Operating Systems

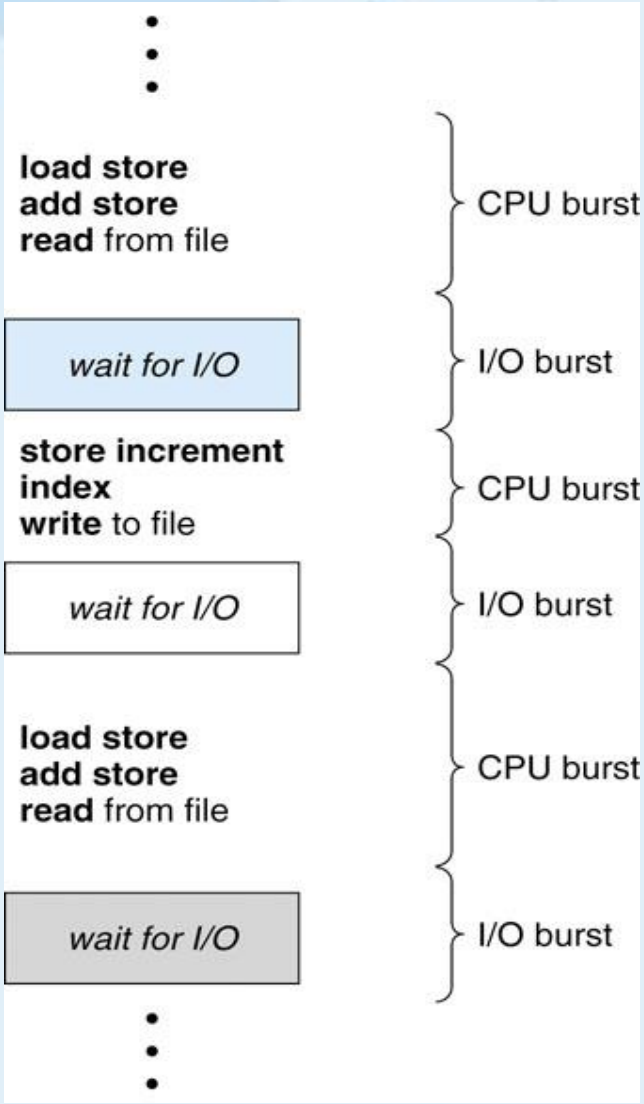
Lecture 3:

CPU Scheduling

Lecture Topics

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Algorithm Evaluation
- Summary

