

9' Esercitazione

<https://politecnicomilano.webex.com/meet/gianenrico.conti>

27 maggio 2021

Gian Enrico Conti

Scheduling, Memoria virtuale

Architettura dei Calcolatori e Sistemi Operativi 2020-21

Outline

- **Scheduling**

- FCFS
- RR
- SRR
- SJF
- SPN
- HRRN
- SRT
- Multilevel Queues
- Multilevel Feedback Queues

FCFS –First Come First Serve

- Dati i seguenti processi con i rispettivi tempo di arrivo, tempo di servizio

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

- Disegnare il grafico temporale dello scheduling
- Calcolare il tempo medio di attesa

$$T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$$

- Calcolare il rapporto di prestazioni

$$R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$$

FCFS –First Come First Serve: Grafico

- FCFS: uno dopo altro
- Dobbiamo aspettare ogni fine
- durata complessiva: $4+3+6+8$
- Vengono COMUNQUE accodati.

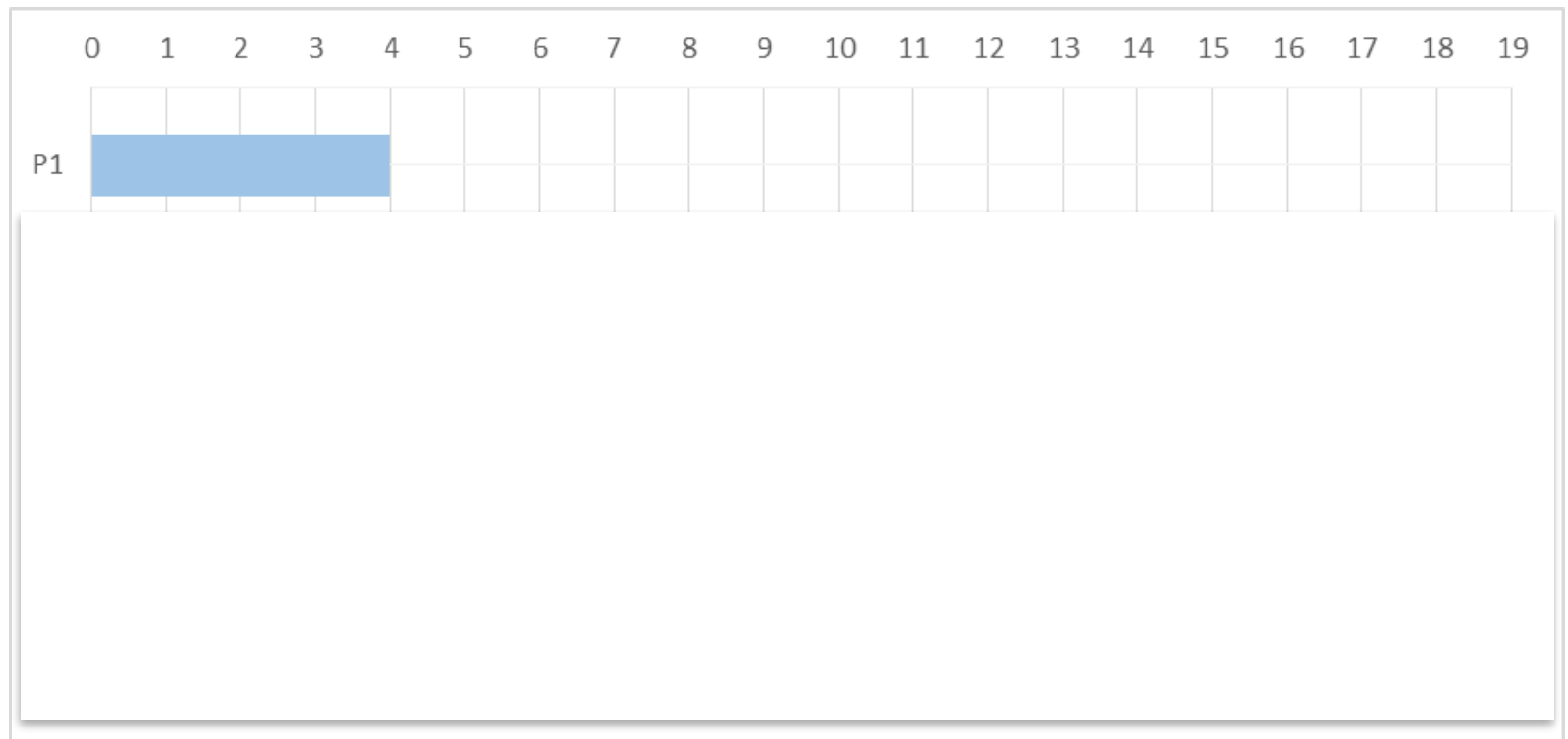
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

...

FCFS –First Come First Serve: Grafico

– P1

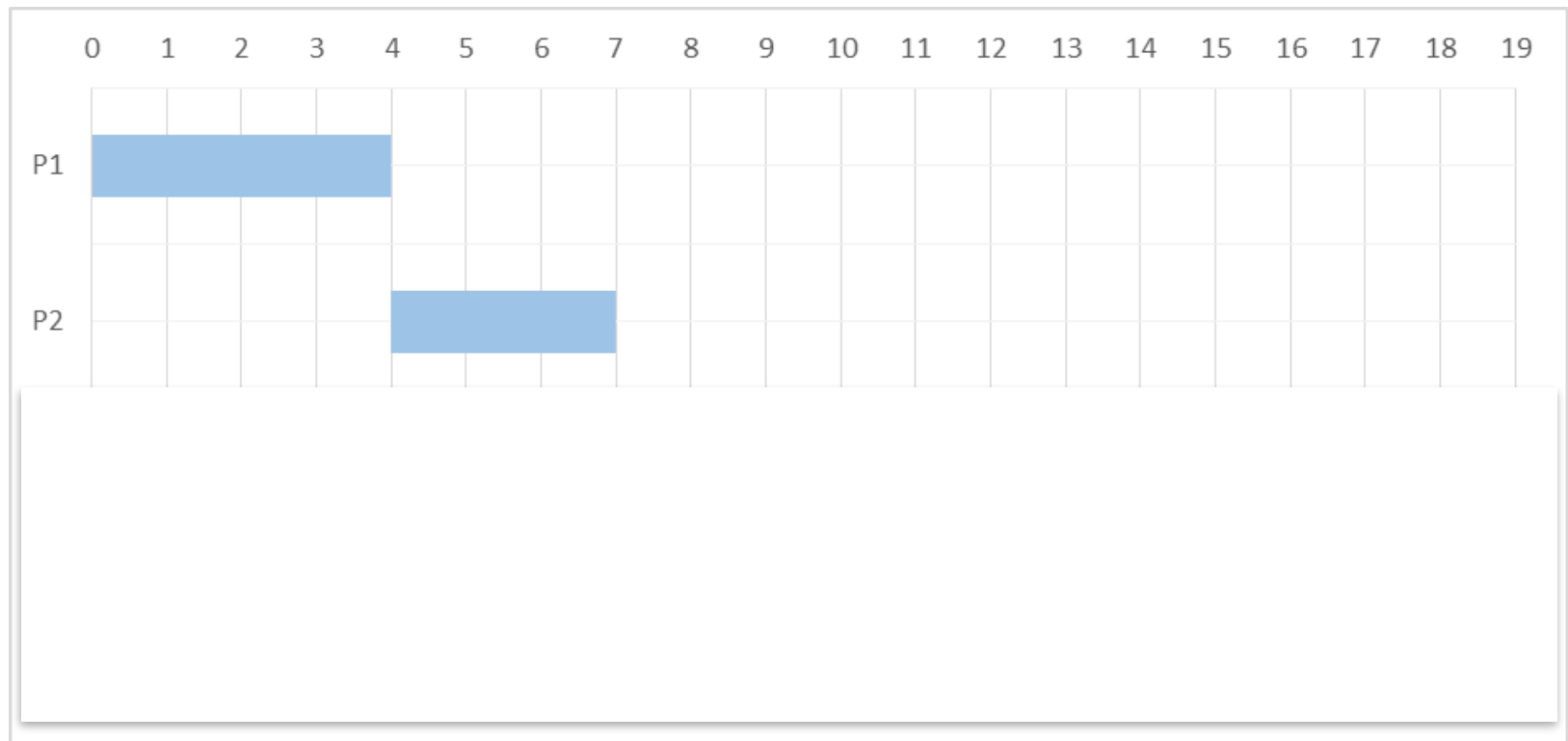
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



FCFS –First Come First Serve: Grafico

– P2

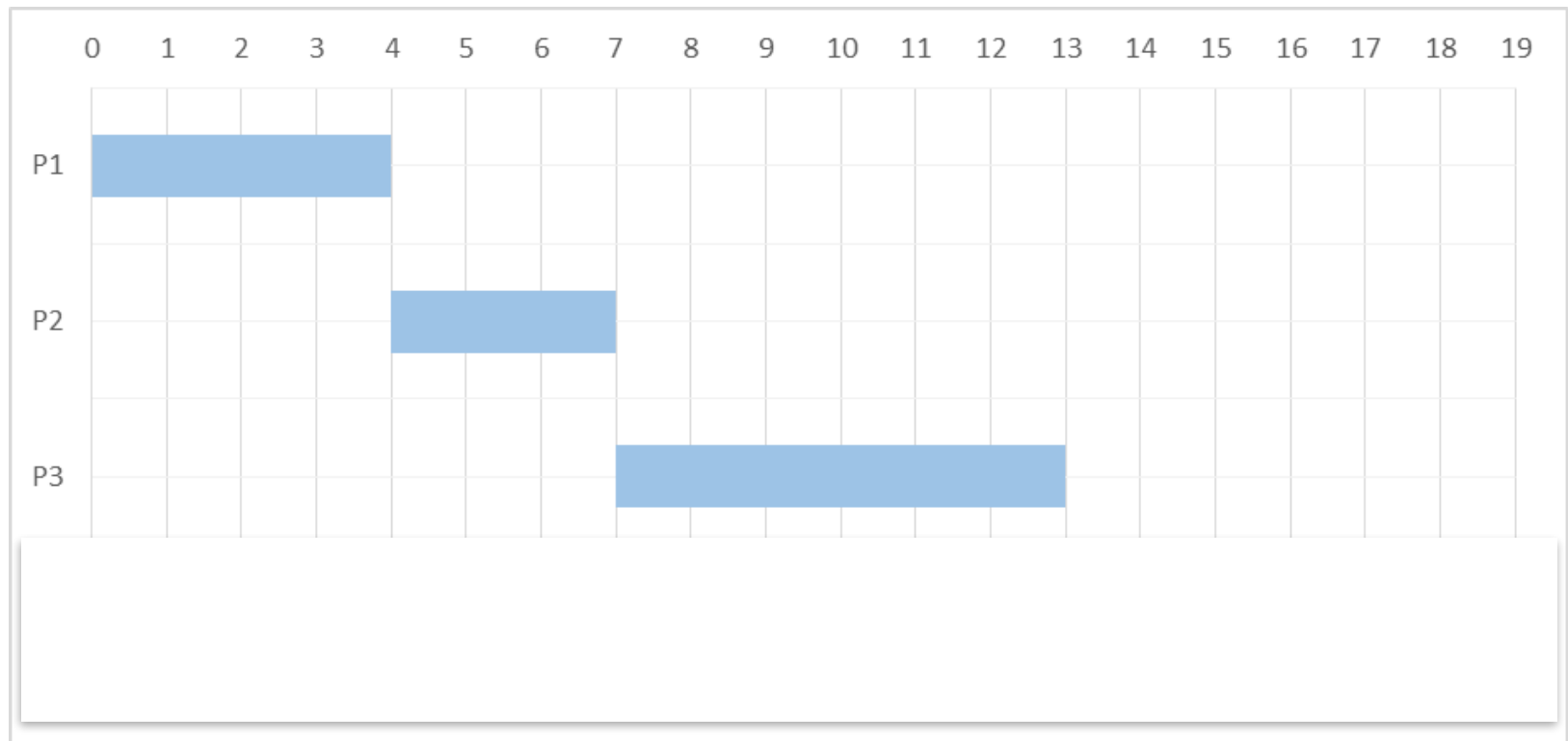
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



FCFS –First Come First Serve: Grafico

– P3

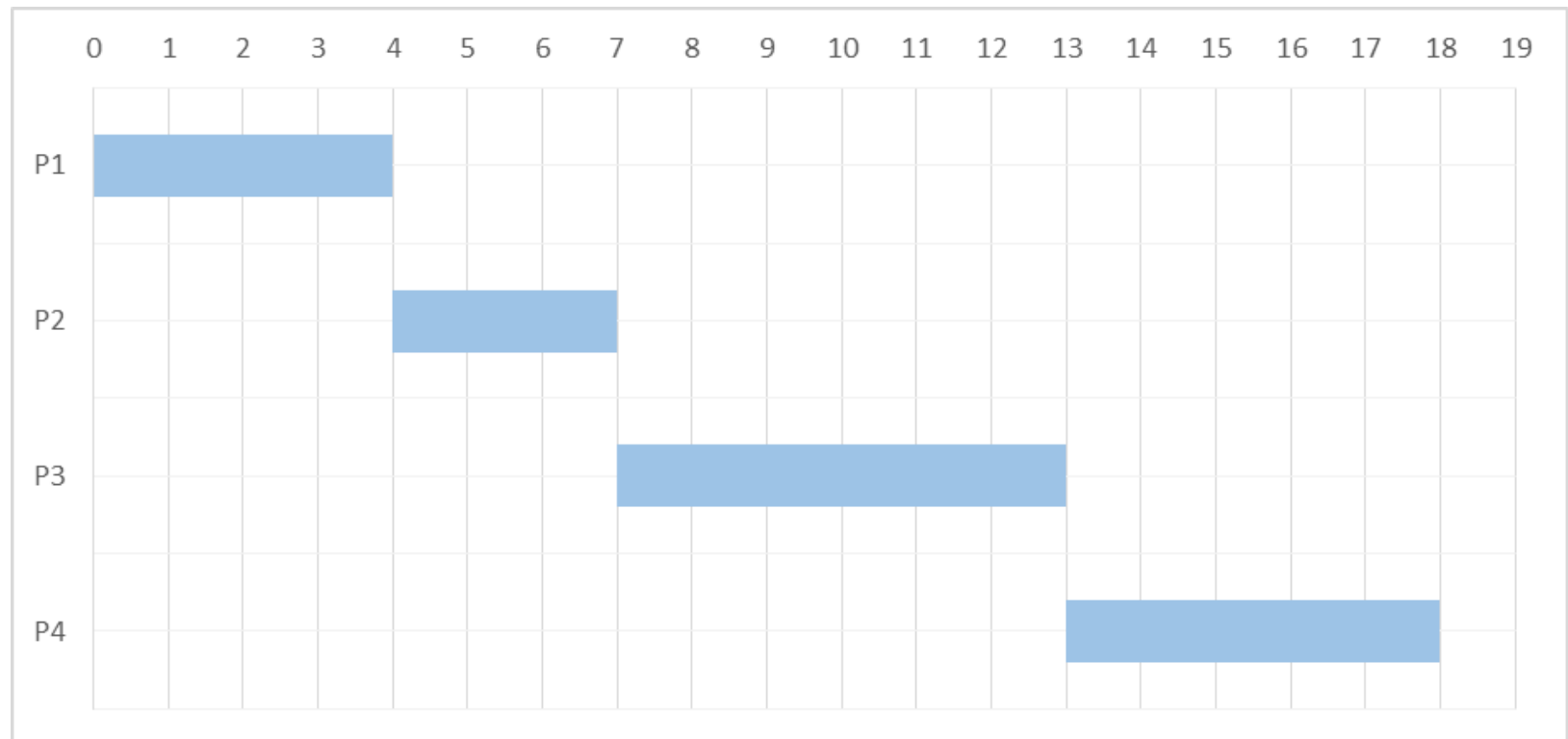
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



FCFS –First Come First Serve: Grafico

– P4

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



FCFS: tempo medio di attesa NOTA:

– nella formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

– T fine = DAL GRAFICO TEMPORALE

– T servizio = Dalla Tabella

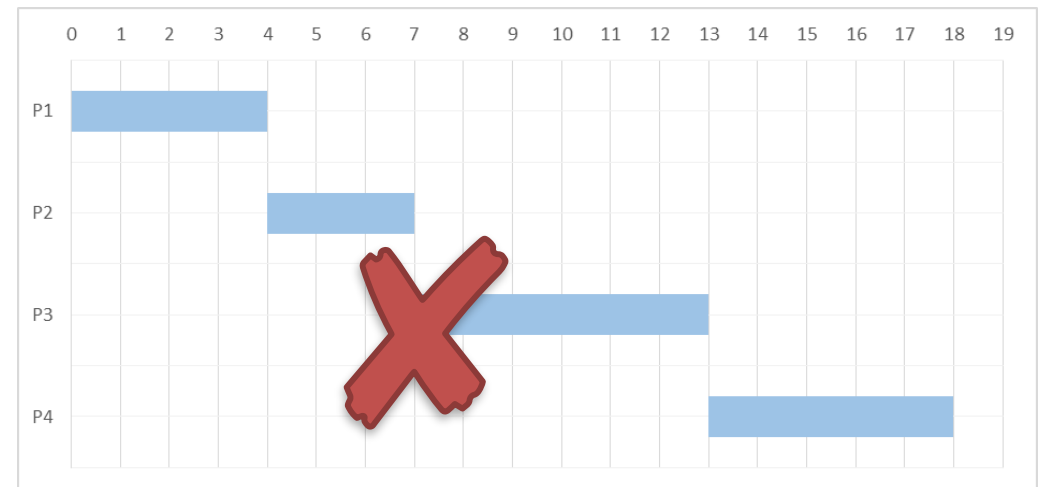
– T arrivo = Dalla Tabella

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

– ES:

T Arrivo di P3 e' **3**, NON 7!

– Ci interessa sapere quando avrebbe dovuto partire!

