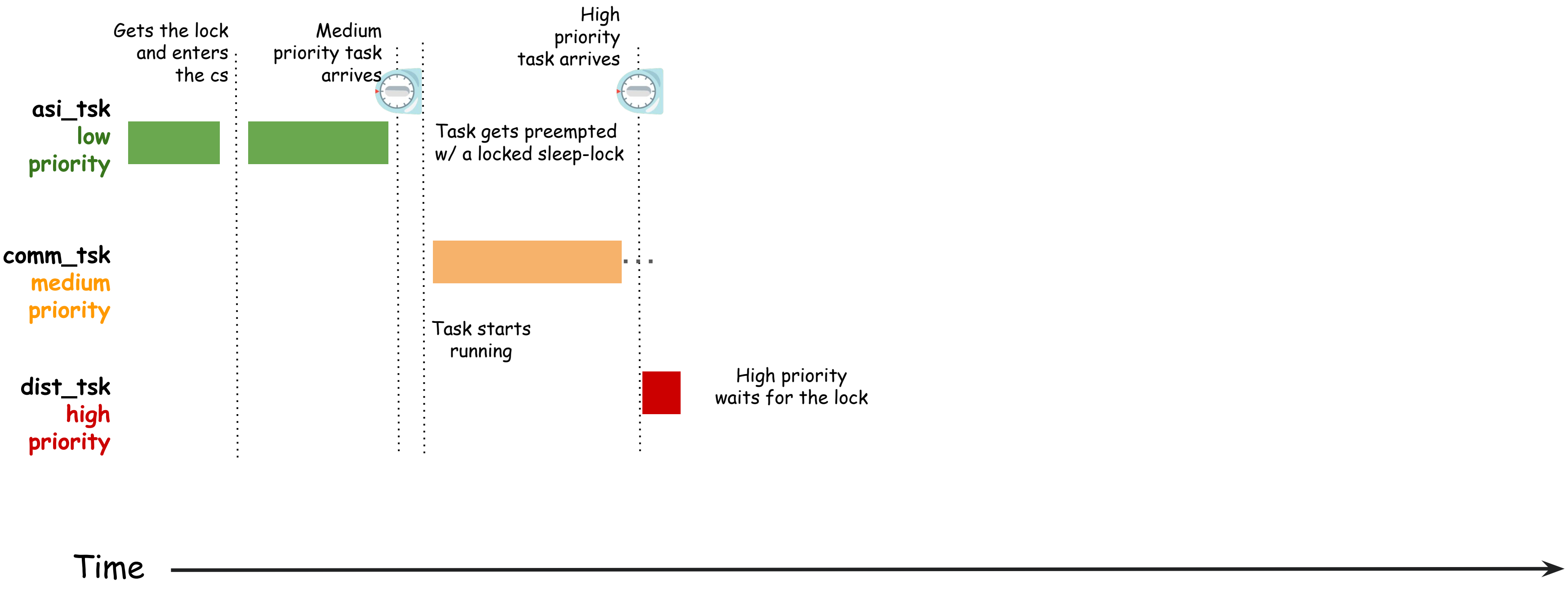
asi_tsk low priority

comm_tsk medium priority

dist_tsk high priority

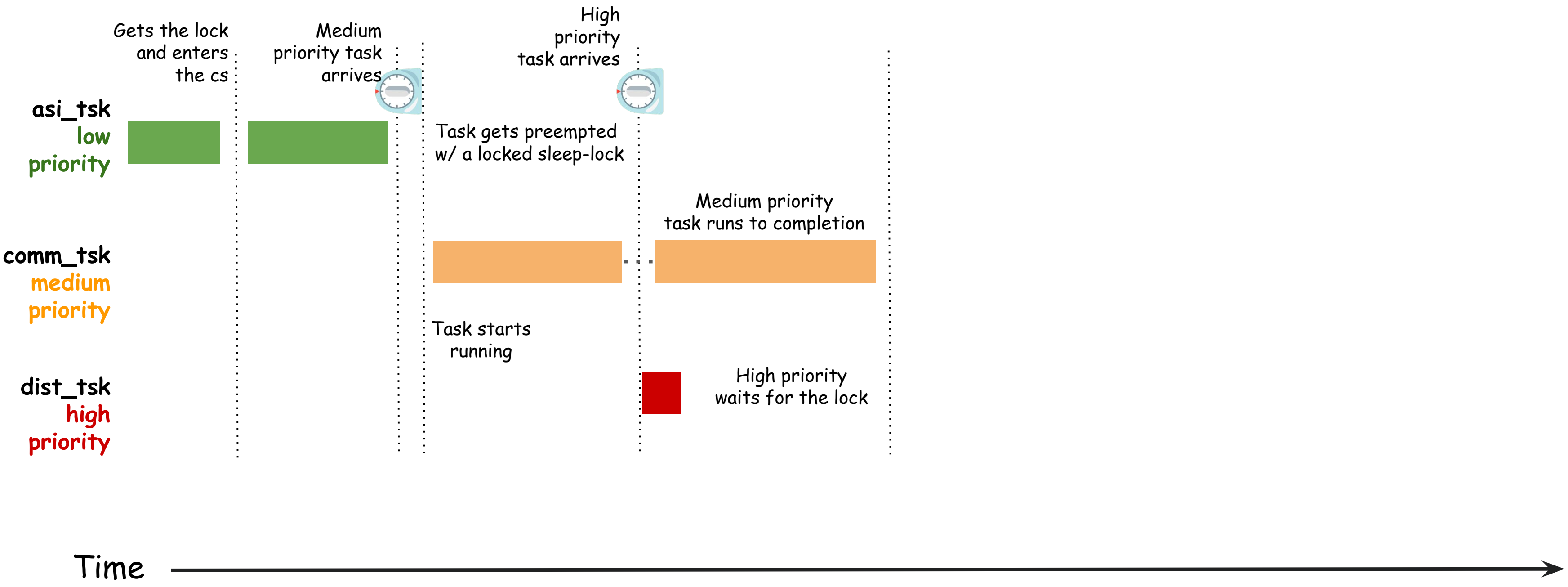
Time

Ready for your first bug in the outer universe?