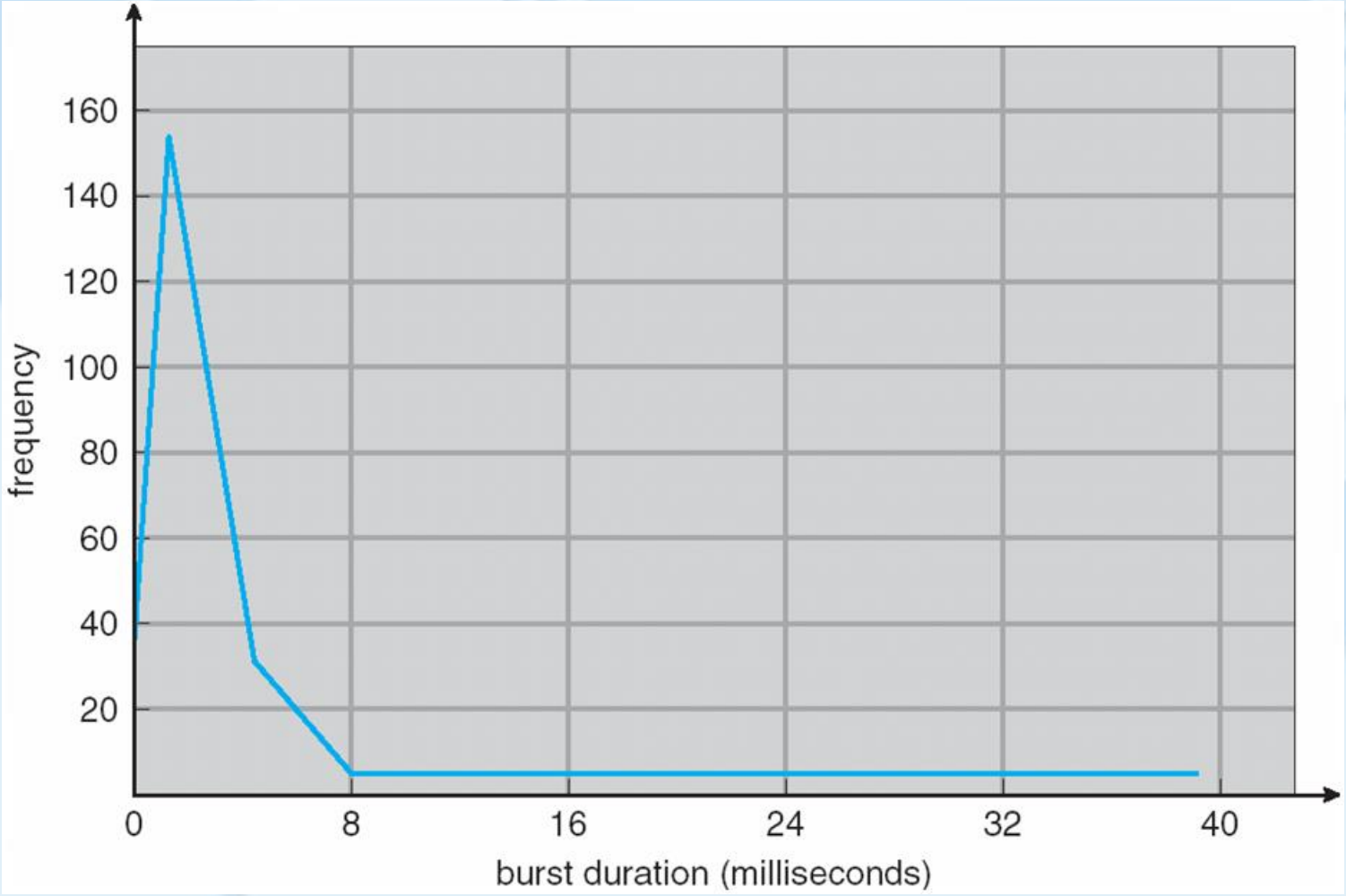
CPU-I/O Burst Cycle



- Process execution consists of a **cycle** of *CPU execution* and *I/O wait*.
- Process execution begins with a **CPU burst** followed by an **I/O burst**, then another **CPU burst**, and so on.
- Usually, in a process execution, there are a large number of **short** CPU bursts and small number of **long** CPU bursts.

CPU burst time

CPU burst time histogram



CPU Scheduler

- **Selects** a process in memory that is ready, and **allocates** the CPU to it.
- CPU scheduling decisions may take place when a process:
  1. Switches from **running** to **waiting** state,
  2. Switches from **running** to **ready** state,
  3. Switches from **waiting** to **ready** state,
  4. **Terminates**.
- Scheduling under 1 and 4 is **non-preemptive.**
- All other scheduling cases are **preemptive.**

Dispatcher

- This module gives the **control** of the CPU to the process selected by the CPU scheduler.
- It involves:
  - Context switch,
  - Switching to user mode,
  - Jumping to the proper location of the user program to restart that program.
- The time it takes to stop one process and start another is called **dispatch latency**.

Scheduling Measures

- **CPU utilization** - keep the CPU as busy as possible.
- **Throughput** – number of processes that complete their execution per time unit.
- **Turnaround time** - amount of time to execute a particular process.
- **Waiting time** – amount of time a process has been waiting in the ready queue.
- **Response time** – amount of time from the submission of a request until the first response is produced.

Scheduling Criteria

It is necessary:

- **To maximize**
  - CPU utilization,
  - Throughput.
- **To minimize**
  - Turnaround time,
  - Waiting time,
  - Response time.

Usually, the **average** measure is optimized, but in some cases, the **minimum** and **maximum** measures are also optimized.

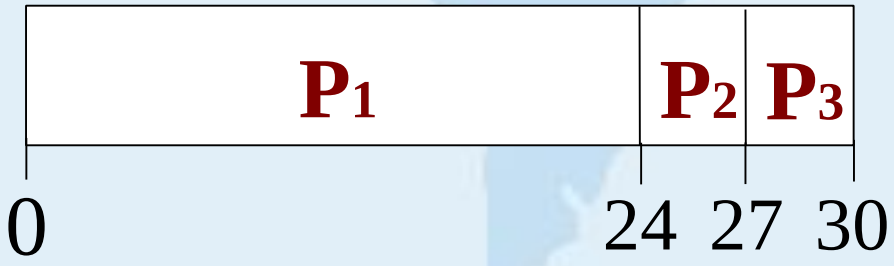
First-Come, First-Served (FCFS) Scheduling

- **FCFS** is the simplest CPU scheduling algorithm.
- **FCFS** can be realized by managing the ready queue as a FIFO queue.
- **FCFS** is simple and **non-preemptive**. The measures are often not good.