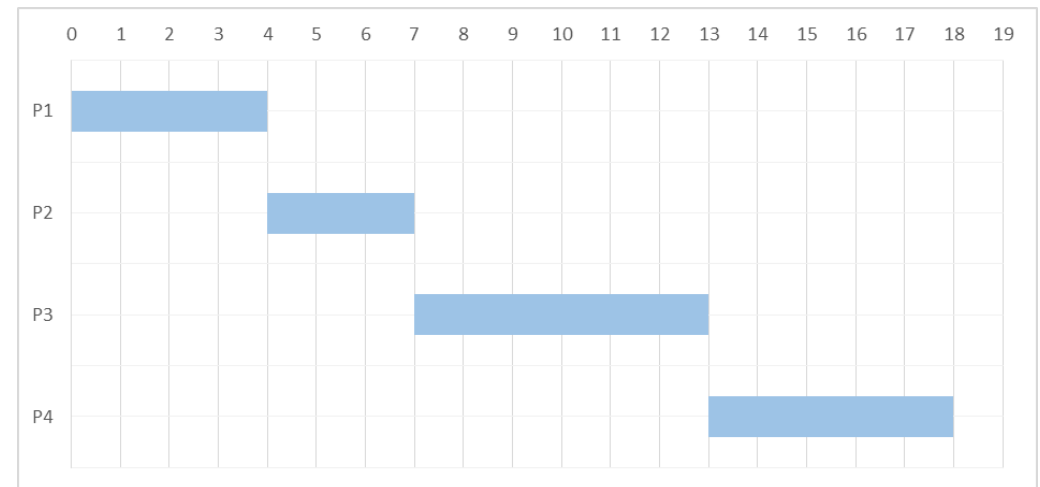


FCFS: tempo medio di attesa P1

– formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

– $T_{a P1} = 4 - 4 - 0$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



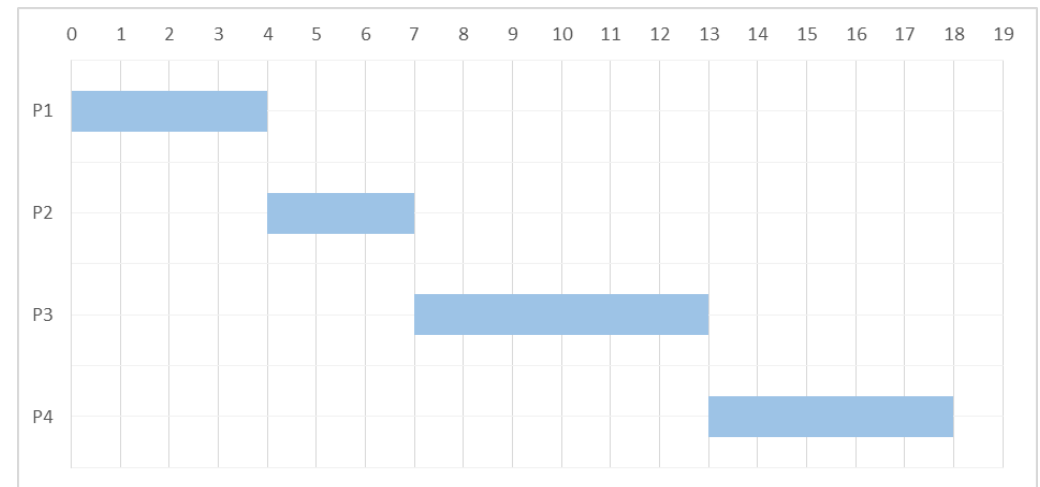
FCFS: tempo medio di attesa P2

– formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

– $T_{a P1} = 4 - 4 - 0$

– $T_{a P2} = 7 - 3 - 1$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



FCFS: tempo medio di attesa P3

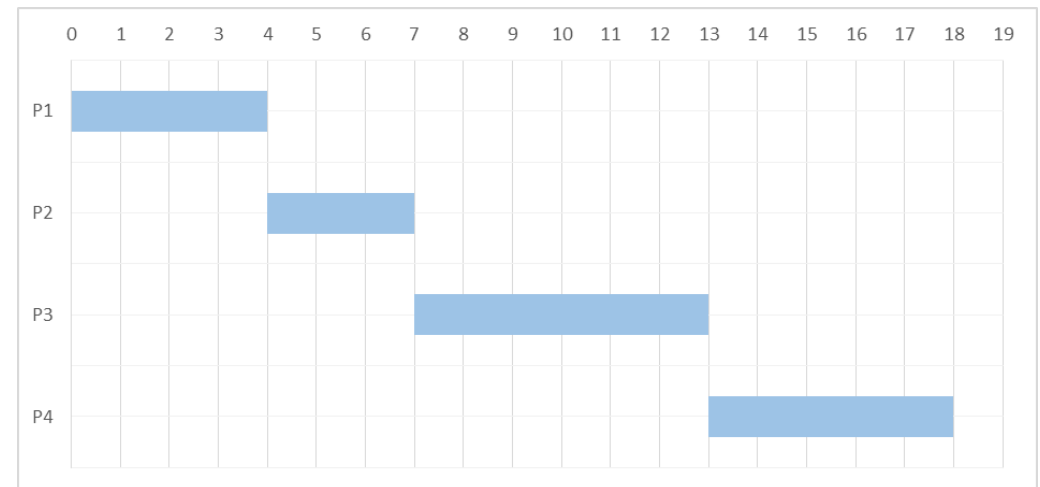
– formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

– $T_{a P1} = 4 - 4 - 0 = 0$

– $T_{a P2} = 7 - 3 - 1 = 3$

– $T_{a P3} = 13 - 6 - 3 = 4$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



FCFS: tempo medio di attesa P4

– formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

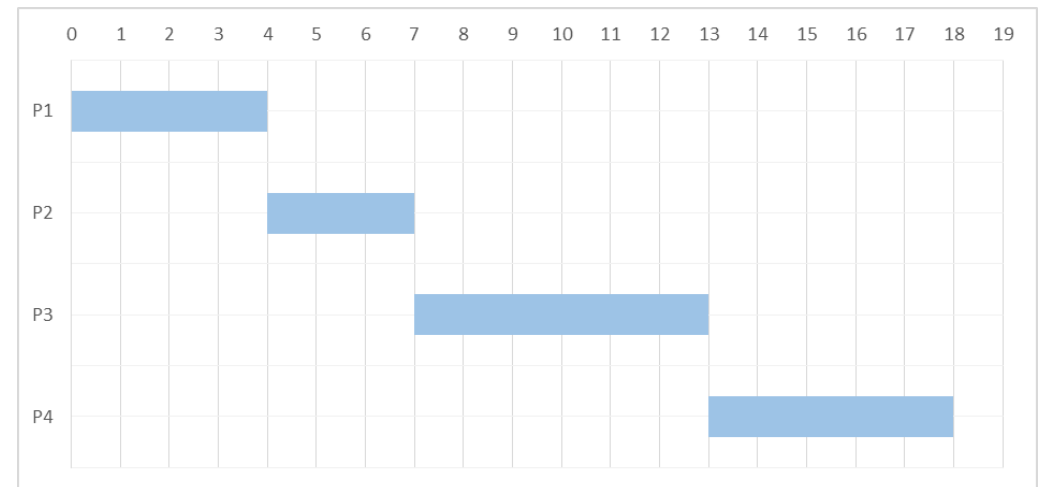
– $T_{a P1} = 4 - 4 - 0 = 0$

– $T_{a P2} = 7 - 3 - 1 = 3$

– $T_{a P3} = 13 - 6 - 3 = 4$

– $T_{a P4} = 18 - 5 - 5 = 8$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



FCFS: tempo medio di attesa: attesa media

– formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

– $T_{a P1} = 4 - 4 - 0 = 0$

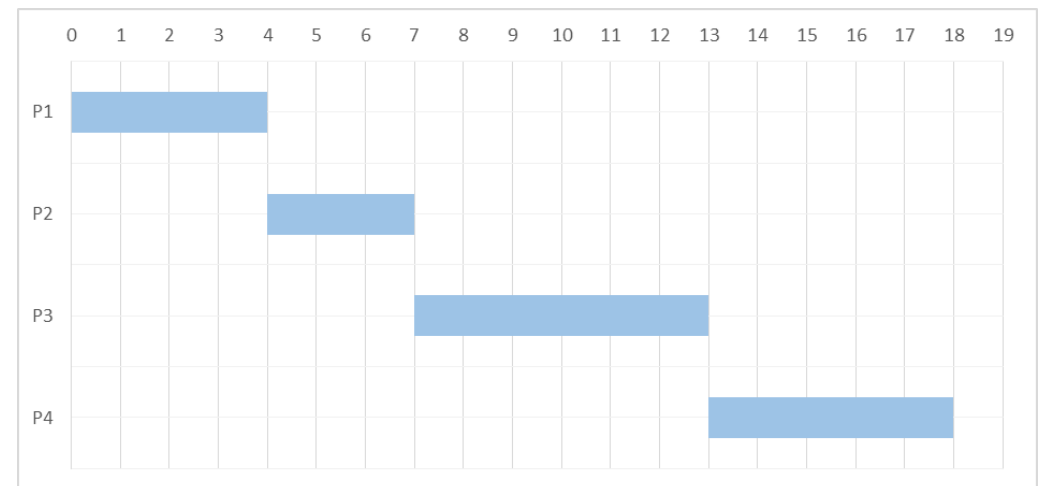
– $T_{a P2} = 7 - 3 - 1 = 3$

– $T_{a P3} = 13 - 6 - 3 = 4$

– $T_{a P4} = 18 - 5 - 5 = 8$

– **→ Attesa media = $15/4 = 3.75$**

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

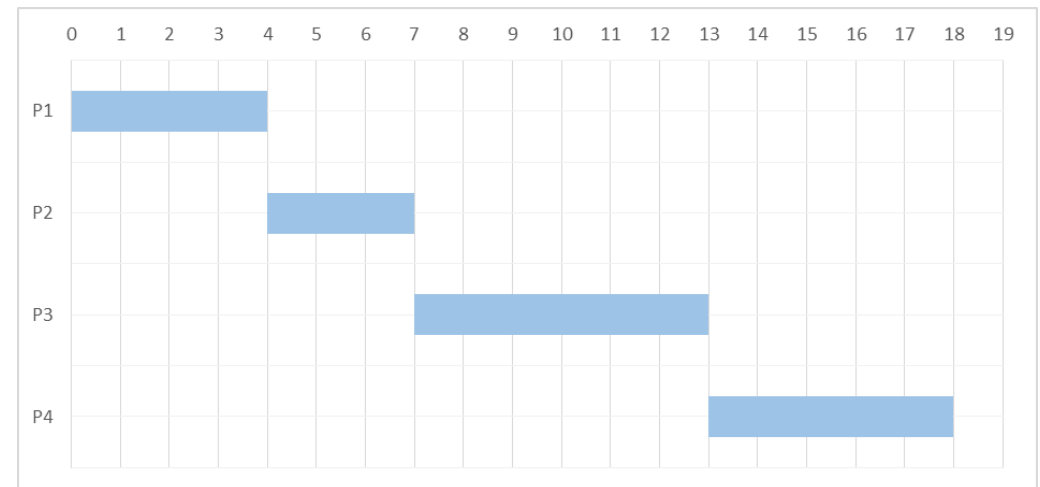


FCFS:rapporto di prestazioni P1

– formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

– $R_{P1} = 4 / (4-0)$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



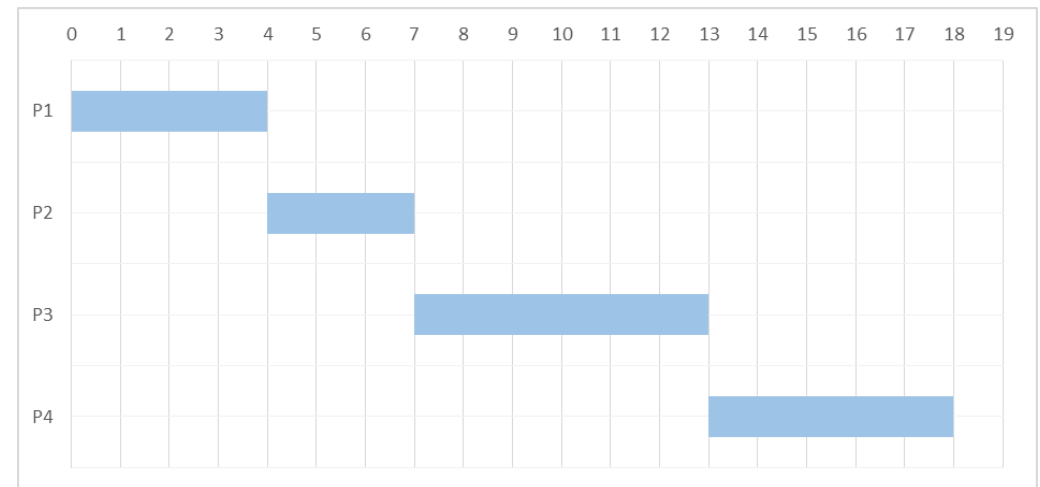
FCFS:rapporto di prestazioni P2

– formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

– $R_{P1} = 4 / (4-0) = 1$

– $R_{P2} = 3 / (7-1) = 0.5$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



FCFS:rapporto di prestazioni P3

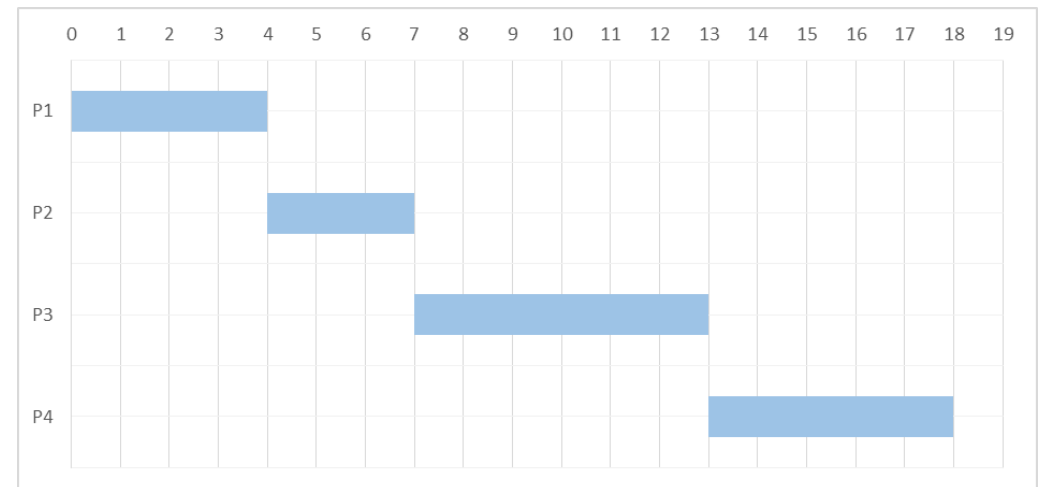
– formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

– $R_{P1} = 4 / (4-0) = 1$

– $R_{P2} = 3 / (7-1) = 0.5$

– $R_{P3} = 6 / (13-3) = 0.6$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



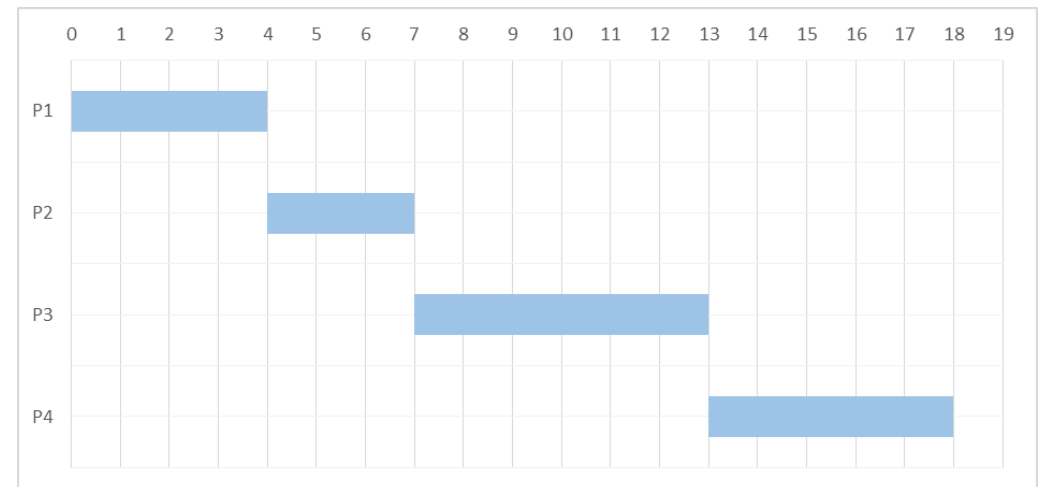
FCFS:rapporto di prestazioni P4

– formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

- $R_{P1} = 4 / (4-0) = 1$
- $R_{P2} = 3 / (7-1) = 0.5$
- $R_{P3} = 6 / (13-3) = 0.6$
- $R_{P4} = 5 / (18-5) = 0.3846$

→ Prestazioni medie =
 $(1+0.5+0.6+0.38)$
 $= 2.48 / 4 =$
 0.62

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



RR – Round Robin

- Based on FCFS (i.e. FIFO)
- Processes run only for a limited amount of time called a time slice or quantum
- Preemptible