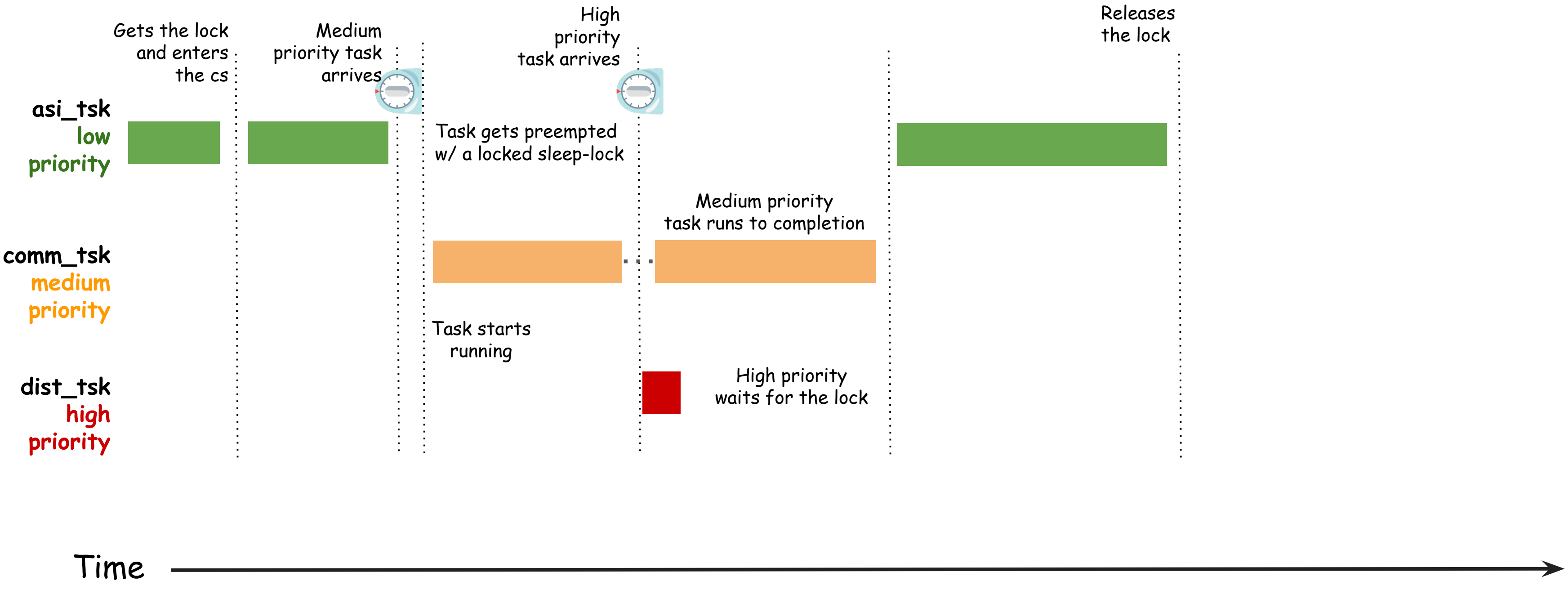


Gets the lock and enters the cs
Medium priority task arrives
High priority task arrives
asi_tsk
low priority
Task gets preempted w/ a locked sleep-lock
comm_tsk
medium priority
Task starts running
dist_tsk
high priority
High priority waits for the lock
Time

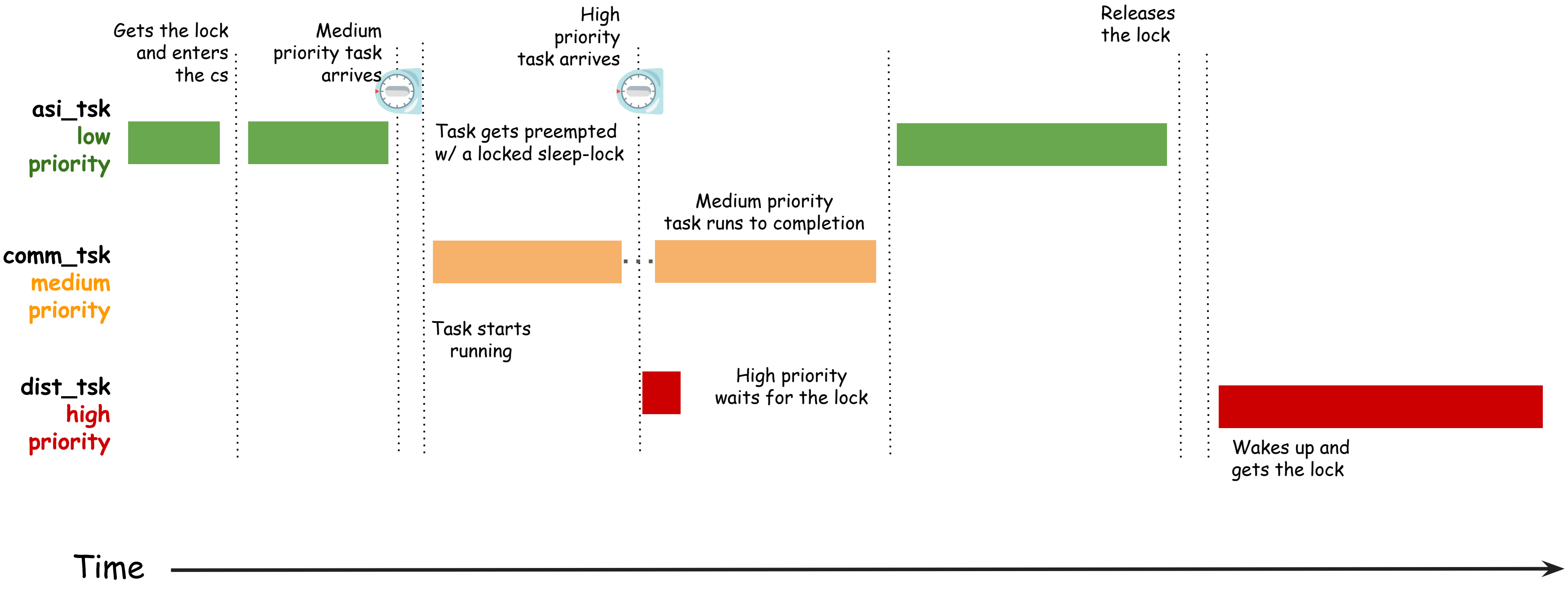
Ready for your first bug in the outer universe?