FCFS Scheduling

Gantt Chart

Process	Burst time (ms)
P_1	24
P_2	3
P_3	3



Turnaround Time (ms):
P1 = 24; **P2** = 27; **P3** = 30

Waiting Time (ms):
P1 = 0; **P2** = 24; **P3** = 27
Total: **P1** + **P2** + **P3** = 51

Average waiting time:
Total / 3 = 51 / 3 = **17 ms**

FCFS Scheduling

Gantt Chart

Process	Burst time (ms)
P_2	3
P_3	3
P_1	24

Turnaround Time:

P1 = 30; **P2** = 3; **P3** = 6

Waiting Time:

P1 = 6; **P2** = 0; **P3** = 3

Total: **P1** + **P2** + **P3** = 9

Average waiting time:

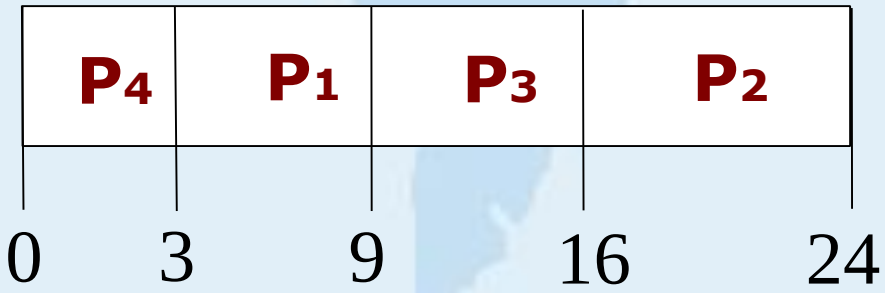
Total / 3 = 9 / 3 = **3 ms**

Shortest-Job-First (SJF) Scheduling

- **SJF** algorithm associates with each process the length of its next CPU burst.
- CPU is assigned to the process that has the smallest next CPU burst.
- **FCFS** is used for the processes with the *same* next CPU burst.
- The difficulty is knowing the length of the **next** CPU request.

SJF Scheduling

Process	Burst time (ms)
P_1	6
P_2	8
P_3	7
P_4	3



SJF average waiting time:

(3+16+9+0) / 4 = **7 ms**

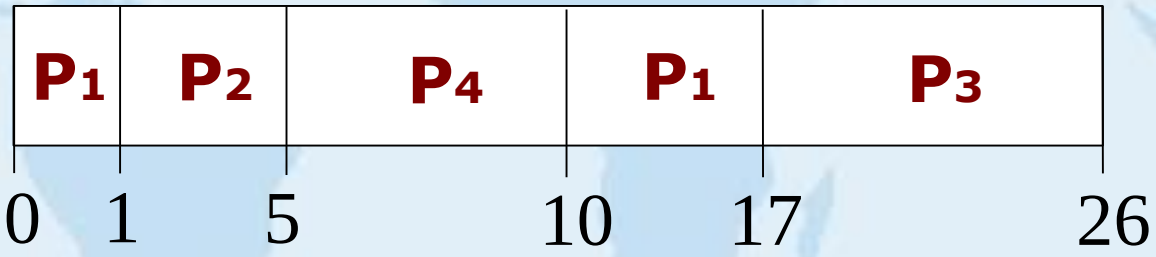
FCFS gives **10.25 ms**

Preemptive SJF Scheduling

- The SJF is **non-preemptive**. It allows the currently executing process to finish its CPU burst.
- A new process can have a **shorter** next CPU burst than processes currently executing.
- A preemptive SFJ algorithm will **preempt** the currently executing process.
- Such an approach is called **Shortest Remaining Time First (SRTF)** scheduling.

SRTF Scheduling

Process	Arrival time	Burst time (ms)
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5



Average waiting time: (**SJF** time = **7.75 ms)**

[(10-1)+(1-1)+(17-2)+(5-3)] / 4 = **6.5 ms**

Priority Scheduling

- A **priority** number (integer) is associated with each process.
- CPU is allocated to the process with the **highest** priority (smaller integer = higher priority).
- The **SJF** algorithm is a special case of the priority scheduling where the priority is the **inverse** of the next CPU burst.
- **Problem**: **Starvation** – low priority processes may never execute.
- **Solution**: **Aging** – as time progresses increase priority of the processes.