RR – Round Robin

- Dati i seguenti processi con i rispettivi tempo di arrivo, tempo di servizio

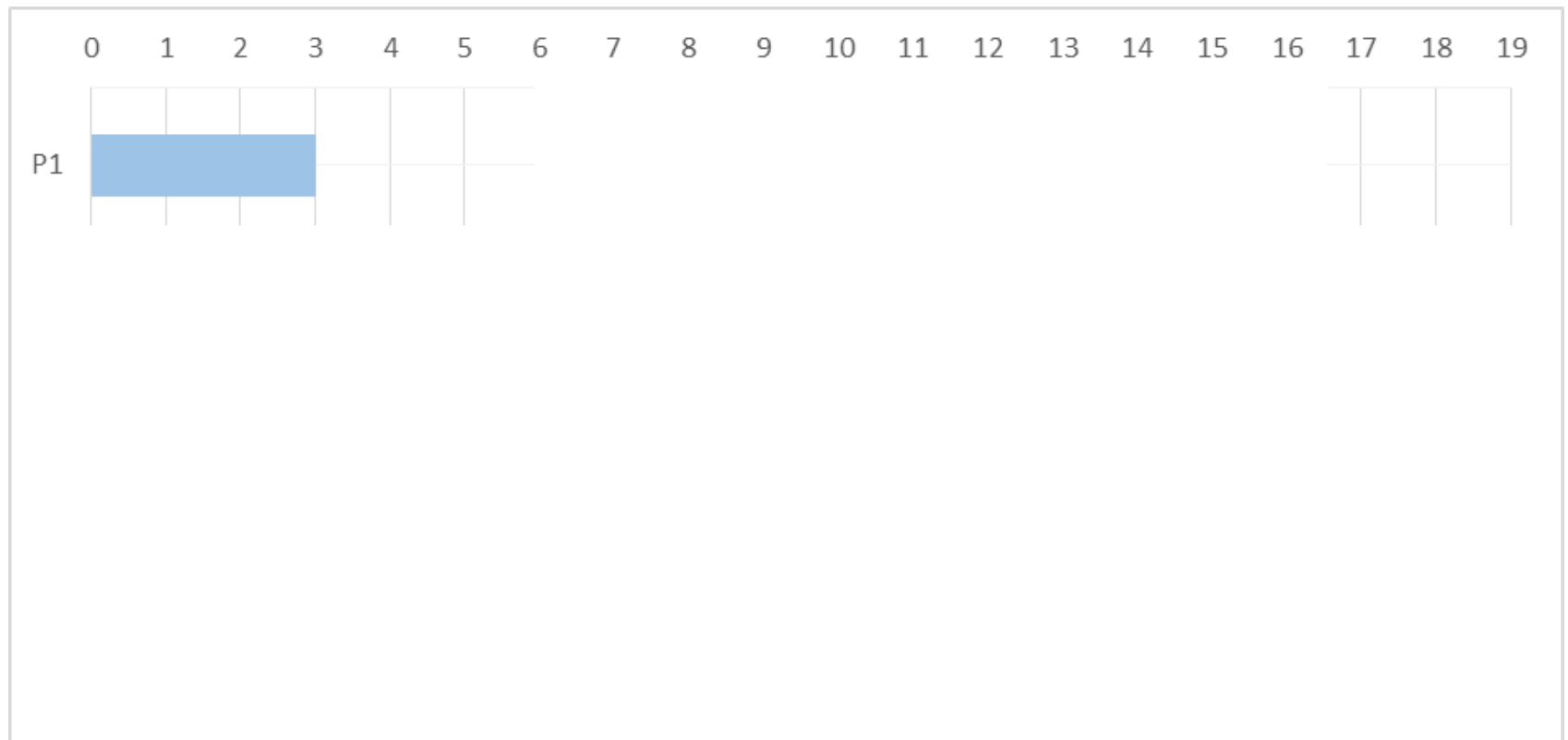
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

- Quanto di tempo = 3
 - Disegnare il grafico temporale dello scheduling
 - Calcolare il tempo medio di attesa
 - Calcolare il rapporto di prestazioni

RR – Round Robin: grafico P1

- (Quanto di tempo = 3)
- Si noti P1 viene interrotto dopo 3 Q
- Toccherà a P2 che arriva al $T = 1$

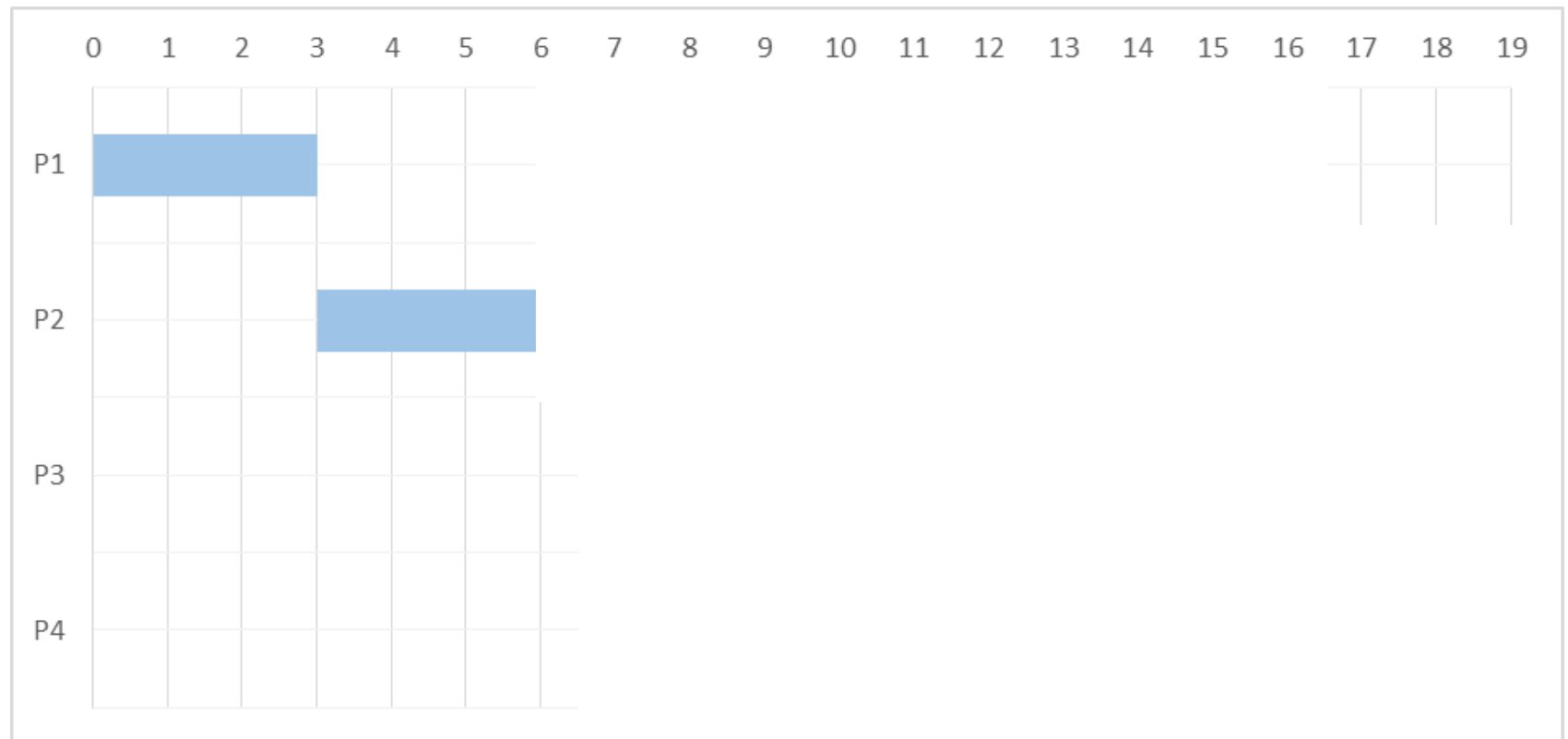
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



RR – Round Robin: grafico P2

- (Quanto di tempo = 3)
- (P1 e' interrotto)
- P2 ha durata $3 \leq QT$

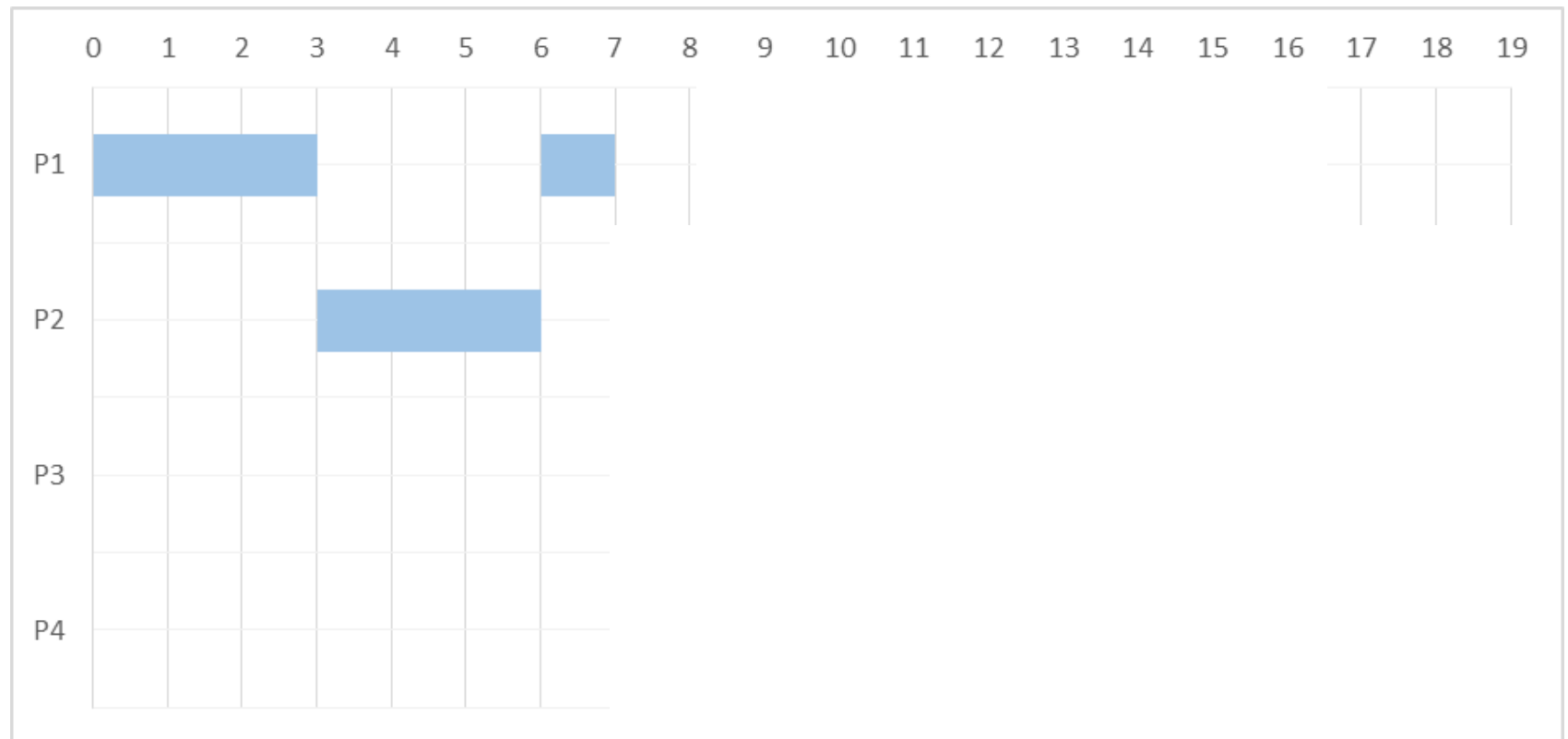
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



RR – Round Robin: grafico P1 cont.

- (Quanto di tempo = 3)
- (P1 e' interrotto)
- **P3** arriva a **T = 3** ma **P1** e' in coda e termina:

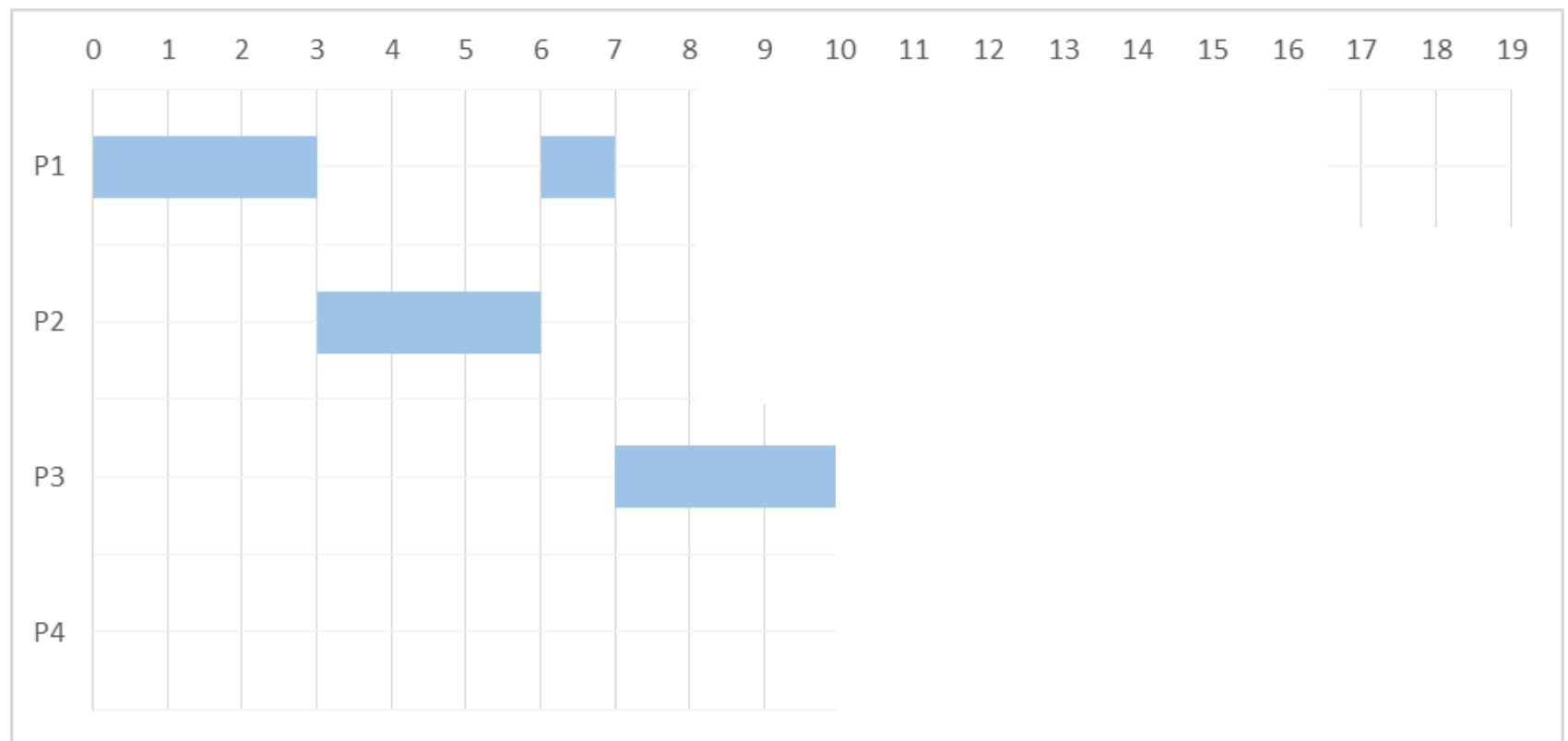
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



RR – Round Robin: grafico P3

- (Quanto di tempo = 3)
- (P1 e P2 terminati)
- **P3 inizia, ma si ferma a 3 su 6 QT**

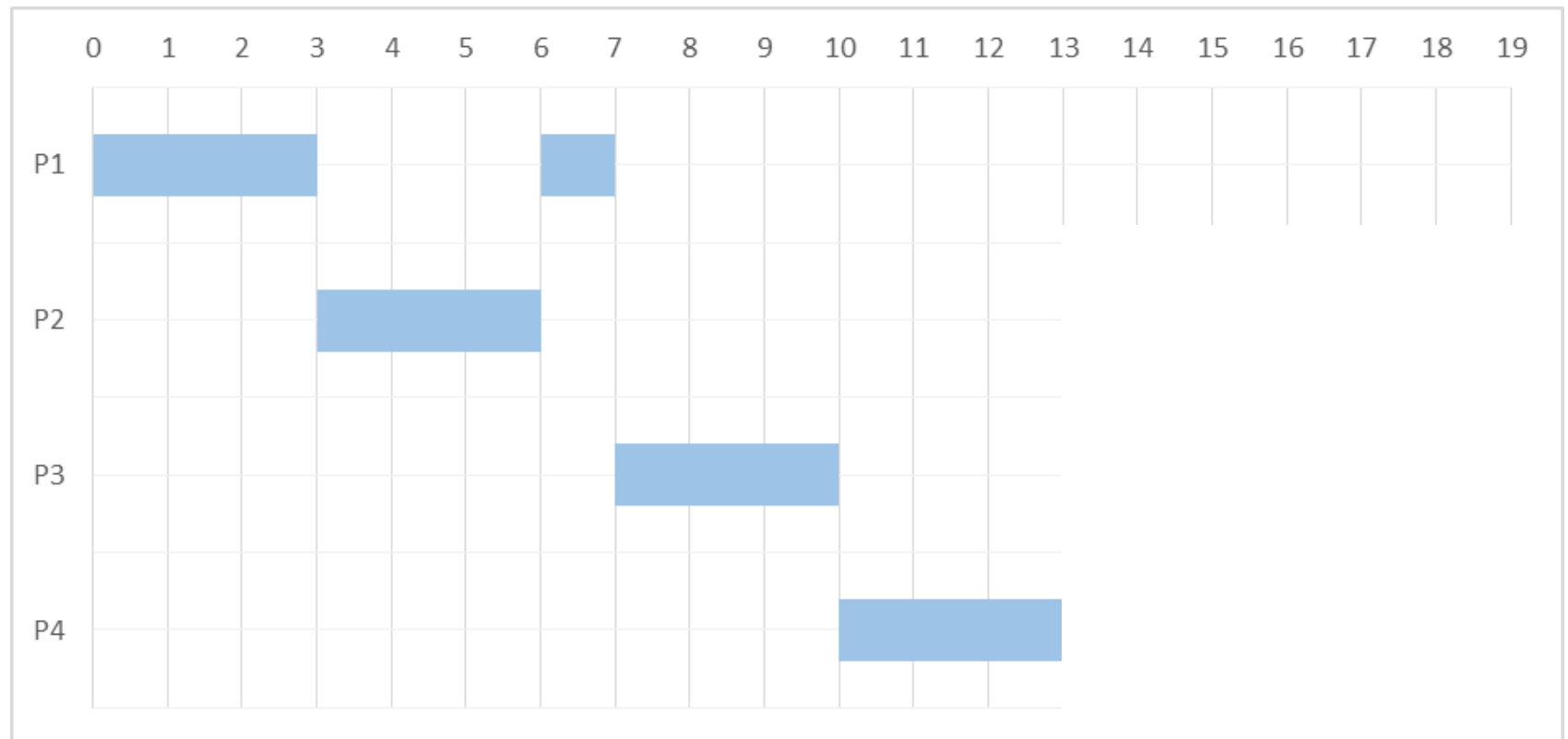
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