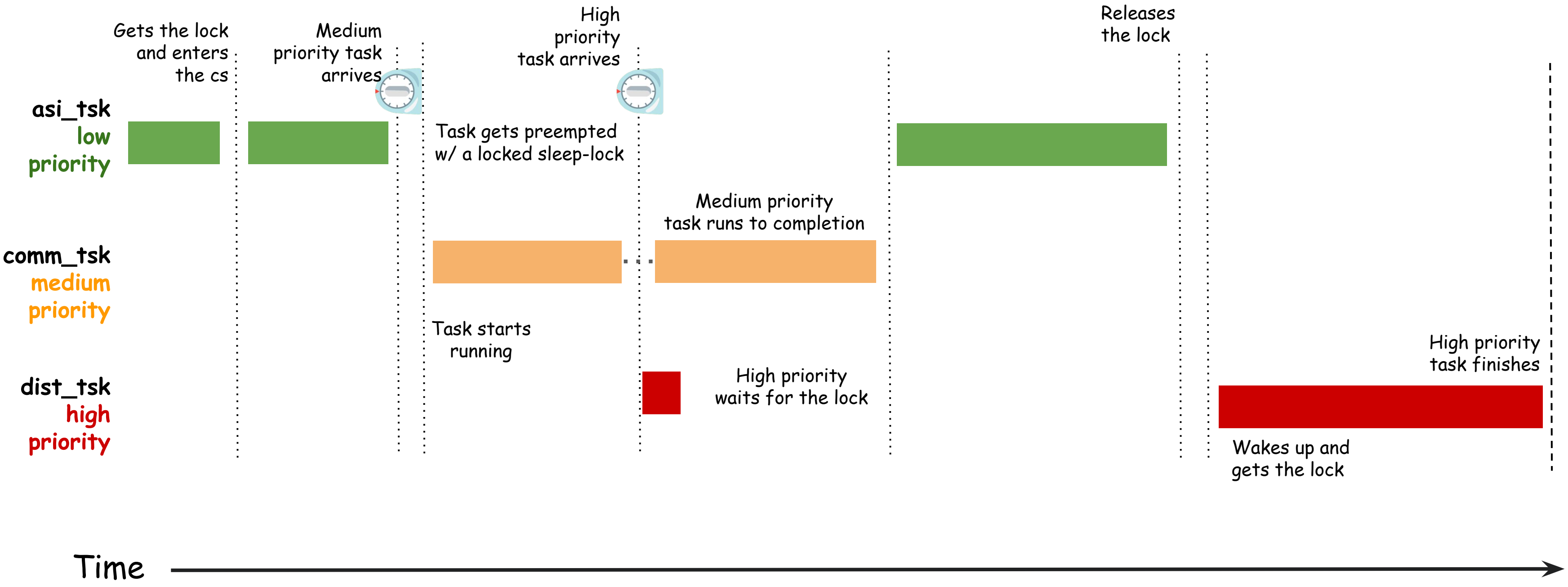
Ready for your first bug in the outer universe?