Priority Scheduling

Process	Burst time	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

Average waiting time (6+0+16+18+1) / 5 = **8.2 ms**

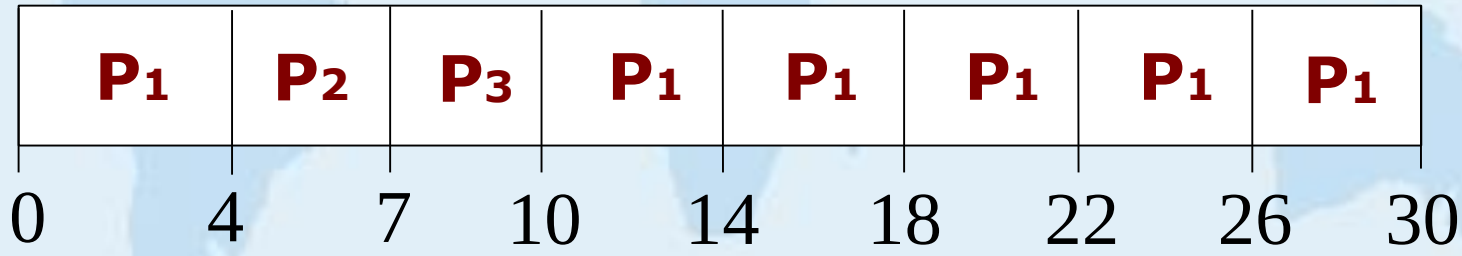
Round-Robin (RR) Scheduling

- A small unit of time, **time quantum** or **time slice**, is defined. The ready queue is treated as a **FIFO** circular queue. New processes are put to the **tail** of the ready queue.
- The CPU scheduler picks the first process from the queue and **sets** a **timer** to interrupt after 1 time quantum, and dispatches the process. If the dispatched process has a CPU burst longer than 1 time quantum, the process is put to the tail of the ready queue.

RR Scheduling

Process	Burst time (ms)
P_1	24
P_2	3
P_3	3

The time quantum is 4 ms.

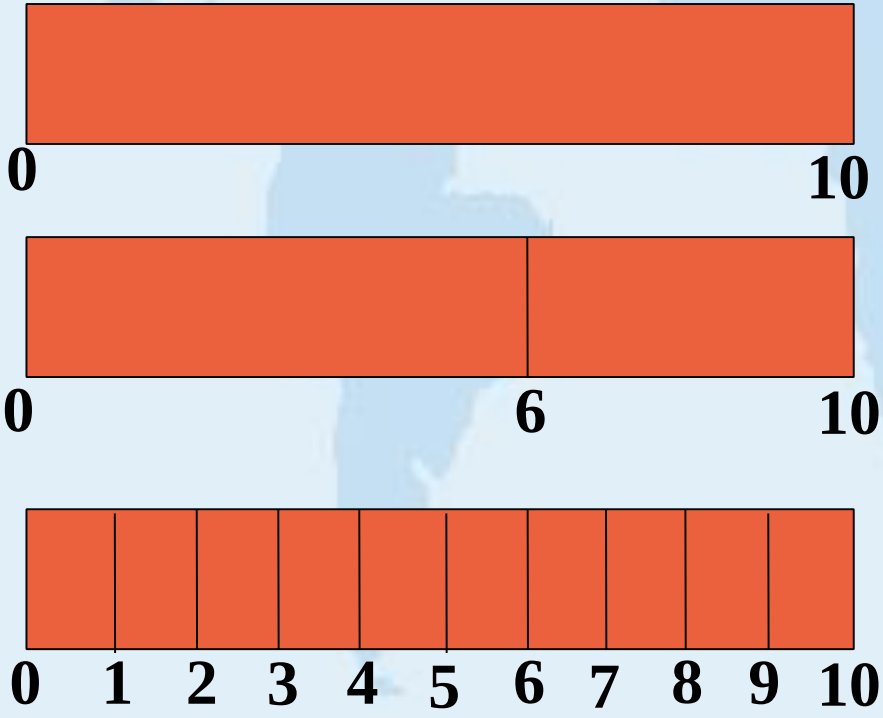


Average waiting time (6+4+7) / 3 = **5.66 ms**

RR Scheduling

The effect of time quantum length – **smaller** time quantum **increases** context switches.