RR – Round Robin: grafico P4

- (Quanto di tempo = 3)
- (P3 a 3/6 QT)
- P4 inizia, ma si ferma a 3 su 5 QT

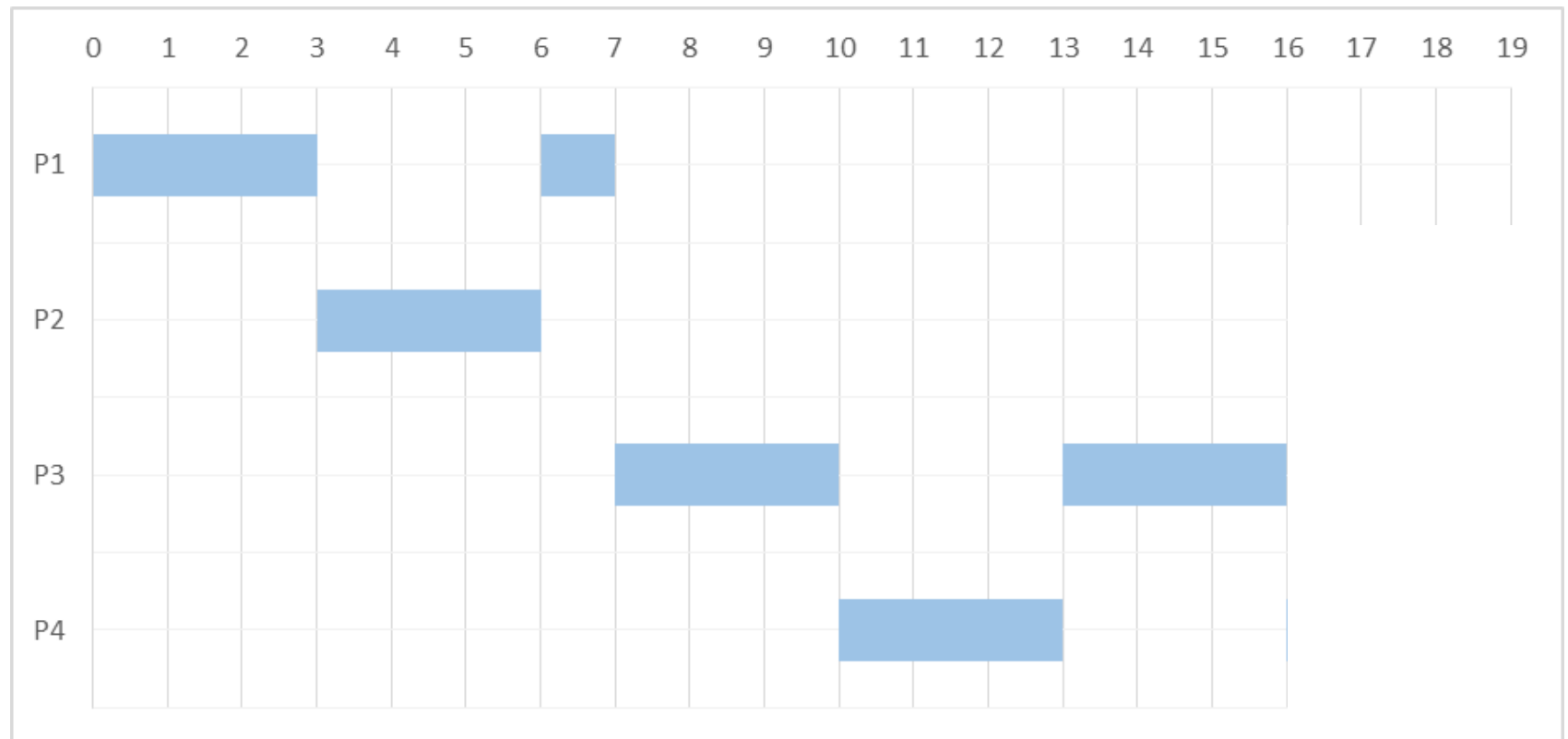
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



RR – Round Robin: grafico P3 riprende

- (Quanto di tempo = 3)
- P3 segue i 3 QT rimanenti (era attesa)

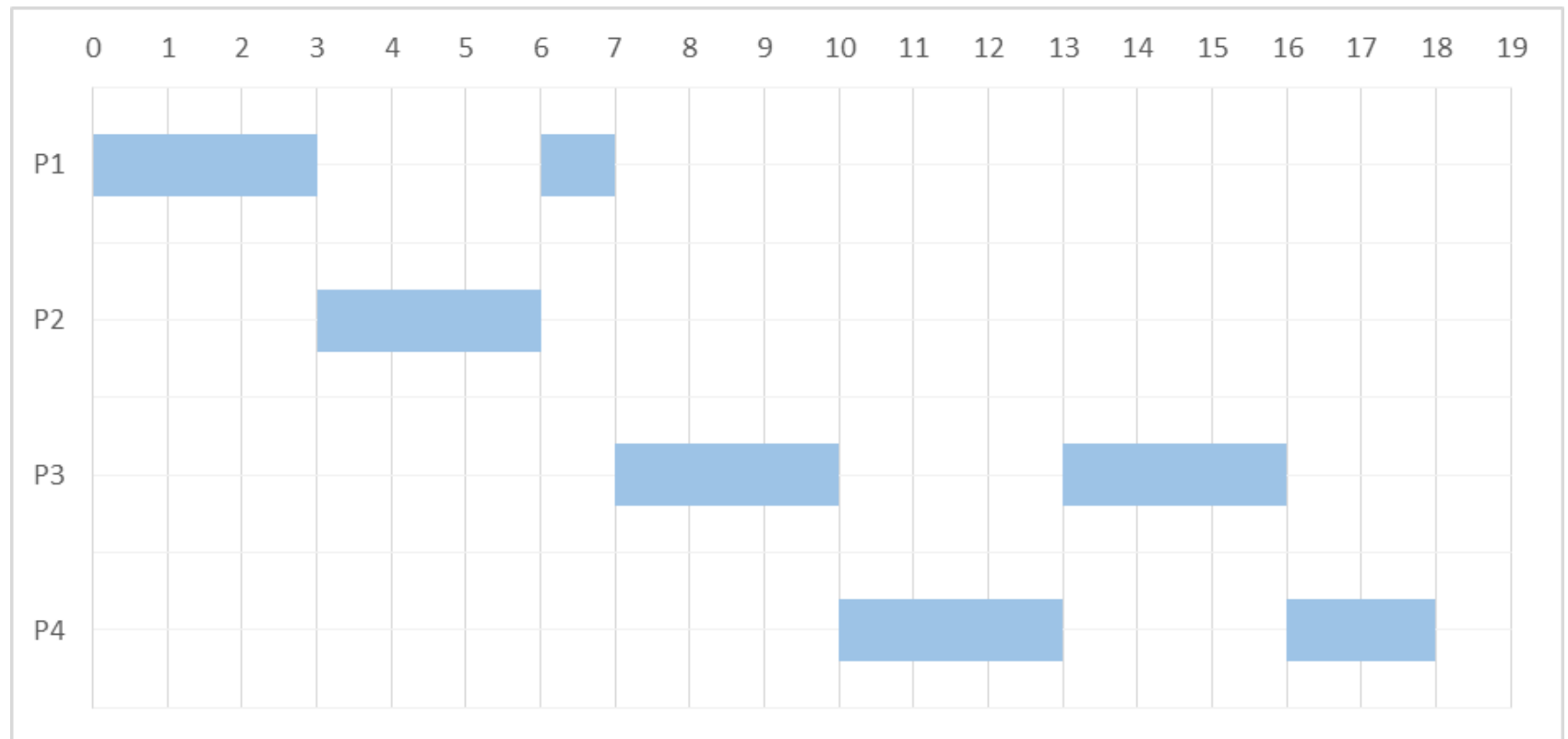
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



RR – Round Robin: grafico P4 riprende

- (Quanto di tempo = 3)
- P4 segue i 2 QT rimanenti

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

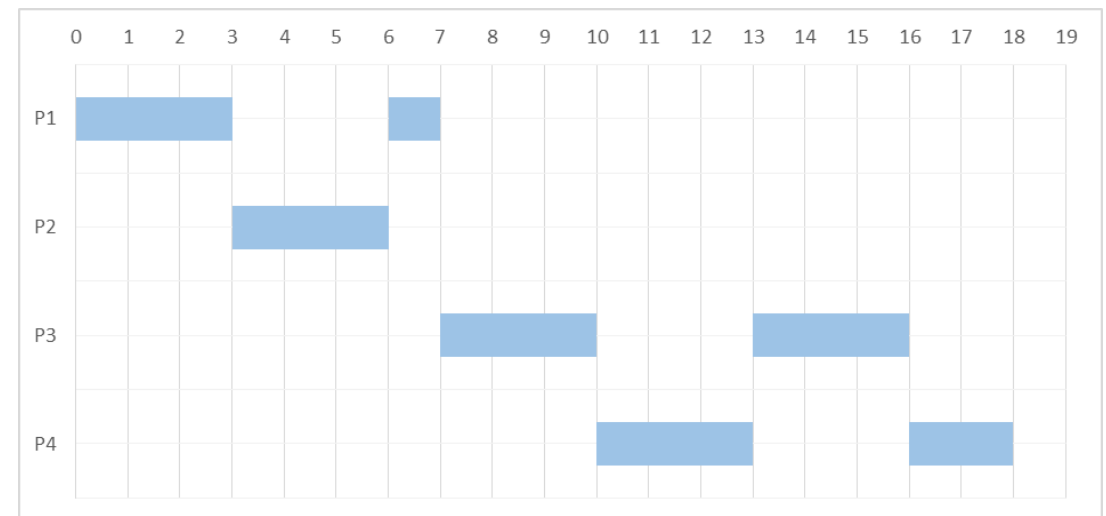


RR – Round Robin - Attesa P1

formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

– $T_{a P1} = 7 - 4 - 0 = 3$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

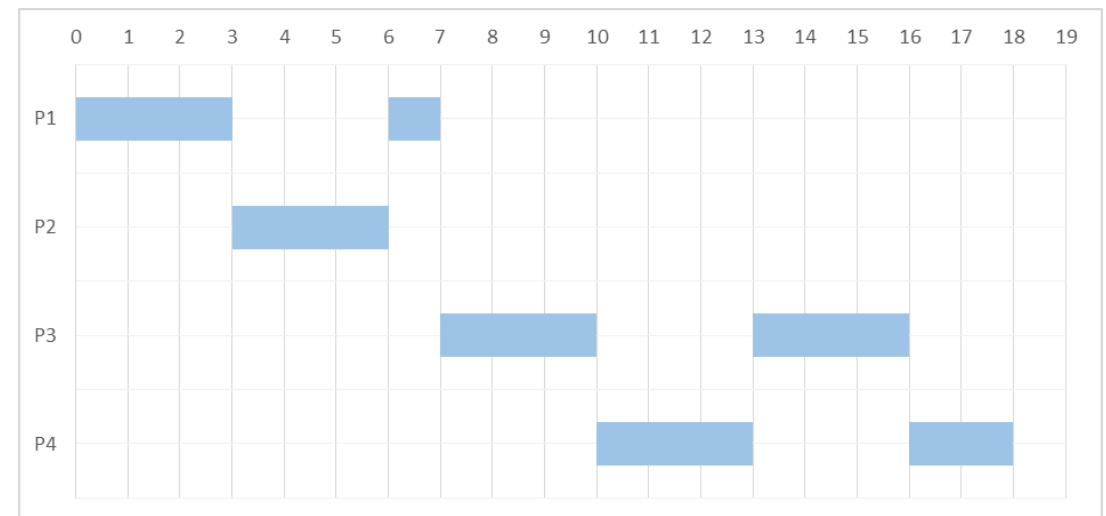


RR – Round Robin - Attesa P2

formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

- $T_{a P1} = 7 - 4 - 0 = 3$
- $T_{a P2} = 6 - 3 - 1 = 2$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

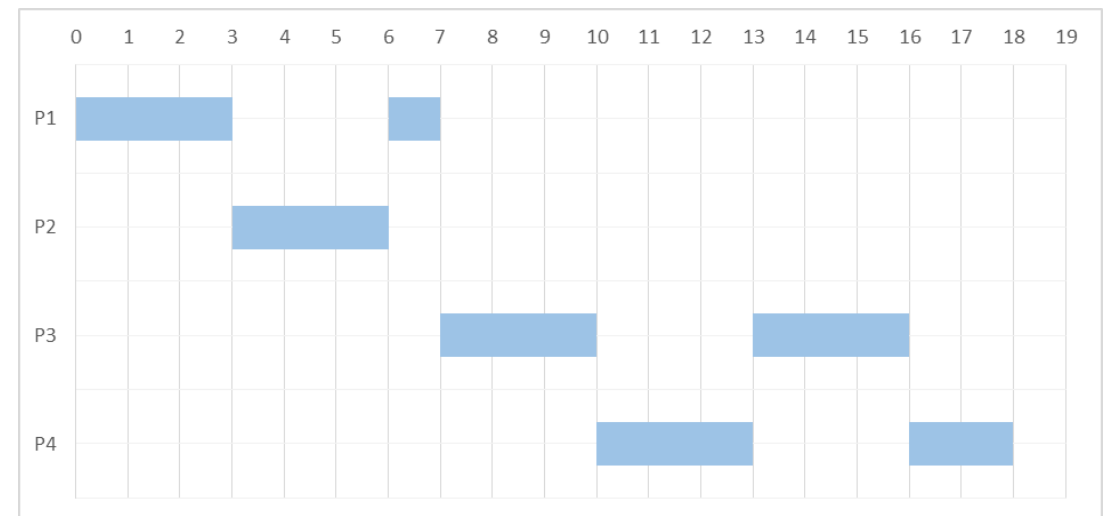


RR – Round Robin - Attesa P3

formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

- $T_{a P1} = 7 - 4 - 0 = 3$
- $T_{a P2} = 6 - 3 - 1 = 2$
- $T_{a P3} = 16 - 6 - 3 = 7$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

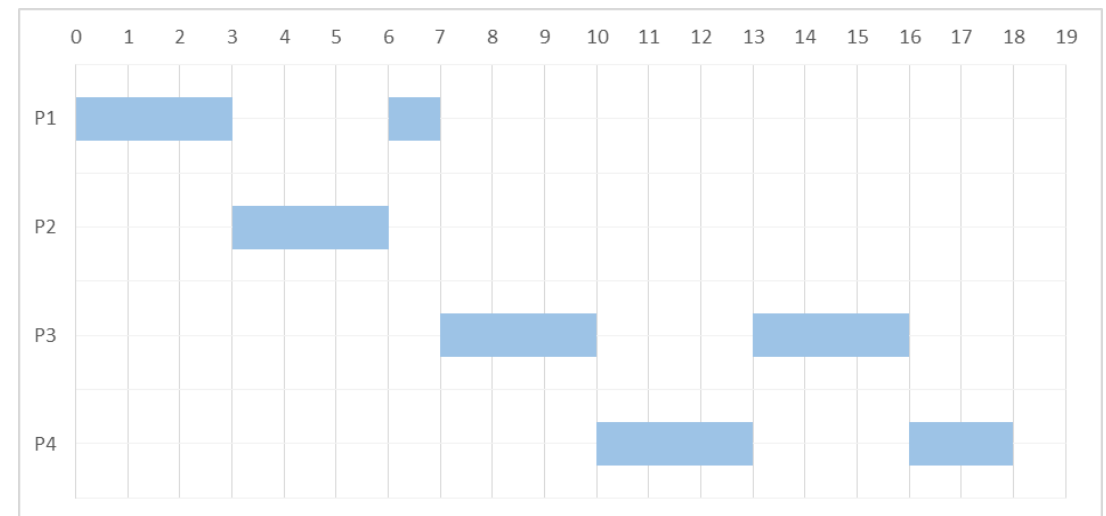


RR – Round Robin - Attesa P4

formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

- $T_{a P1} = 7 - 4 - 0 = 3$
- $T_{a P2} = 6 - 3 - 1 = 2$
- $T_{a P3} = 16 - 6 - 3 = 7$
- $T_{a P4} = 18 - 5 - 5 = 8$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



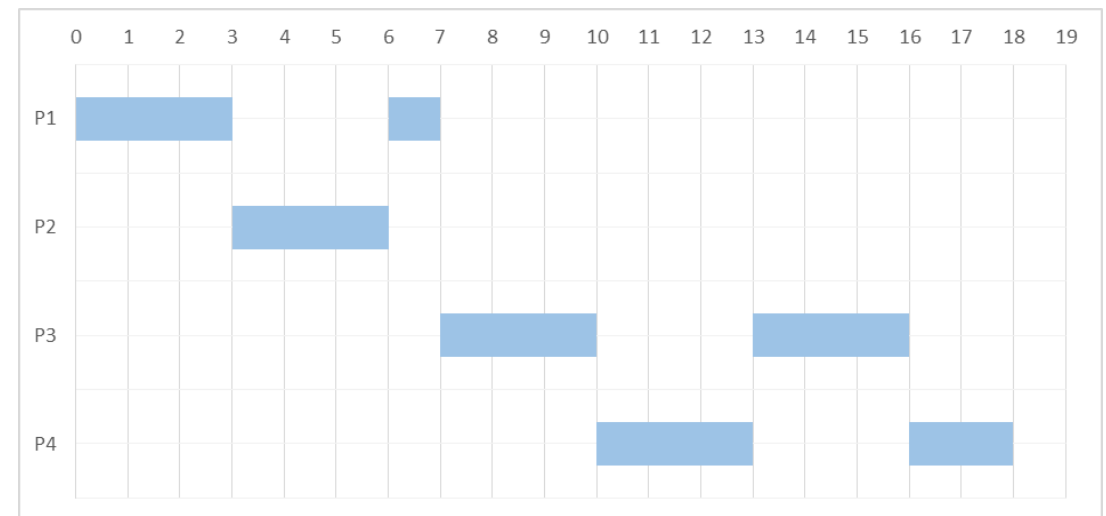
RR – Round Robin - Attesa media

formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

- $T_{a P1} = 7 - 4 - 0 = 3$
- $T_{a P2} = 6 - 3 - 1 = 2$
- $T_{a P3} = 16 - 6 - 3 = 7$
- $T_{a P4} = 18 - 5 - 5 = 8$

- $\rightarrow \text{Attesa media} = (3 + 2 + 7 + 8) / 4 = 20 / 4 = 5$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