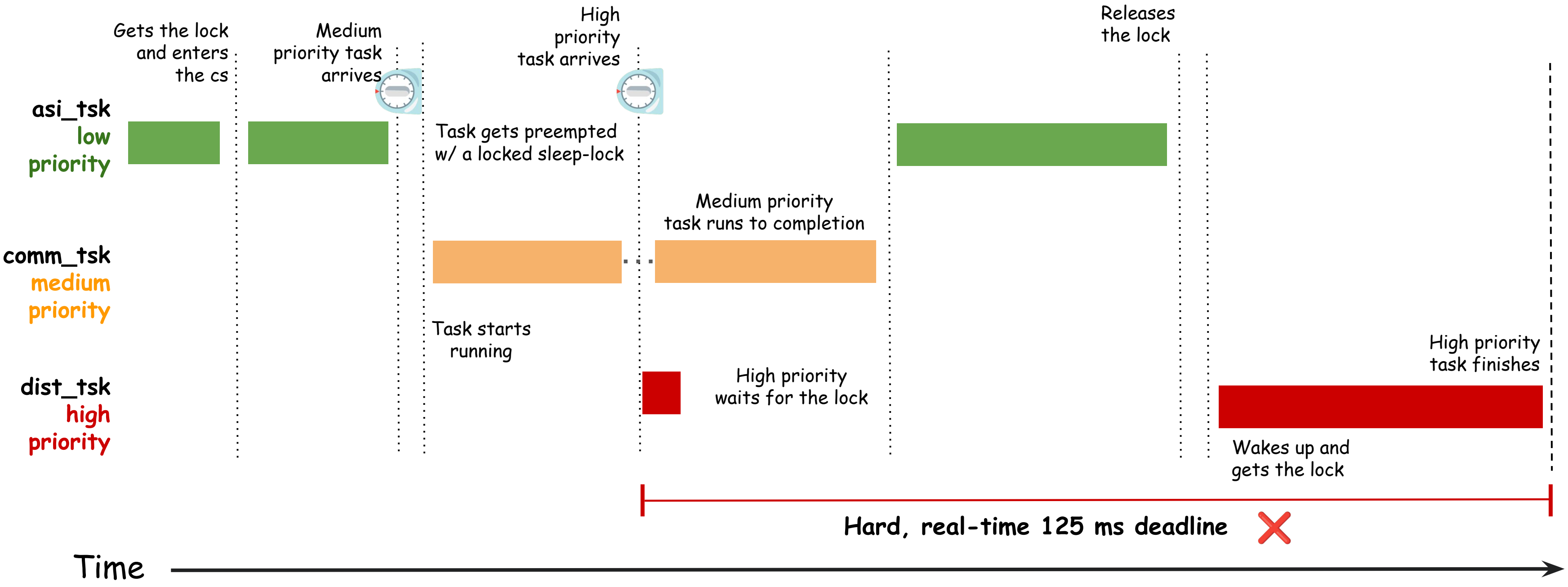
Ready for your first bug in the outer universe?


Gets the lock and enters the cs
Medium priority task arrives
High priority task arrives
Releases the lock
asi_tsk
low priority
Task gets preempted w/ a locked sleep-lock
Medium priority task runs to completion
comm_tsk
medium priority
Task starts running
dist_tsk
high priority
High priority waits for the lock
Wakes up and gets the lock
Time

Ready for your first bug in the outer universe?