Process burst time = 10 ms

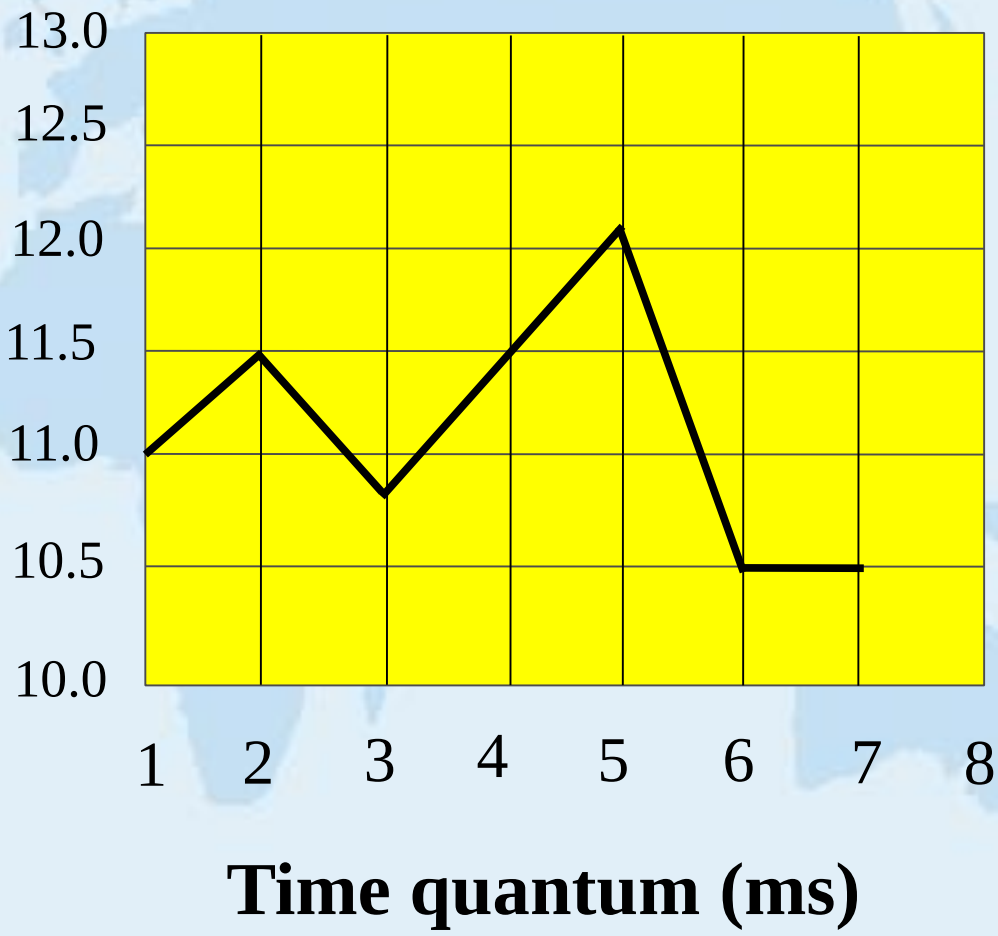


Quantum length	# of context switches
12	0
6	1
1	9

RR Scheduling

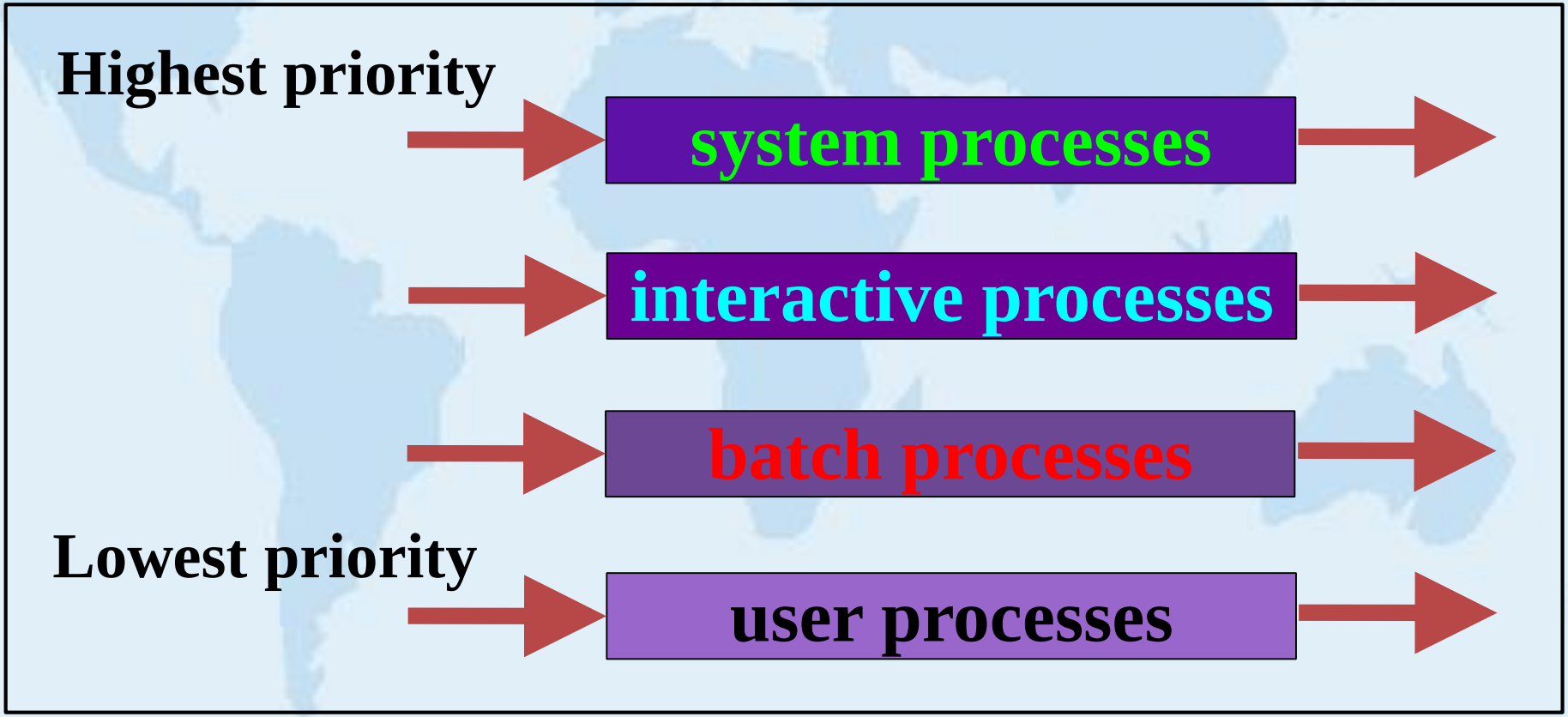
Average turnaround time (ms) for the following processes:

Process	Time
P_1	6
P_2	3
P_3	1
P_4	7



Multilevel queue scheduling

Ready queue



Algorithm Evaluation

How to select the scheduling algorithm?

Evaluation methods:

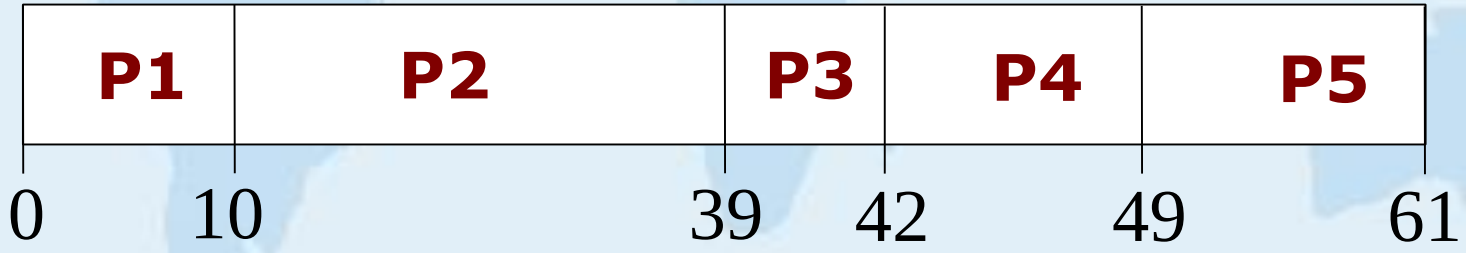
- Deterministic modelling.
- Queuing models.
- Simulations.
- Implementation.

Deterministic modelling

- This method takes a particular **predetermined** workload and defines the performance of each algorithm for the workload.
- It is simple, fast, and gives **exact answers**.
- But it needs exact data of inputs and the answers apply to **only** those cases.