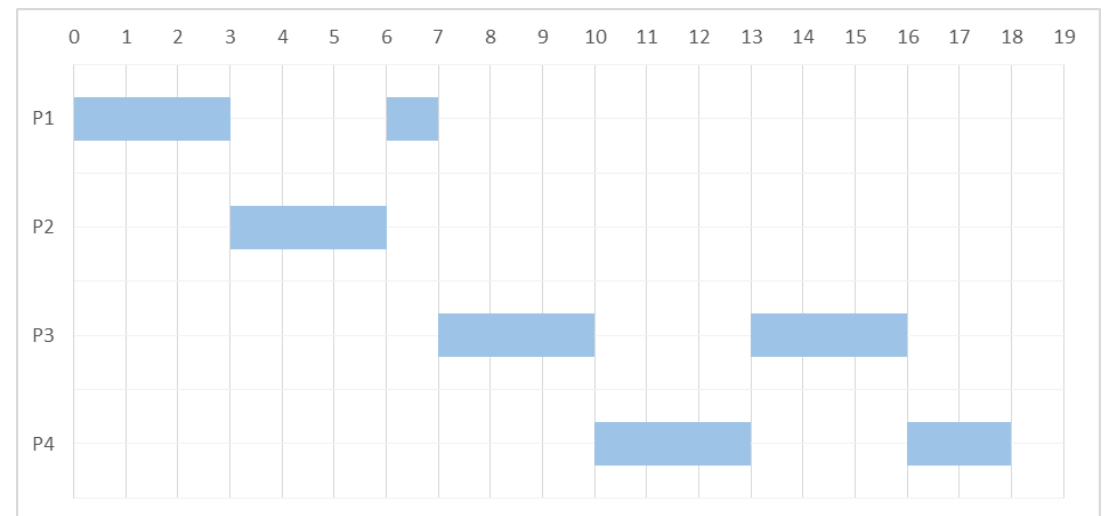


RR – Round Robin - Prestazioni P1

formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

– $R_{a P1} = 4 / (7-0) = 0.57$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

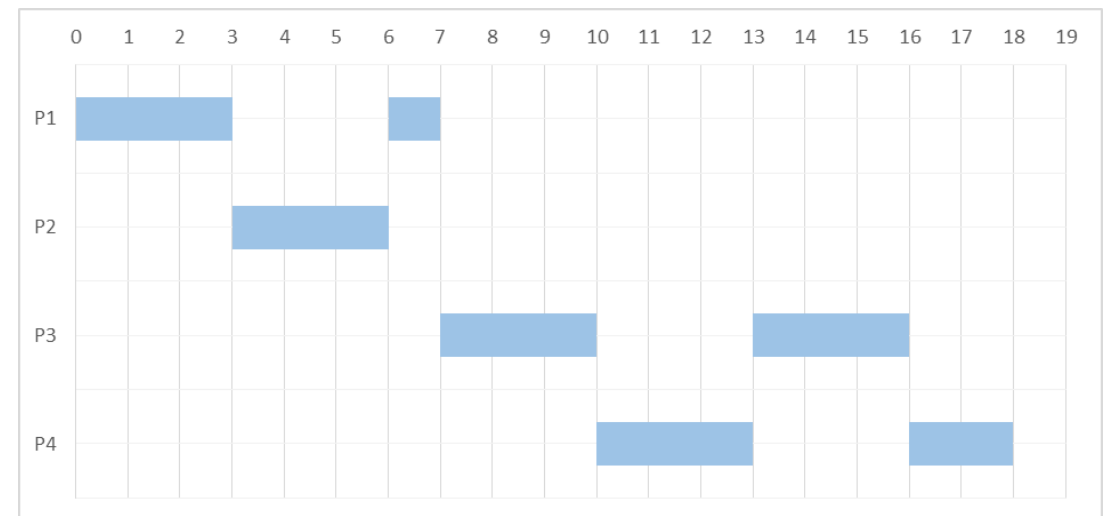


RR – Round Robin - Prestazioni P2

formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

- $R_{a\ P1} = 4 / (7-0) = 0.57$
- $R_{a\ P2} = 3 / (6-1) = 0.6$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

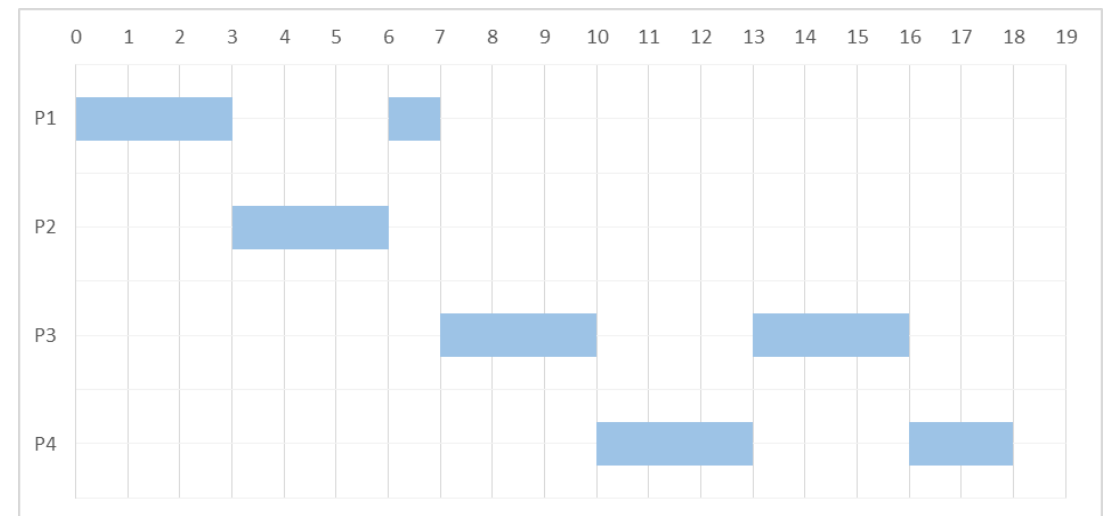


RR – Round Robin - Prestazioni P3

formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

- $R_{a\ P1} = 4 / (7-0) = 0.57$
- $R_{a\ P2} = 3 / (6-1) = 0.6$
- $R_{a\ P3} = 6 / (16-3) = 0.4615$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5



RR – Round Robin - Prestazioni P4

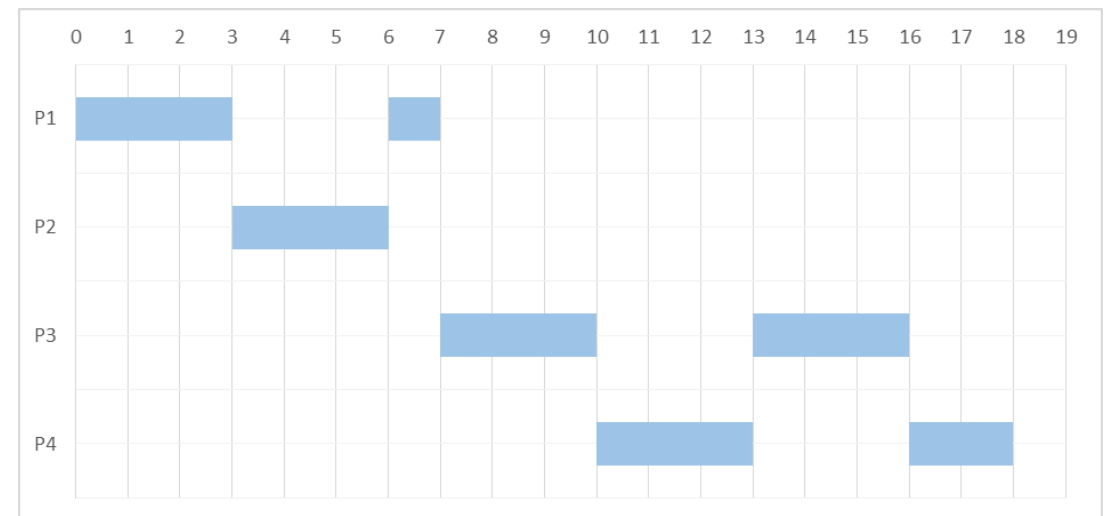
formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

- $R_{a P1} = 4 / (7-0) = 0.57$
- $R_{a P2} = 3 / (6-1) = 0.6$
- $R_{a P3} = 6 / (16-3) = 0.4615$
- $R_{a P4} = 5 / (18-5) = 0.384$

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

→ Prestazioni medie =

$$(0,5+0,6+0,46+0,38)/4 \sim 2/4 = 0.50$$



SRR – Selfish Round Robin

- Increases priority as process ages
- Two queues:
 - Active (coda ei processi attivi)
 - Holding (coda di Attesa)
- Favours older processes to avoid unreasonable delays
(Aumenta la priorità dei P più vecchi)

SRR – Selfish Round Robin

- Dati i seguenti processi con i rispettivi tempo di arrivo, tempo di servizio

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

- Quanto di tempo = 3
 - Disegnare il grafico temporale dello scheduling
 - Calcolare il tempo medio di attesa
 - Calcolare il rapporto di prestazioni

SRR – Selfish Round Robin: grafico

- **NOTA:**
- Quanto di tempo = 3 uguale a RR
- Stessi Tempi di RR

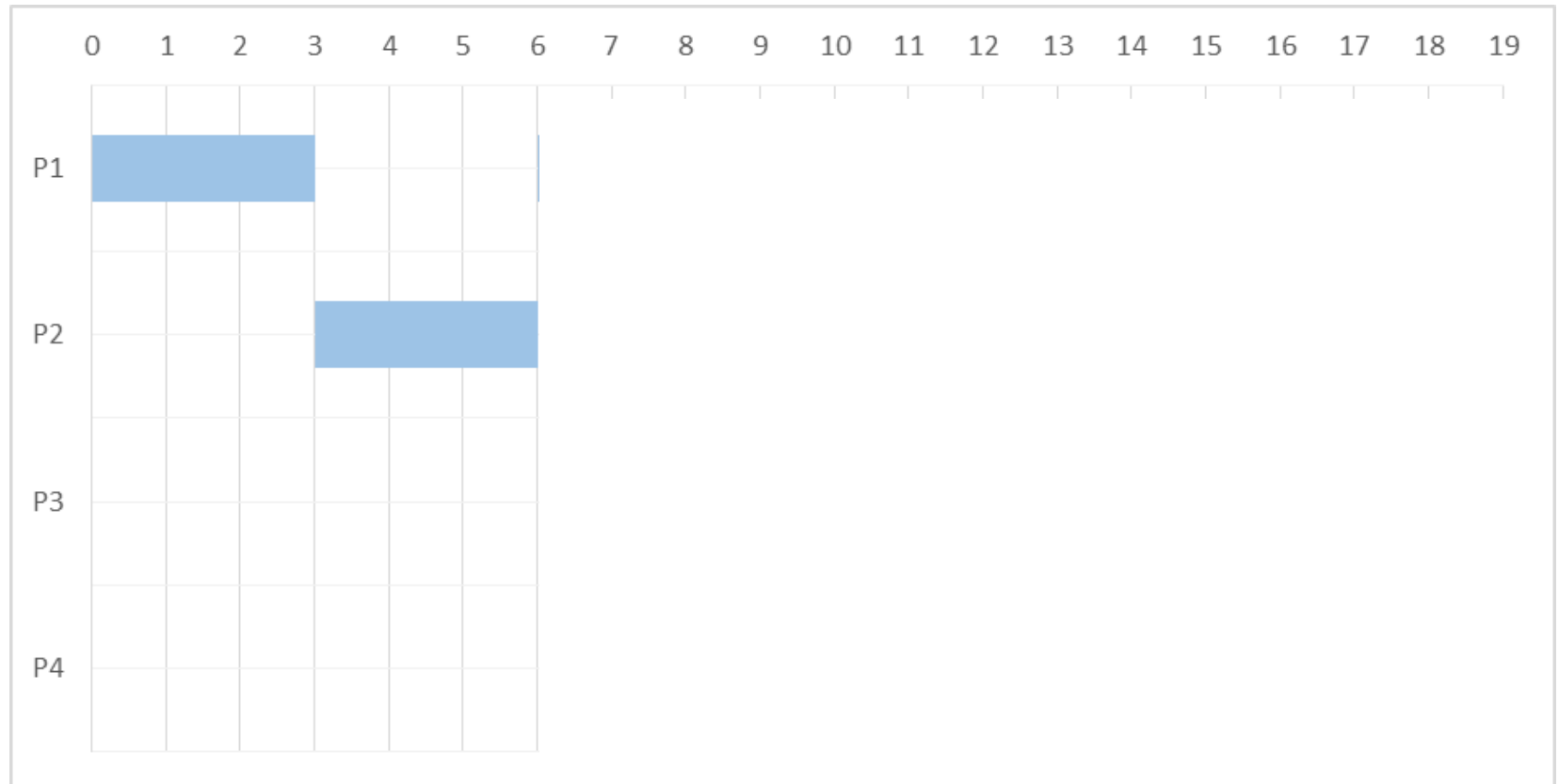
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

- Cosa puo' cambiare?

- ...

SRR – Selfish Round Robin: grafico

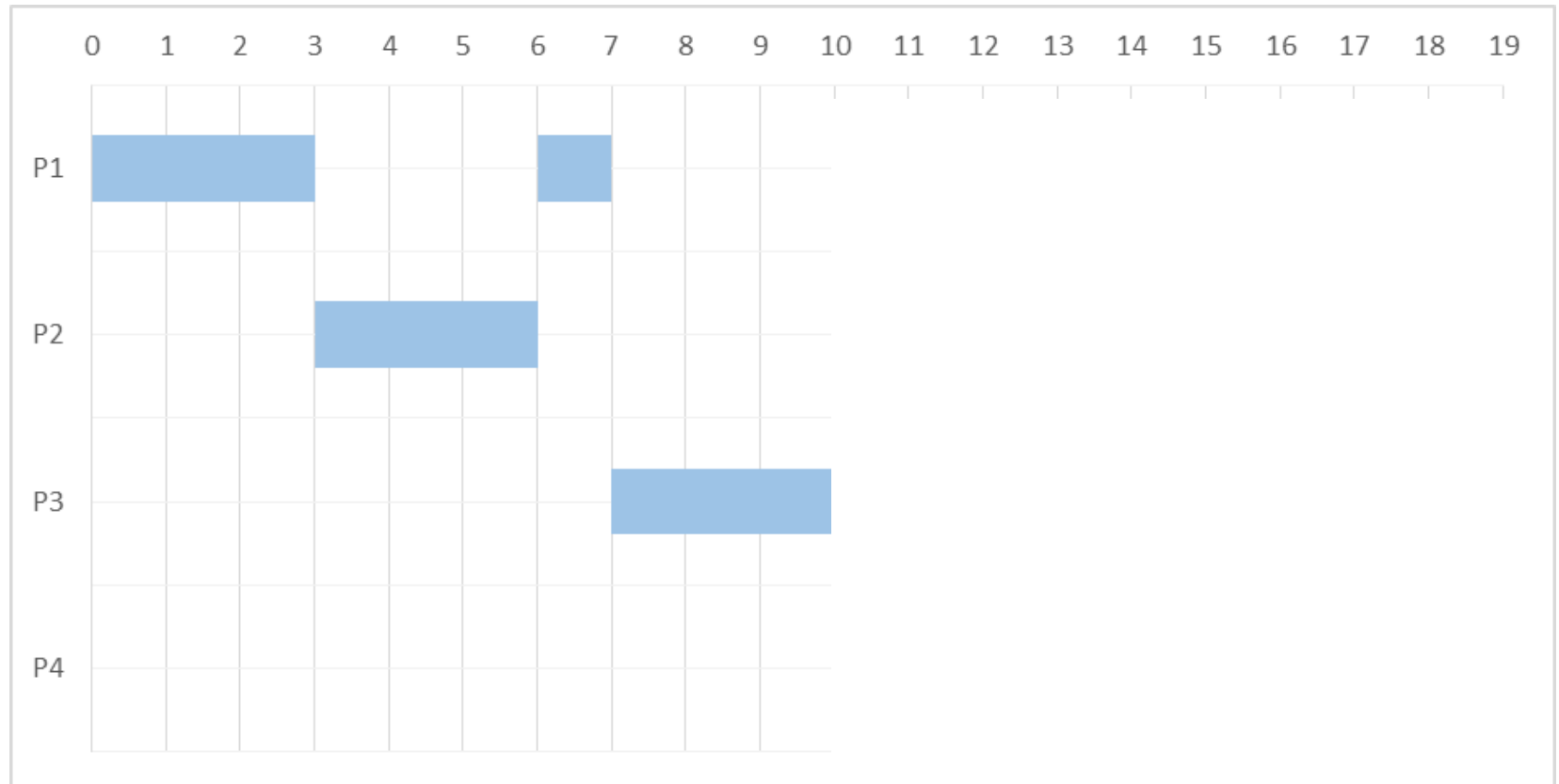
- Quanto di tempo = 3



- Come x RR parte P1, si interrompe dopo 3 QT, parte P2..

SRR – Selfish Round Robin: grafico

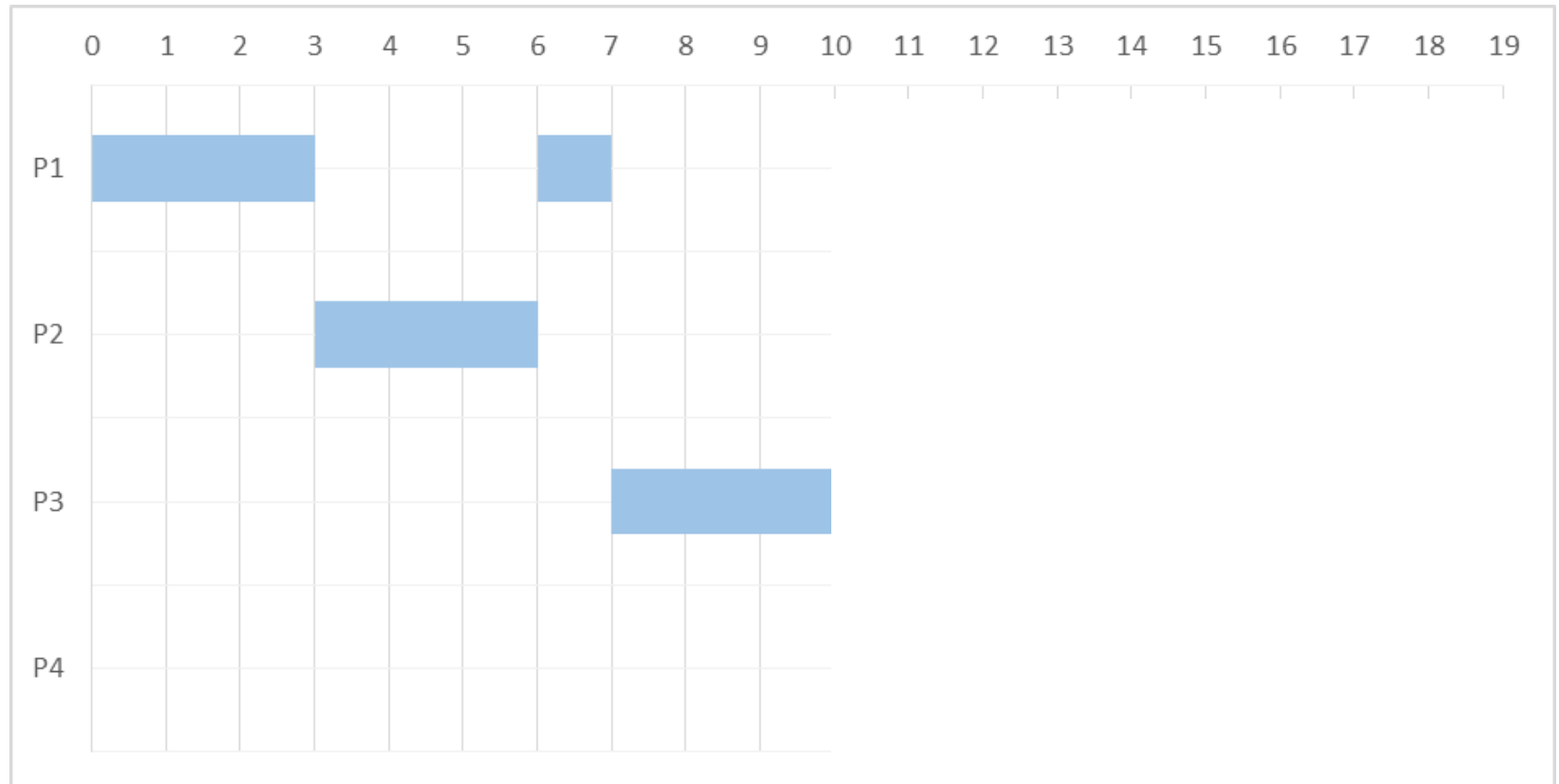
- Quanto di tempo = 3



- Come x RR parte P1, si interrompe dopo 3 QT, parte P2..
- Riparte P1, e parte P3: abbiamo Processi in Aging?