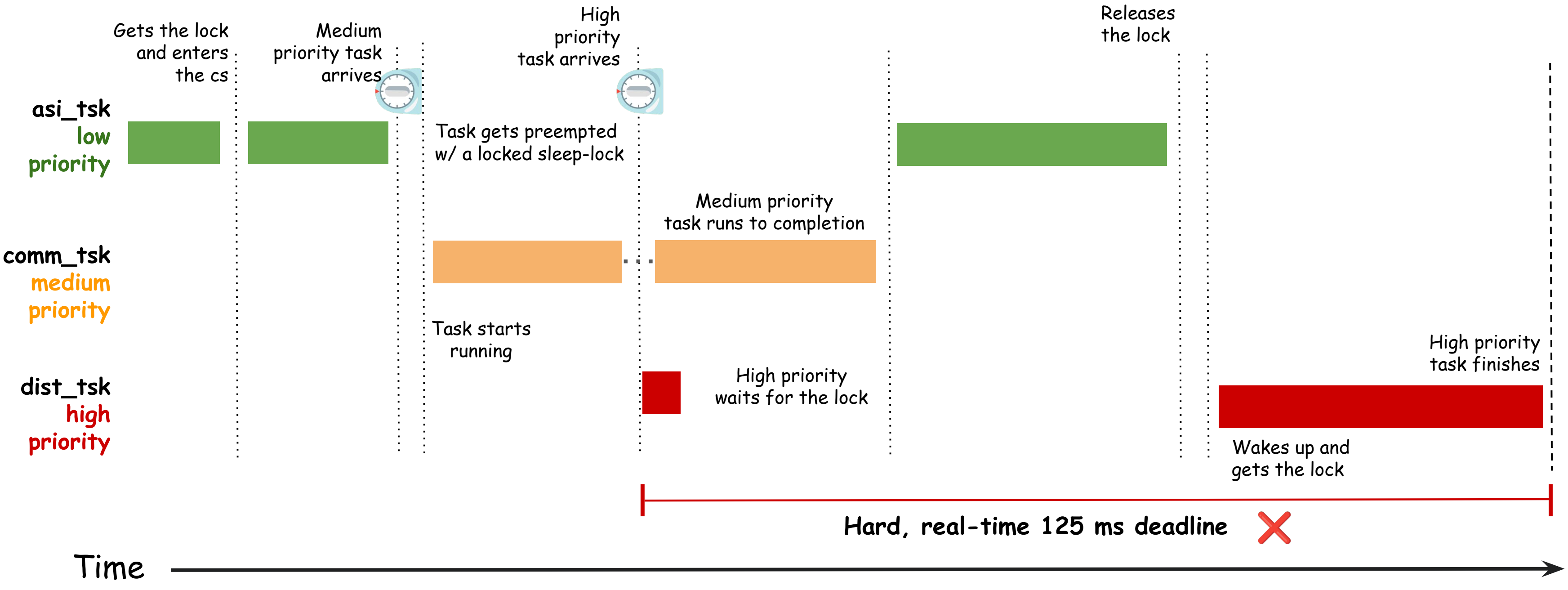


Ready for your first bug in the outer universe?

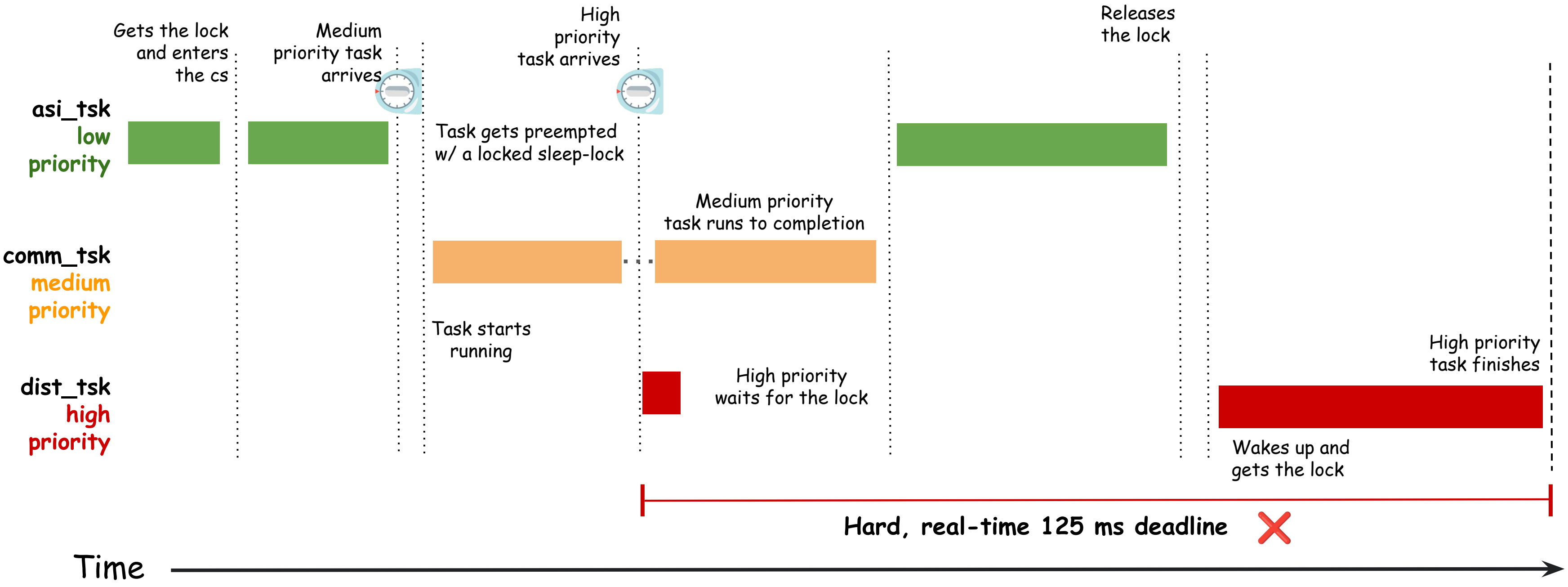


Gets the lock and enters the cs
Medium priority task arrives
High priority task arrives
Releases the lock
asi_tsk
low
priority
Task gets preempted w/ a locked sleep-lock
Medium priority task runs to completion
comm_tsk
medium
priority
Task starts running
High priority task finishes
dist_tsk
high
priority
High priority waits for the lock
Wakes up and gets the lock
Hard, real-time 125 ms deadline
Time