FCFS performance

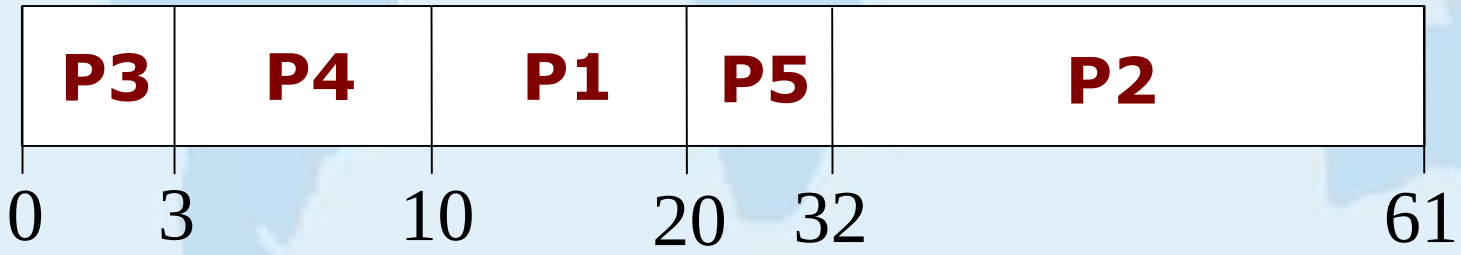
Process	Burst time (ms)
P1	10
P2	29
P3	3
P4	7
P5	12