SRR – Selfish Round Robin: grafico

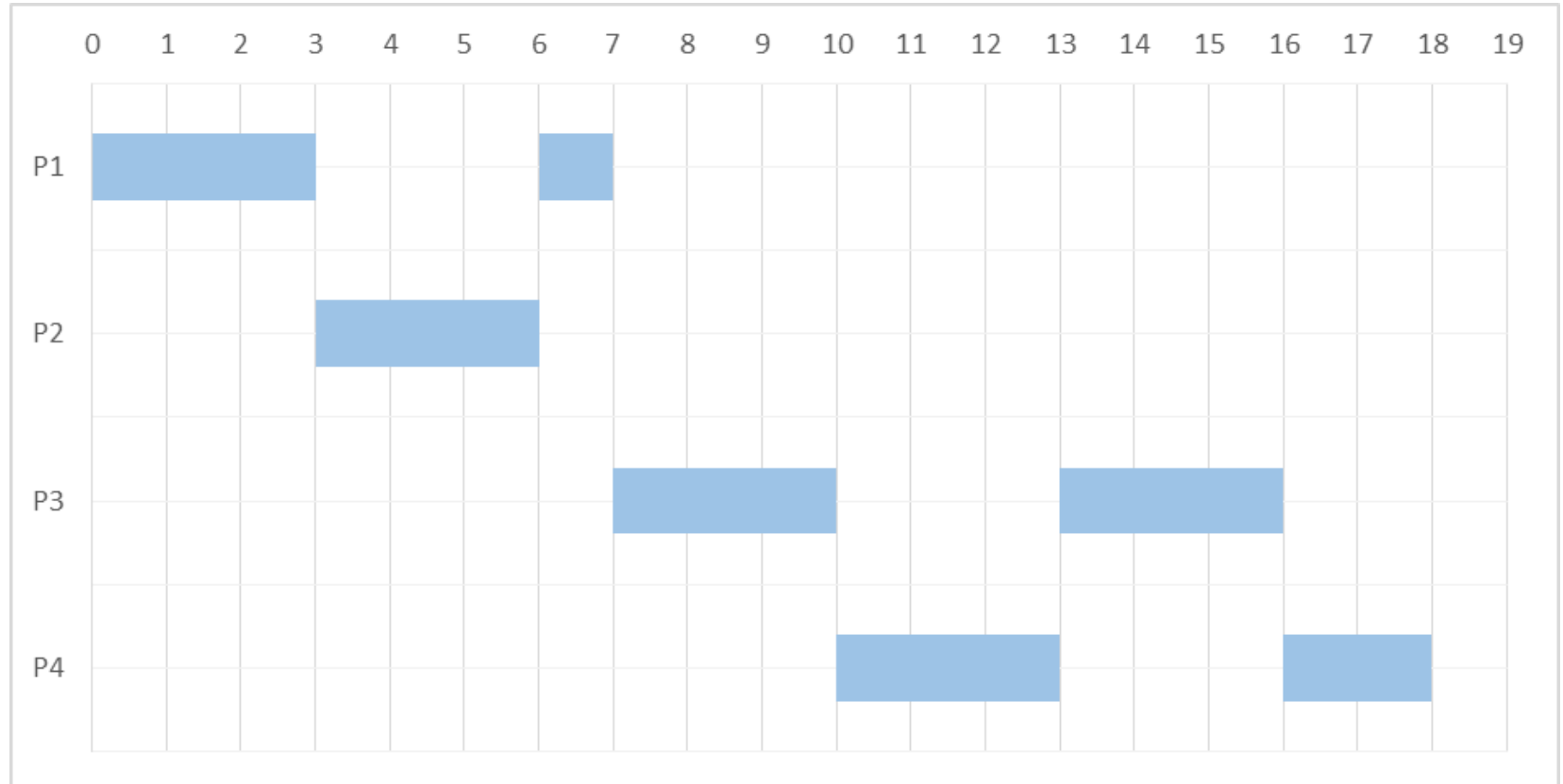
- Quanto di tempo = 3



- Riparte P1, e parte P3: abbiamo Processi in Aging?
- A $T=7$ No
- A $T=10$ P3 finisce, quindi $N \text{ Processi} = 0$

SRR – Selfish Round Robin: grafico

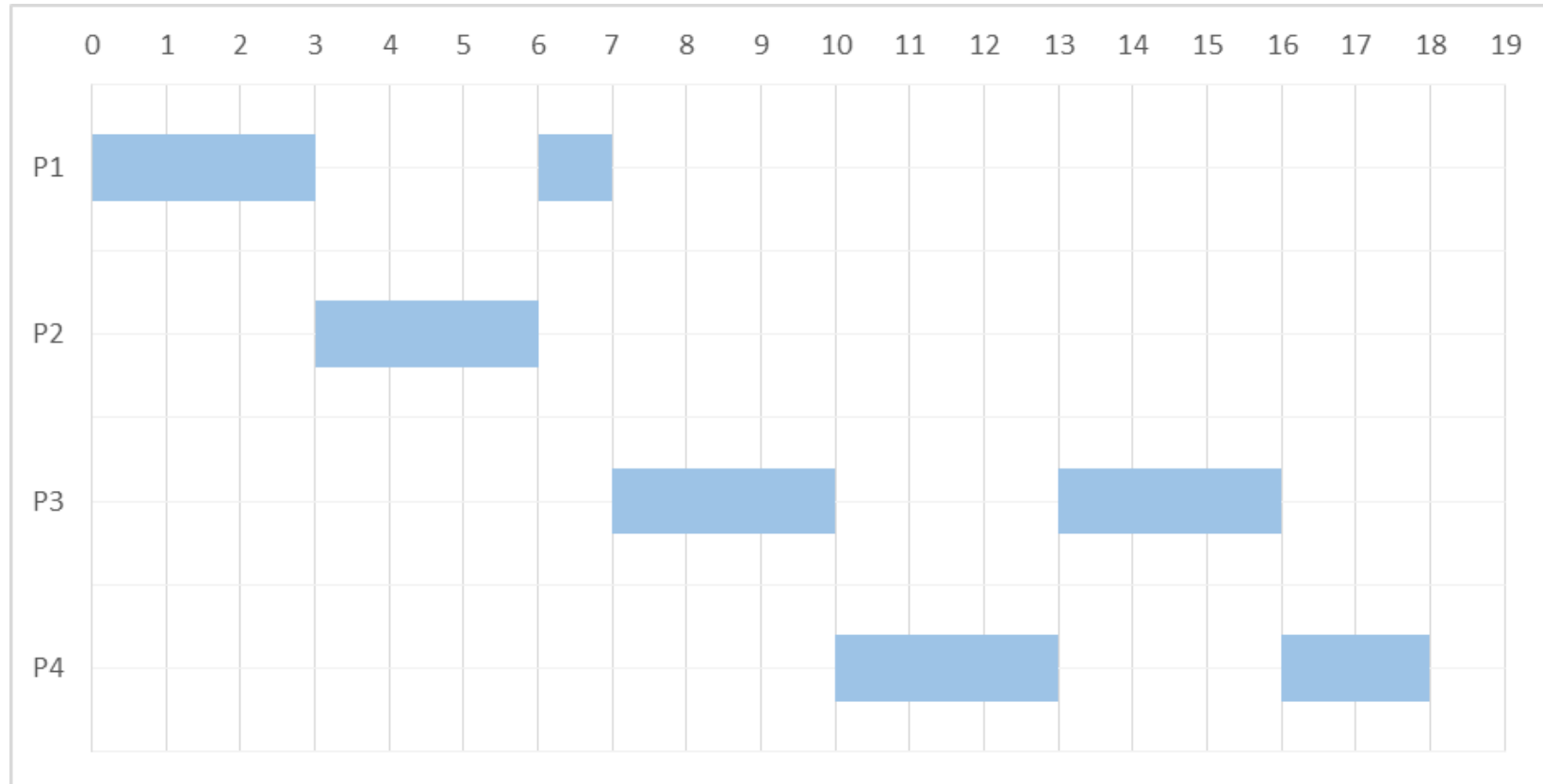
- Quanto di tempo = 3



- Quando parte P4 ci sarebbe in Aging P3, ma a $T = 13$ "tocca" già a P3 che completa....

SRR – Selfish Round Robin

- Lo scheduling risultante è == RR!



- **Attesa**

- $P1 = 3, P2 = 2, P3 = 7, P4 = 8$

→ Attesa media = 5

- **Prestazioni**

- $P1 = 0.57, P2 = 0.60, P3 = 0.46, P4 = 0.38$

→ Prestazioni medie = 0.50

(N)PSJF – (Non) Preemptive Shortest Job First

■ Def:

Tipi d Burst:

- CPU burst: μp is performing calculations
- I/O burst, μp is waiting for data transfer in or out of the system.

Ci interessa:

CPU burst: the amount of time the process uses the processor before it is no longer ready

(N)PSJF – (Non) Preemptive Shortest Job First

■ Funzionamento:

- assumiamo sia NOTO il tempo di Burst x ogni processo

(- la lunghezza del burst e' il tempo x cui continuerebbe ad eseguire se NON "preempted")

L' algoritmo STIMA il tempo

All' Inizio valore di "default"

Negli es. Burst Time verra' dato x ipotesi

NPSJF – Non Preemptive Shortest Job First

- Dati i seguenti processi con i rispettivi tempo di arrivo, tempo di burst

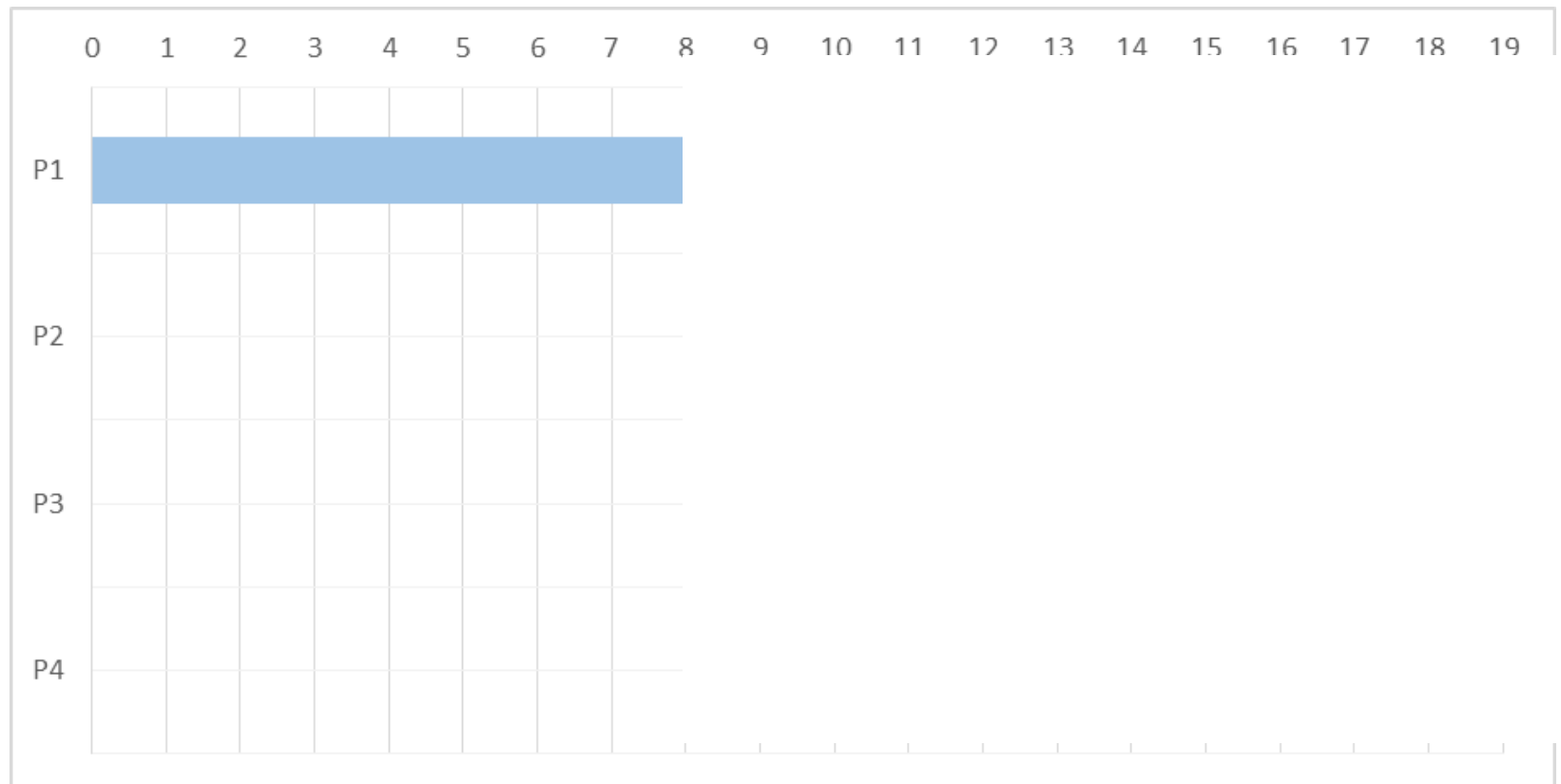
Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1

- Disegnare il grafico temporale dello scheduling
- Calcolare il tempo medio di attesa
- Calcolare il rapporto di prestazioni

NPSJF – Non Preemptive Shortest Job First: GRAFICO

- Pe def. "NonPreemptive":
- Parte P1 NON puo' essere interrotto:

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1

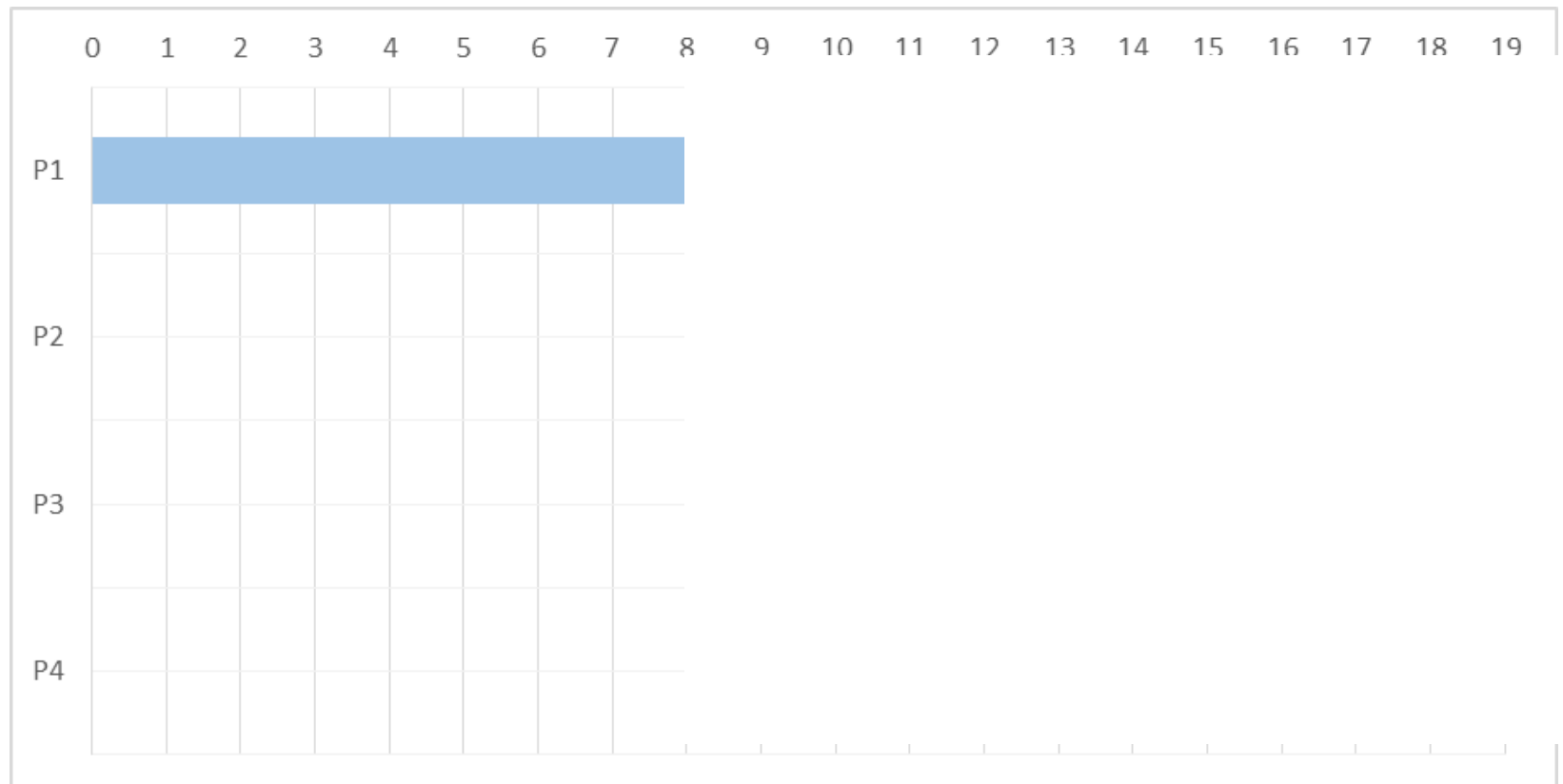


NPSJF – Non Preemptive Shortest Job First: GRAFICO

- P2 termina:
- Al $T = 8$ sono GIA arrivati sia P2 che P3 che P4
- (tutti sarebbero schedulabili..)
- **Chi ha Burst Minore?**