Classic example of priority inversion bug


Gets the lock and enters the cs
Medium priority task arrives
High priority task arrives
Releases the lock
asi_tsk
low priority
Task gets preempted w/ a locked sleep-lock
Medium priority task runs to completion
comm_tsk
medium priority
Task starts running
High priority task finishes
dist_tsk
high priority
High priority waits for the lock
Wakes up and gets the lock
Hard, real-time 125 ms deadline ❌
Time

Solution?

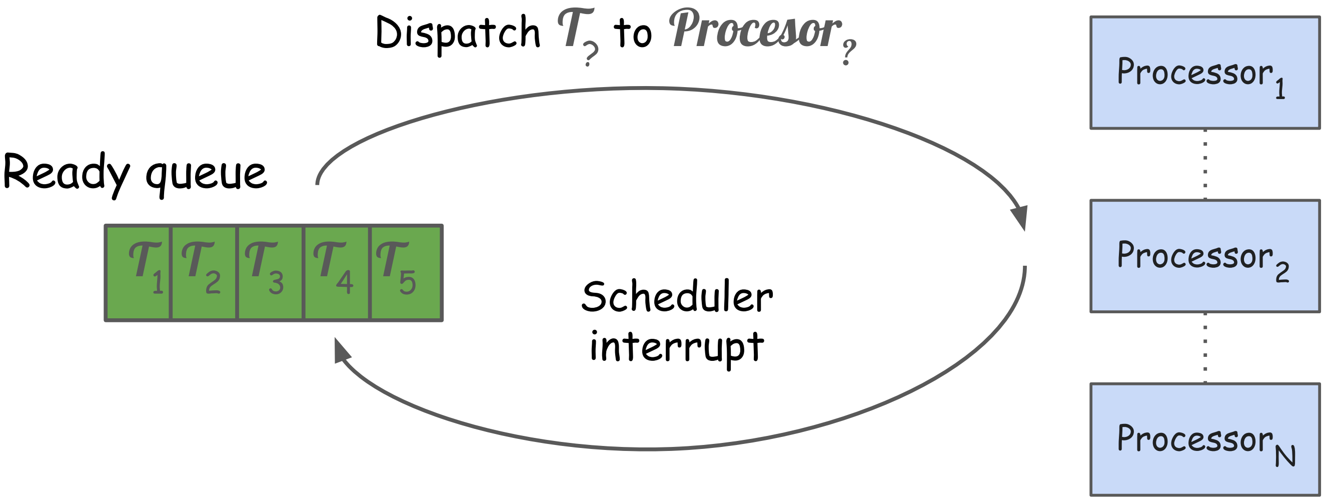

Gets the lock
and enters
the cs
Medium
priority task
arrives
High
priority
task arrives
Releases
the lock
asi_tsk
low
priority
Task gets preempted
w/ a locked sleep-lock
Medium priority
task runs to completion
comm_tsk
medium
priority
Task starts
running
High priority
task finishes
dist_tsk
high
priority
High priority
waits for the lock
Wakes up and
gets the lock
Hard, real-time 125 ms deadline ❌
Time

The Linux scheduler

Precedence Order	Scheduler class	Implemented policies	Usecase	POSIX compliance
1	stop_sched_class	Run Linux kernel-internal tasks	Only used internally by the kernel; preempts anything running in the local processor	No
2	dl_sched_class	SCHED_DEADLINE	Hard real-time tasks whose execution deadlines must be met	No
3	rt_sched_class	SCHED_FIFO, SCHED_RR	Soft real-time tasks (e.g., audio daemon) with priorities [1–99]	Yes
4	cfs_sched_class, eevdf_sched_class	SCHED_NORMAL, SCHED_BATCH, SCHED_IDLE	User tasks with "nice" values in the range [-20–19]	Partially Yes
5	idle_sched_class	Run the Linux kernel "idle" task	Runs when the local processor is idle, and has no other task to run	No

Unicore scheduling

> Given k tasks ready to run in a system with N available processors, which task should be dispatched to which processor at any given point in time?