Average waiting time (0+10+39+42+49) / 5 = **28 ms**

SJF performance

Process	Burst time (ms)
P1	10
P2	29
P3	3
P4	7
P5	12

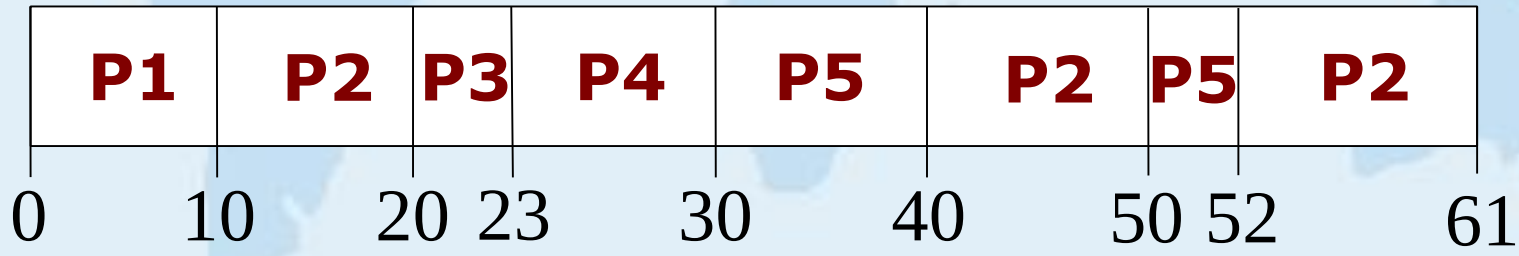


Average waiting time (10+32+0+3+20) / 5 = **13 ms**

RR performance

Process	Burst time (ms)
P1	10
P2	29
P3	3
P4	7
P5	12

Quantum time = 10 ms



Average waiting time (0+32+20+23+40) / 5 = **23 ms**

Evaluation Results

- For the **FCFS** algorithm,

  Average waiting time is **28 ms.**

- For the **SJF** algorithm,

  Average waiting time is **13 ms.**

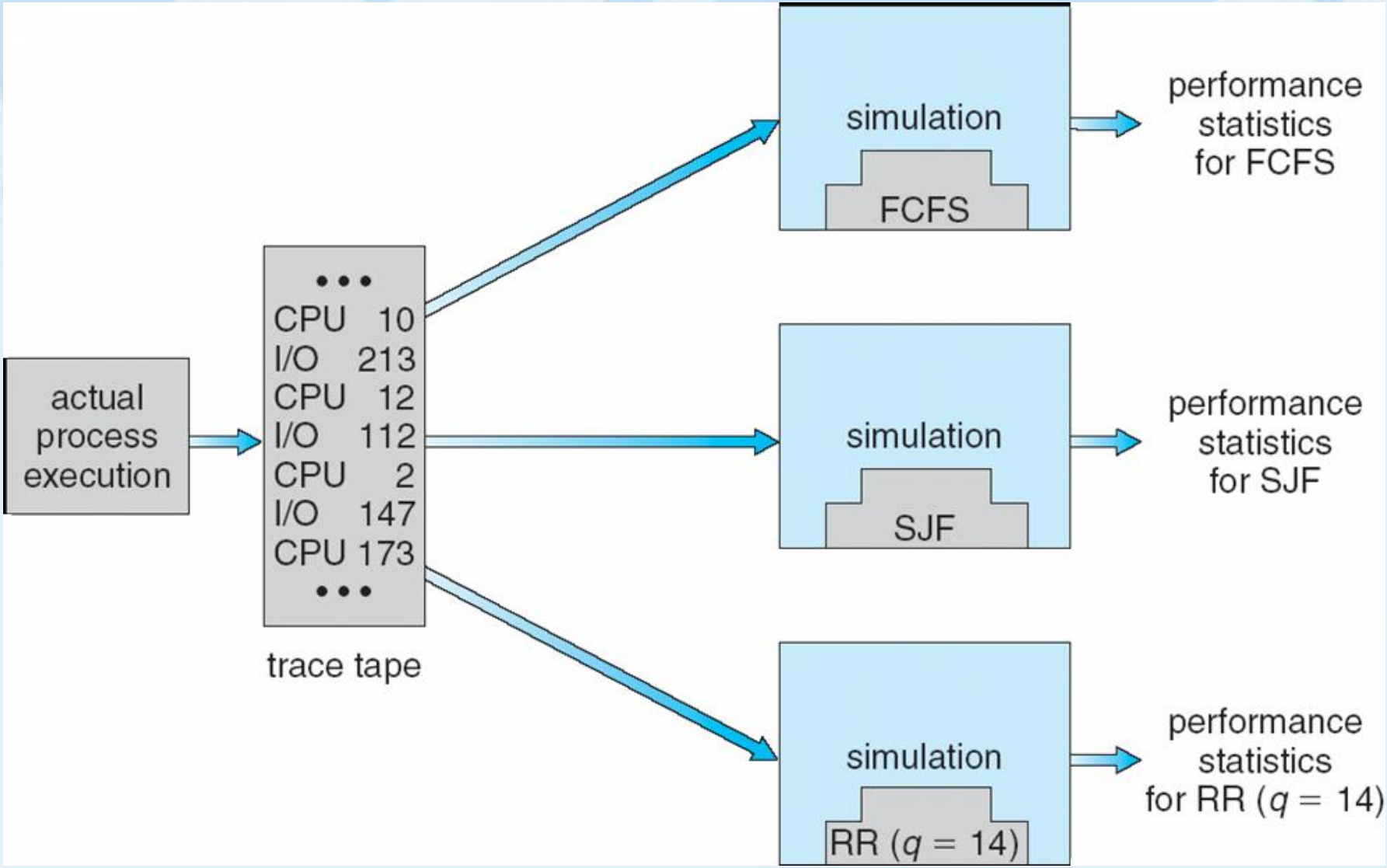
- For the **RR** algorithm,

  Average waiting time is **23 ms.**

Queuing models

- The computer system is described as a **network** of servers.
- The CPU is a **server** with its ready queue of processes.
- Knowing arrival rates and service rates, we can **compute** utilization, average queue length, average wait time, and so on.
- This area of study is called **queuing-network analysis**.

Simulations



Implementation

- The **only** completely accurate way to evaluate a scheduling algorithm is
  - **To code it,**
  - **To put it in the OS,**
  - **To see how it works.**
- We evaluate an algorithm under **real** operating conditions.
- **Difficulties** can be the cost, changing current environment and user's requirements.

Summary - scheduling

- **CPU scheduling** is to select a process from the ready queue and allocate the CPU to it.
- **FCFS algorithm** is the simplest one, but the performance is not good.
- **SJF algorithm** gives the shortest average waiting time, but it is difficult to implement.
- SJF is a special case of **priority scheduling**.
- **Round Robin (RR) algorithm** is more appropriate for time-shared system.

Summary - scheduling

- The **FCFS** is **non preemptive**.
- The **RR** is **preemptive**.
- The **SJF** and **priority scheduling** can be **either** pre-emptive or non pre-emptive.

Summary - evaluation

- **Analytic methods** use mathematical analysis to determine the performance.
- **Simulation methods** determine the performance by imitating samples of processes.
- **In the queuing-network analysis**, the computer system is described as a network of servers.
- **Implementation** evaluates an algorithm under real operating conditions.

That is all for today!