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1

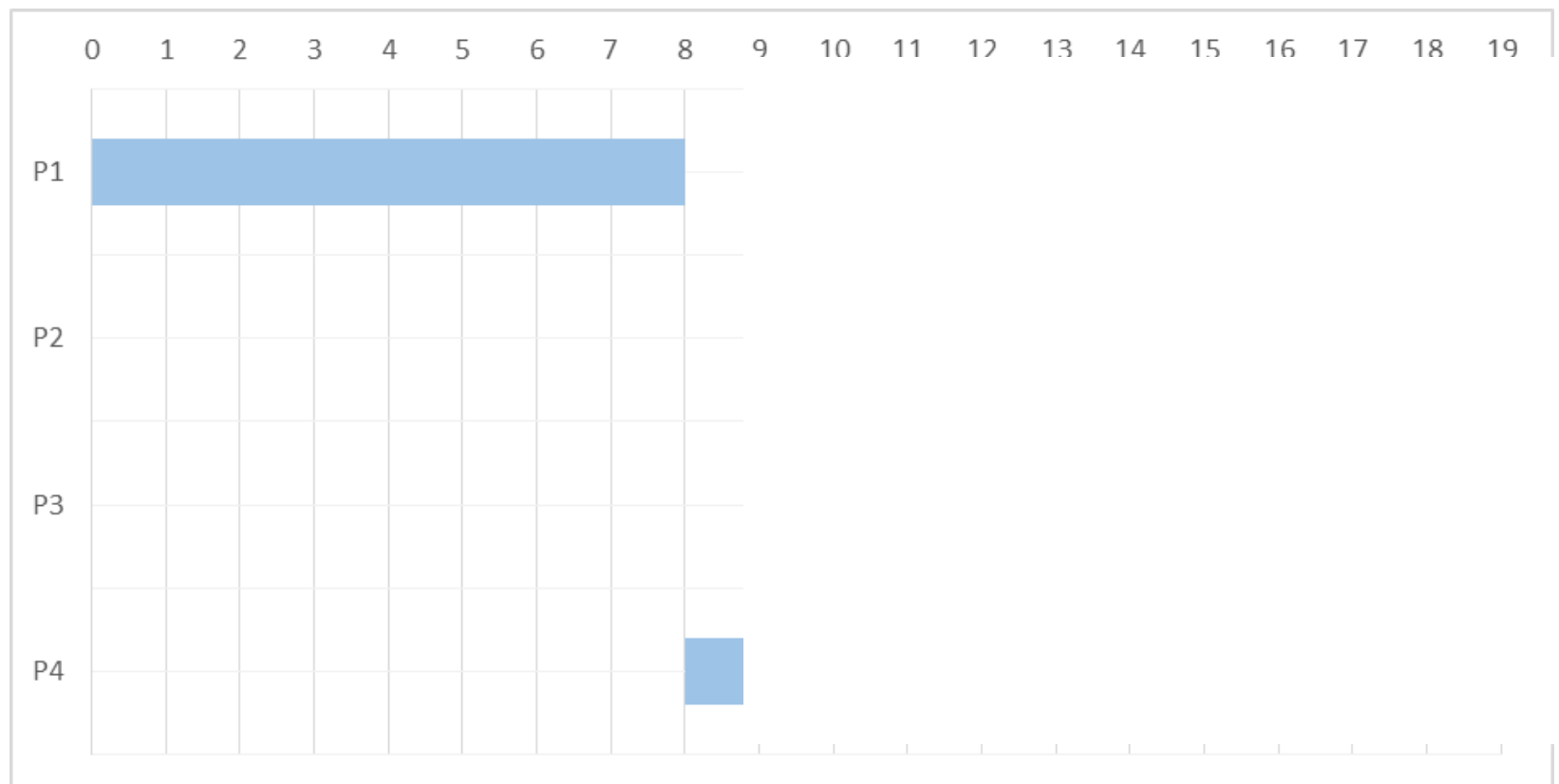
– ...



NPSJF – Non Preemptive Shortest Job First: GRAFICO

- P2 termina:
- Al $T = 8$ sono GIA arrivati sia P2 che P3 che P4
- Chi ha Burst Minore? **P4 (1T)**

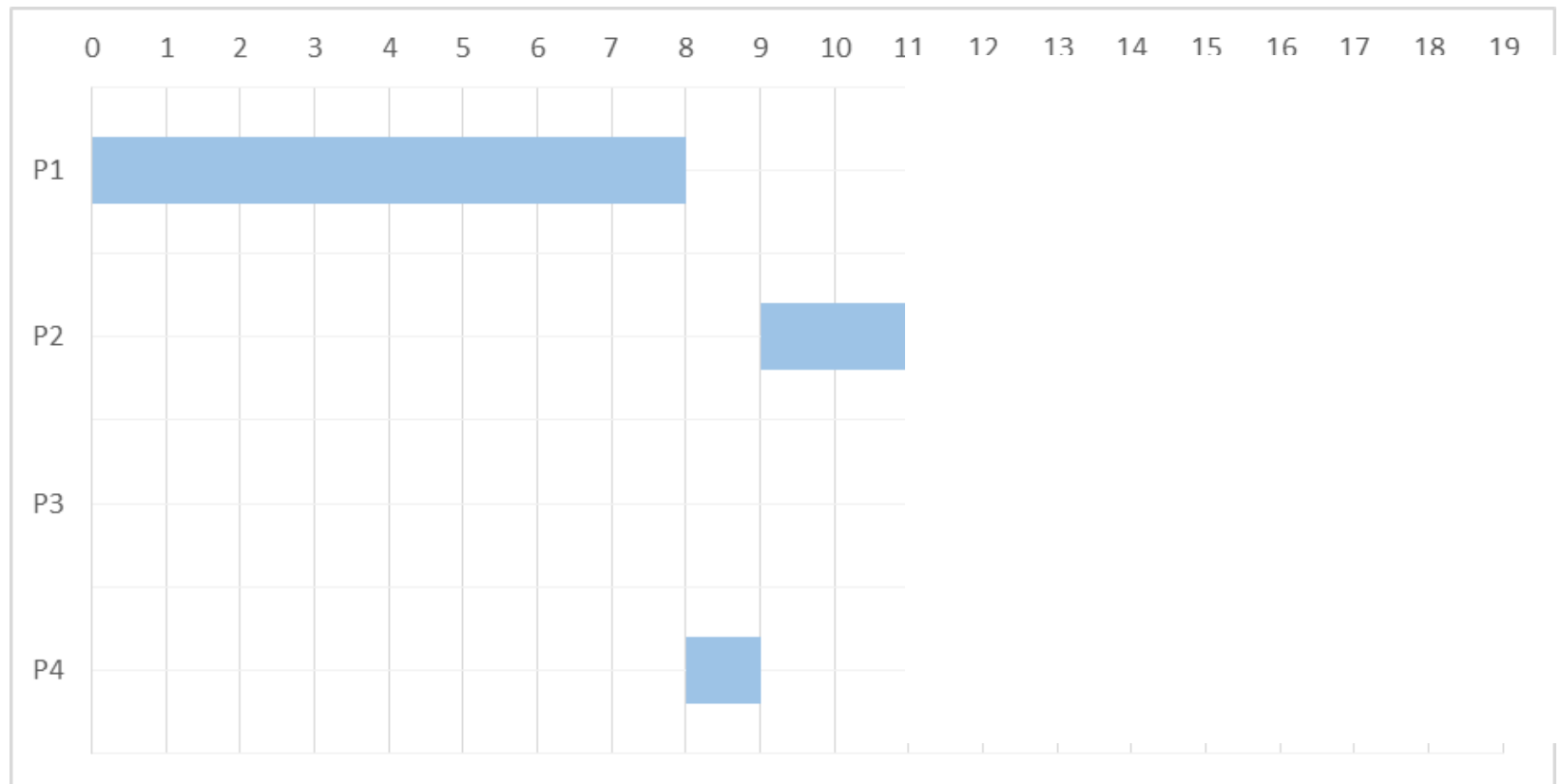
Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1



NPSJF – Non Preemptive Shortest Job First: GRAFICO

- P4 termina:
- Al $T = 9$ sono GIA arrivati sia P2 che P3
- Chi ha Burst Minore? **P2 (2T)**

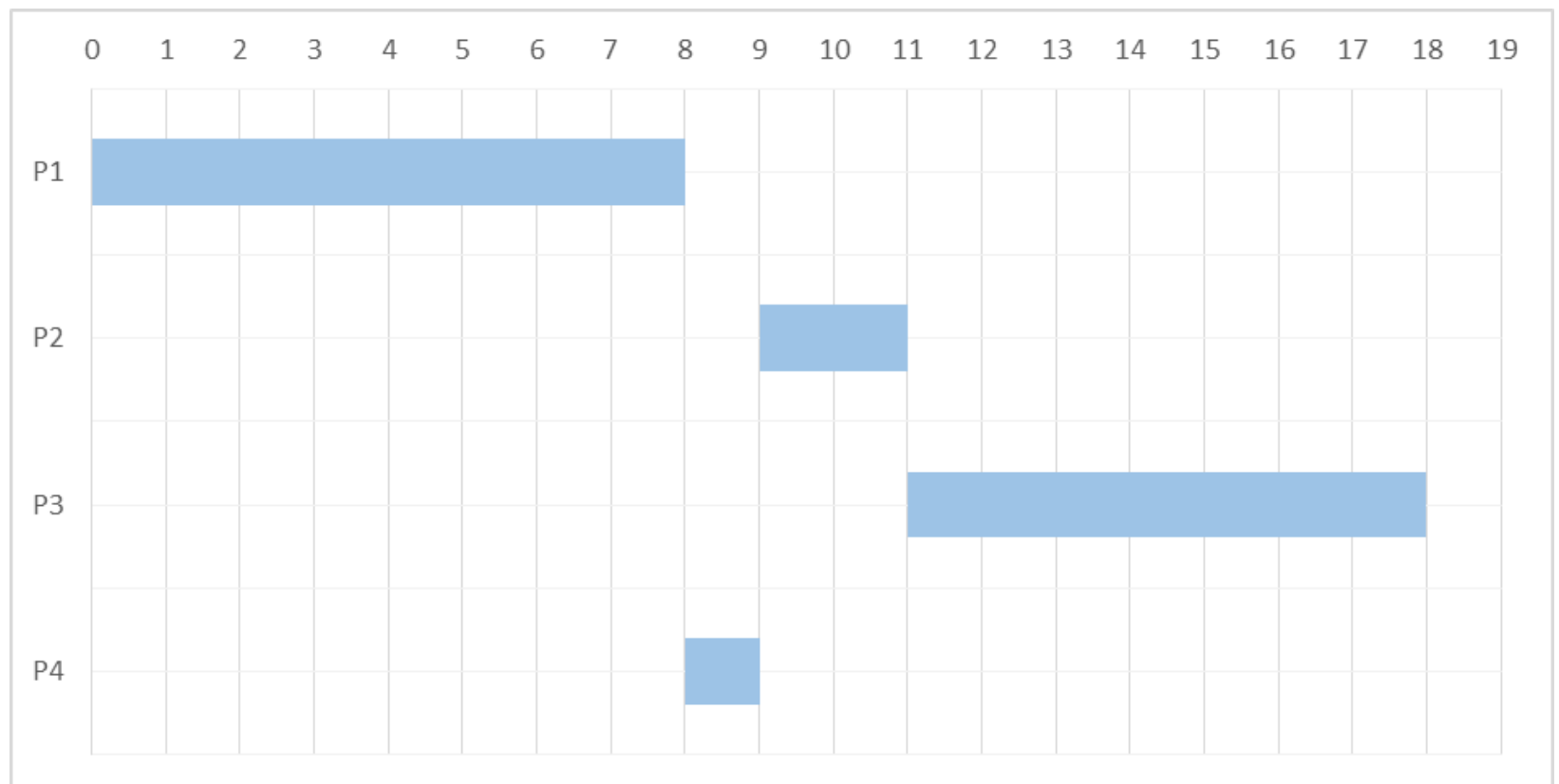
Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1



NPSJF – Non Preemptive Shortest Job First: GRAFICO

– Ultimo in coda **P3**

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1



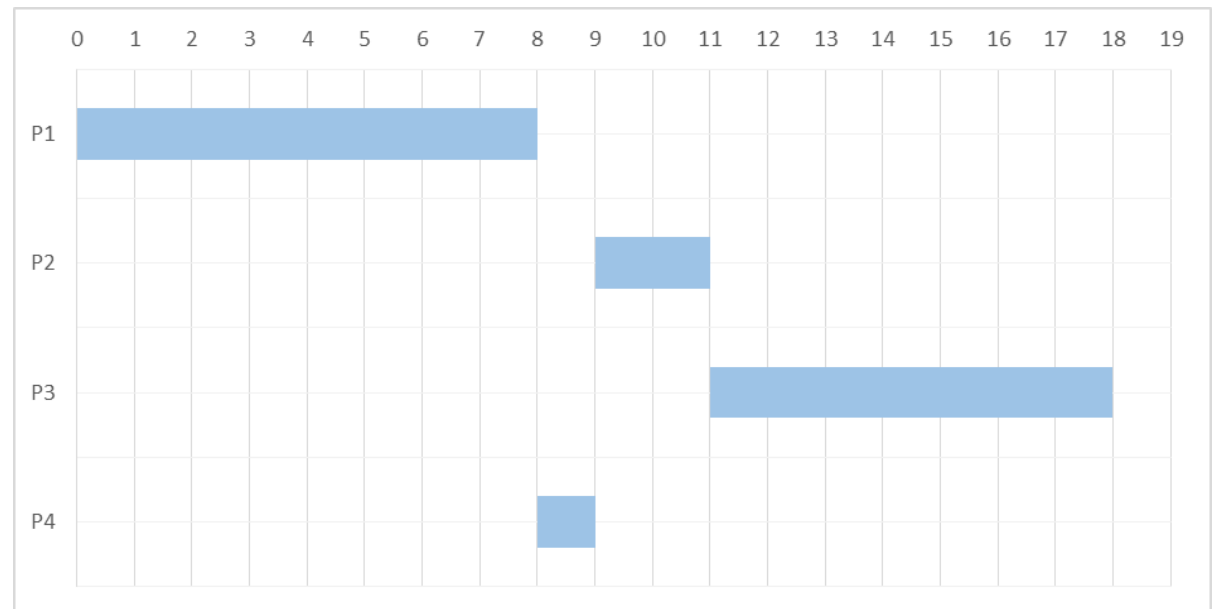
NPSJF – T Attesa

formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

– (In questo caso : $T_{servizio} = T_{burst}$)

– $T_{a P1} = 8 - 8 - 0 = 0$

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1



NPSJF – T Attesa

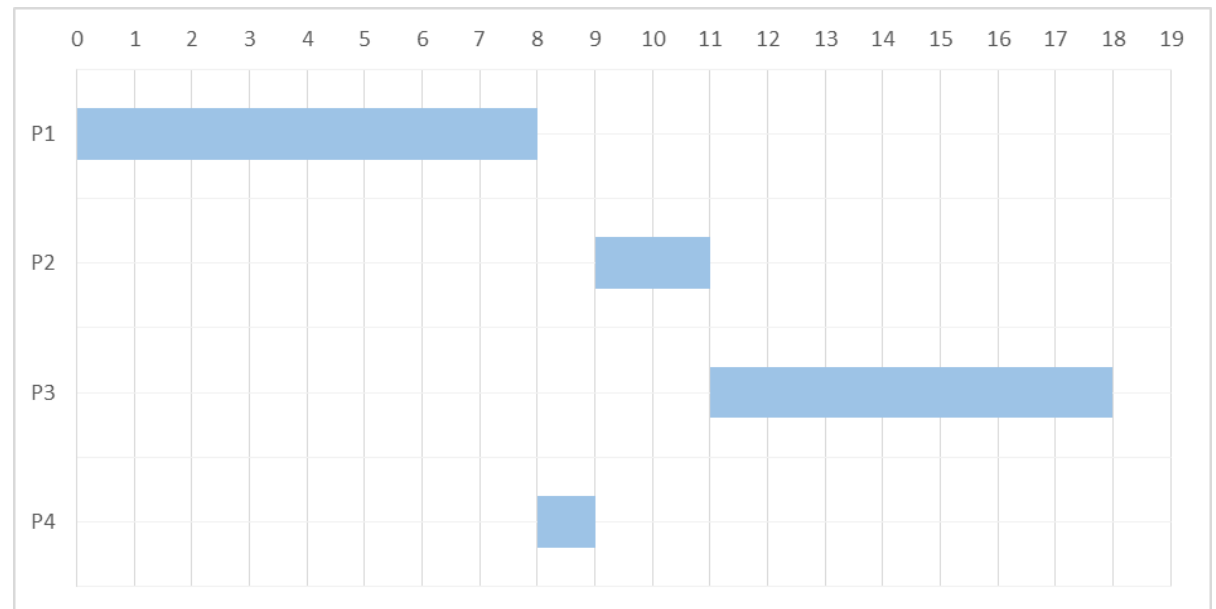
formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

– (In questo caso : $T_{servizio} = T_{burst}$)