Dispatch $\mathcal{T}_?$ to $\mathcal{Procesor}_?$

Ready queue

$\mathcal{T}_1$ $\mathcal{T}_2$ $\mathcal{T}_3$ $\mathcal{T}_4$ $\mathcal{T}_5$

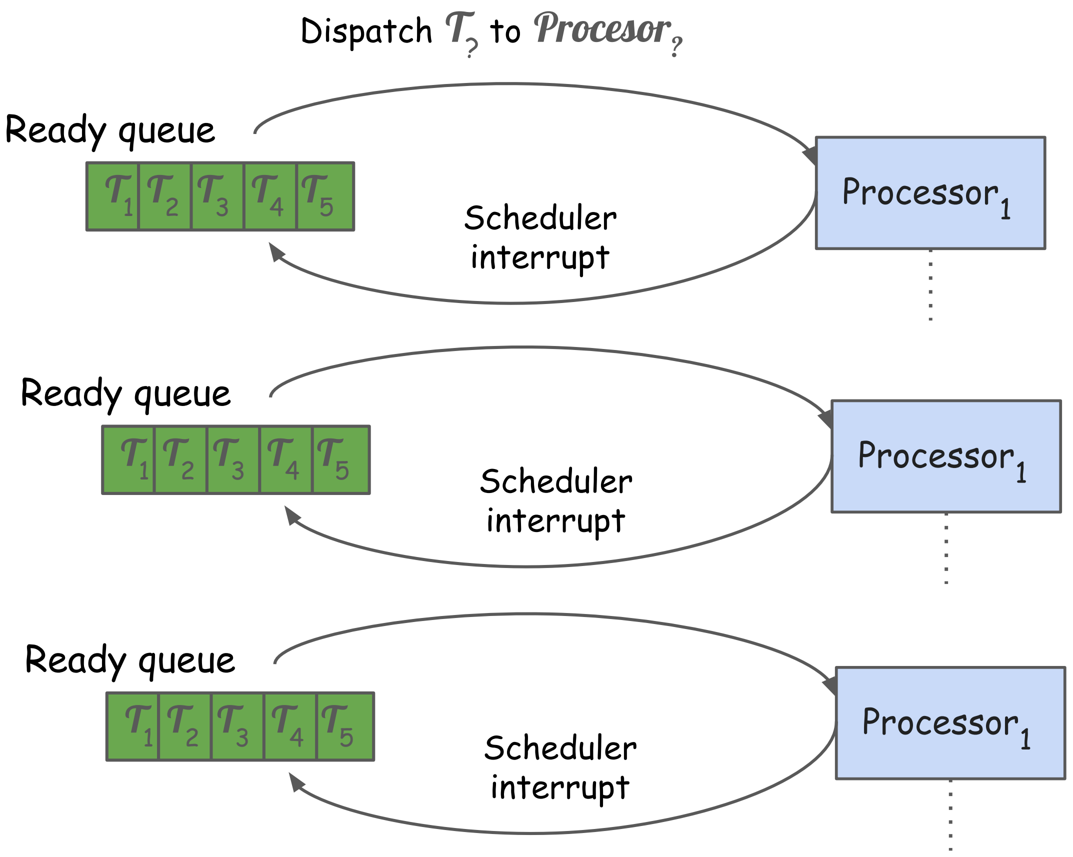
Scheduler interrupt

Processor_1

Processor_2

Processor_N

Multicore scheduling



Dispatch $T_?$ to $Procesor_?$

Ready queue

T_1 T_2 T_3 T_4 T_5

Scheduler interrupt

$Processor_1$

Ready queue

T_1 T_2 T_3 T_4 T_5

Scheduler interrupt

$Processor_1$

Ready queue

T_1 T_2 T_3 T_4 T_5

Scheduler interrupt

$Processor_1$