– $T_{a P1} = 8 - 8 - 0 = 0$

– $T_{a P2} = 11 - 2 - 1 = 8$

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1



NPSJF – T Attesa

formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

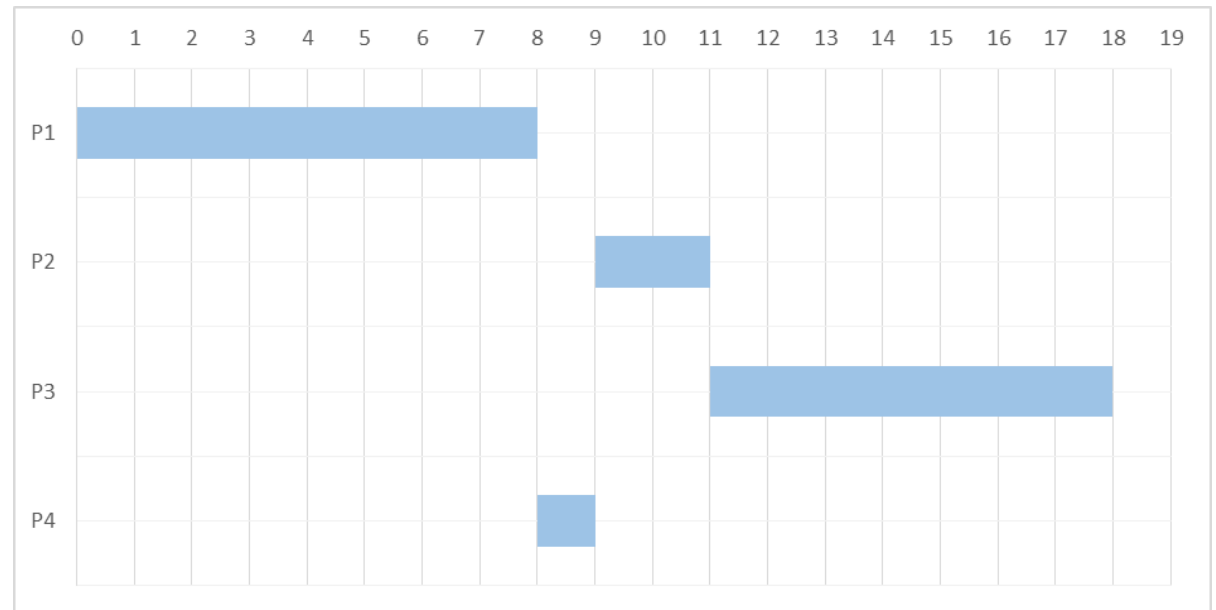
– (In questo caso : $T_{servizio} = T_{burst}$)

– $T_{a P1} = 8 - 8 - 0 = 0$

– $T_{a P2} = 11 - 2 - 1 = 8$

– $T_{a P3} = 18 - 7 - 3 = 8$

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1



NPSJF – T Attesa

formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

– (In questo caso : $T_{servizio} = T_{burst}$)

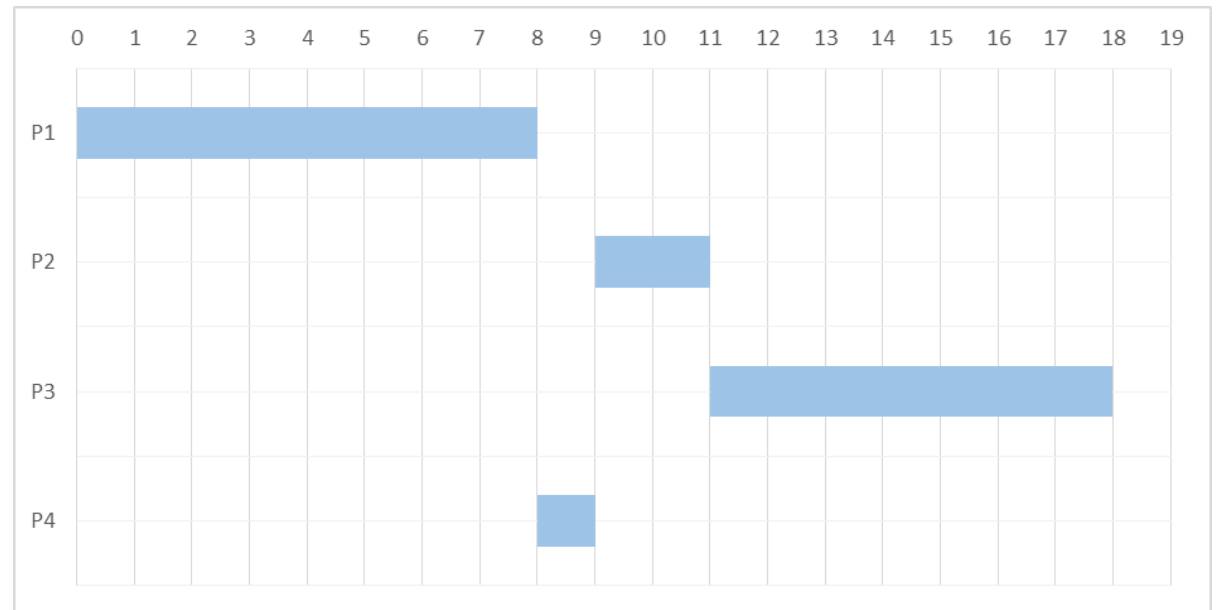
– $T_{a P1} = 8 - 8 - 0 = 0$

– $T_{a P2} = 11 - 2 - 1 = 8$

– $T_{a P3} = 18 - 7 - 3 = 8$

– $T_{a P4} = 9 - 1 - 5 = 3$

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1



NPSJF – T Attesa

formula: $T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVO}$

– (In questo caso : $T_{servizio} = T_{burst}$)

– $T_{a P1} = 8 - 8 - 0 = 0$

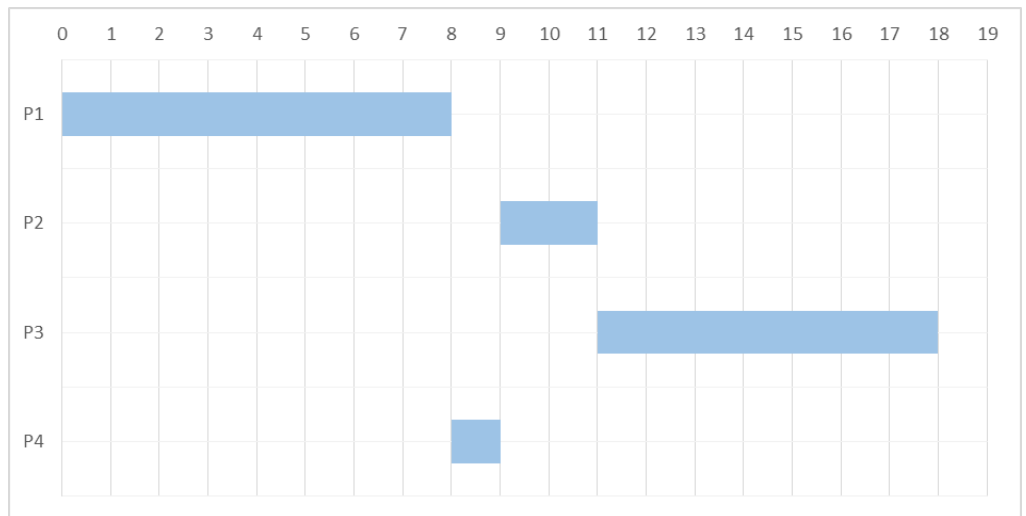
– $T_{a P2} = 11 - 2 - 1 = 8$

– $T_{a P3} = 18 - 7 - 3 = 8$

– $T_{a P4} = 9 - 1 - 5 = 3$

– **Attesa media = $(0+8+8+3)/4 = 4.75$**

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1



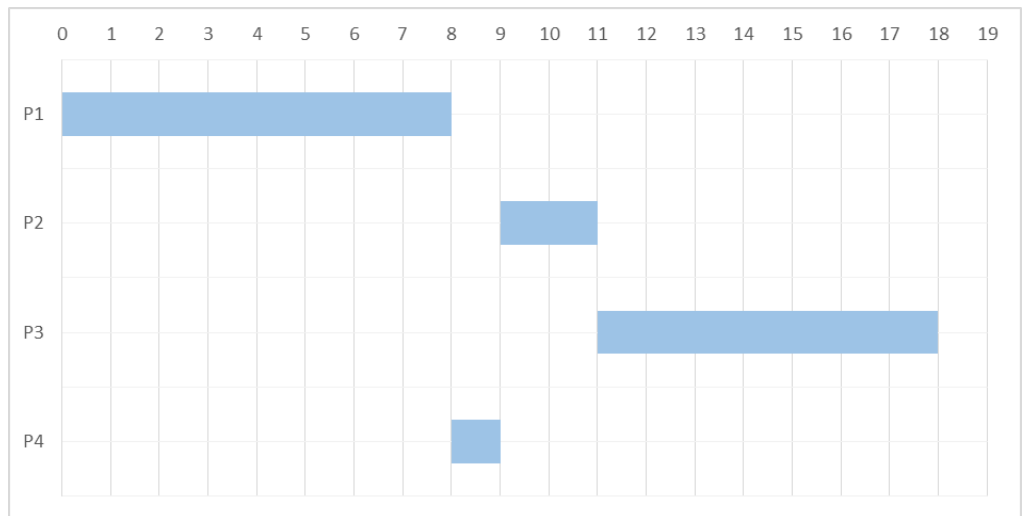
NPSJF – Prestazioni

formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

– (In questo caso : $T_{servizio} = T_{burst}$)

– $R_{a P1} = 8 / (8-0) = 1$

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1



NPSJF – Prestazioni

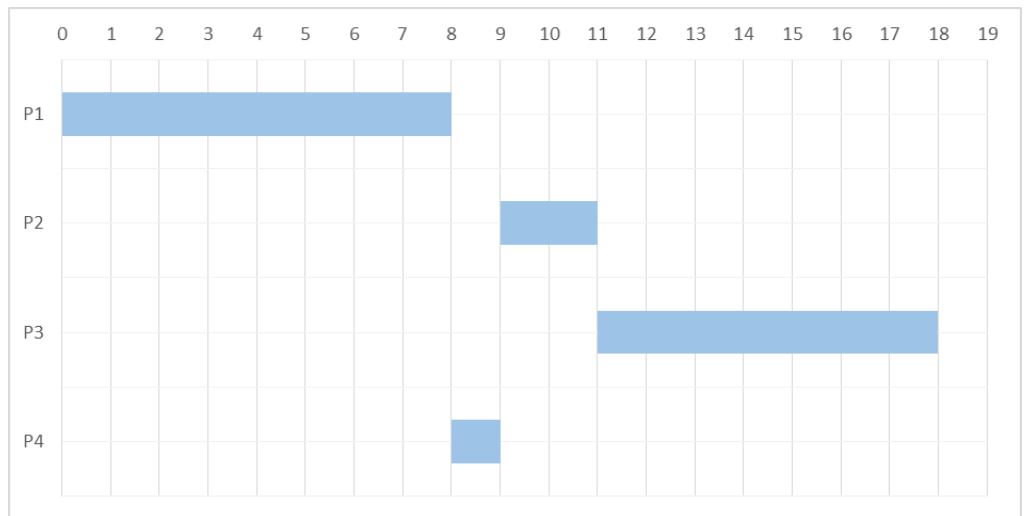
formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

– (In questo caso : $T_{servizio} = T_{burst}$)

– $R_{a\ P1} = 8 / (8-0) = 1$

– $R_{a\ P2} = 2 / (11-9) = 0.2$

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1



NPSJF – Prestazioni

formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

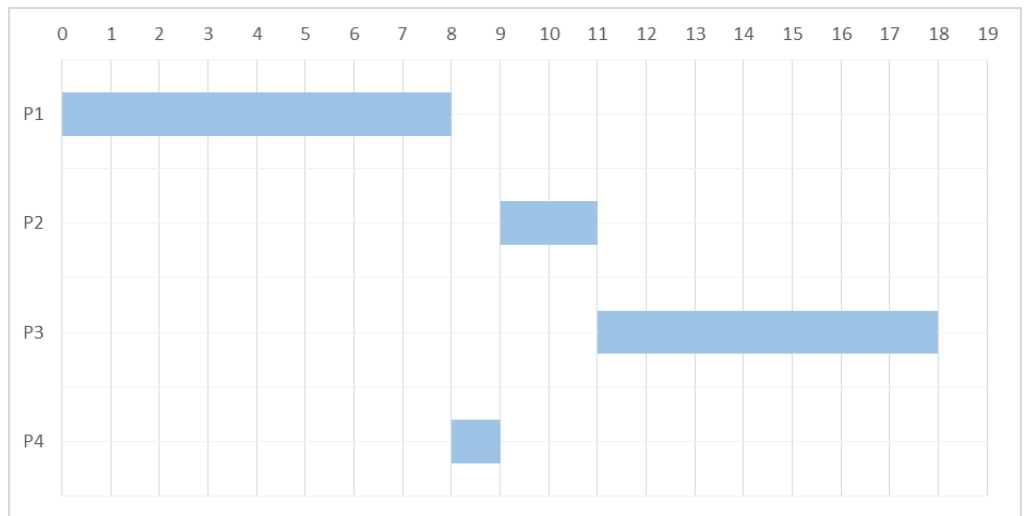
– (In questo caso : $T_{servizio} = T_{burst}$)

– $R_{a\ P1} = 8 / (8-0) = 1$

– $R_{a\ P2} = 2 / (11-9) = 0.2$

– $R_{a\ P3} = 7 / (18-3) = 0.4666$

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1



NPSJF – Prestazioni

formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

– (In questo caso : $T_{servizio} = T_{burst}$)

– $R_{a\ P1} = 8 / (8-0) = 1$

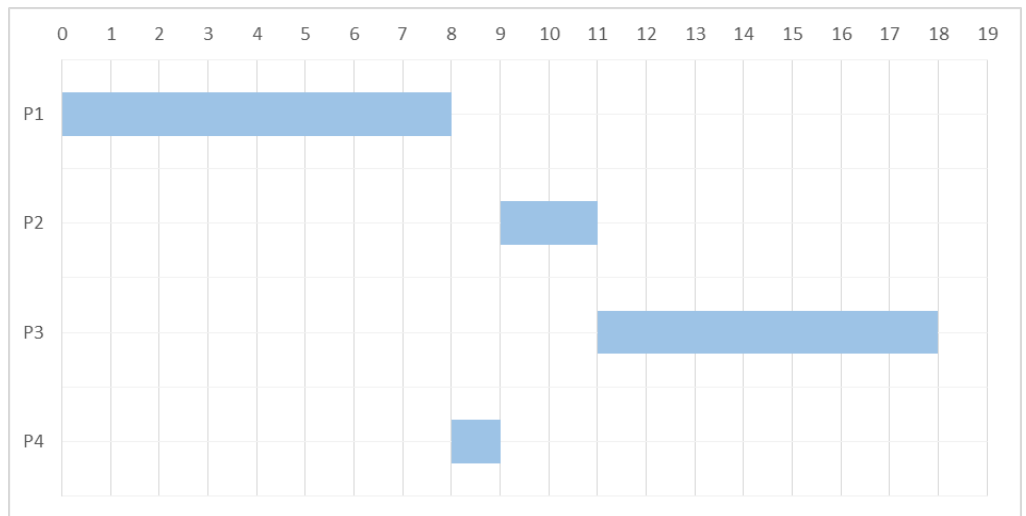
– $R_{a\ P2} = 2 / (11-9) = 0.2$

– $R_{a\ P3} = 7 / (18-3) = 0.4666$

– $R_{a\ P4} = 1 / (9-5) = 0.25$

– → Prestazioni medie = 0.48

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	7
P4	5	1



PSJF – Preemptive Shortest Job First

- theory:

t_n	lunghezza dell'n-esimo CPU burst
τ_{n+1}	valore predetto del prossimo CPU burst
α	$0 \leq \alpha \leq 1$
$\tau_{n+1} = \alpha t_n + (1 - \alpha) \tau_n$	

- $\alpha = 0$: non ho **storia**, $\tau_{n+1} = \tau_n$
- $\alpha = 1$: vale ultimo valore REALE $\tau_{n+1} = T_n$

PSJF – Preemptive Shortest Job First

- **Def:** "Shortest Job First Scheduling, jobs are put into ready queue as they arrive, but as a process with short burst time arrives, the existing process is preempted or removed from execution, and the shorter job is executed first.
- **Logica:** se arriva un P con burst piu piccolo, lo servo con pre-emption.

PSJF – Preemptive Shortest Job First

- Dati i seguenti processi con i rispettivi tempo di arrivo, tempo di burst

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	6 *
P4	5	1

- Alpha = 0.5

$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n$$

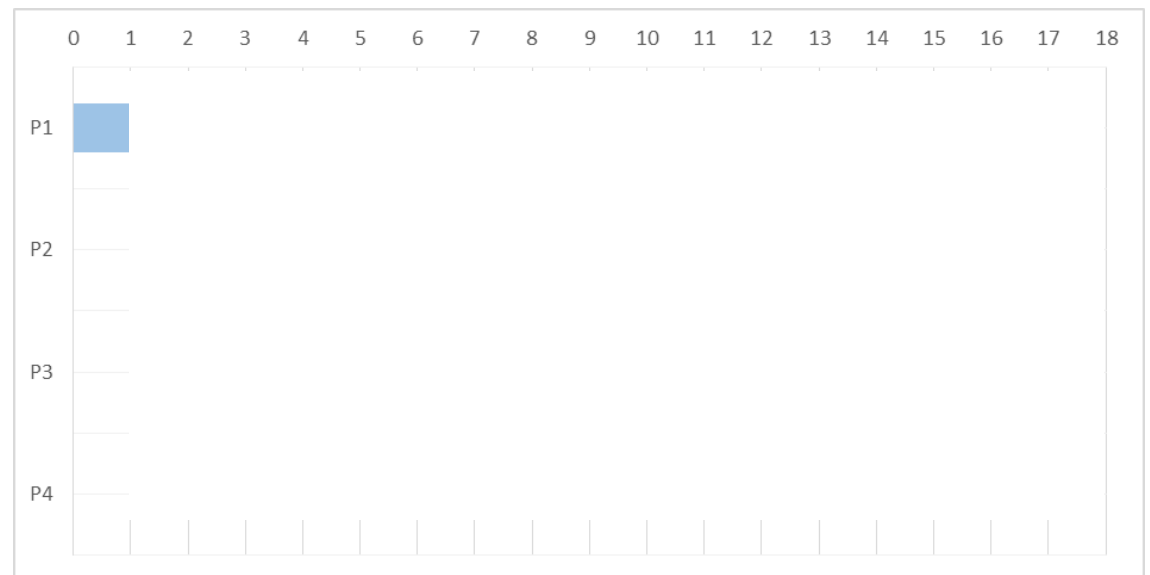
- Disegnare il grafico temporale dello scheduling
- Calcolare il tempo medio di attesa
- Calcolare il rapporto di prestazioni

* errore slide OLD era 7

PSJF – Preemptive Shortest Job First

- Parte P1
- Ma al tempo $T=1$ arriva P2...

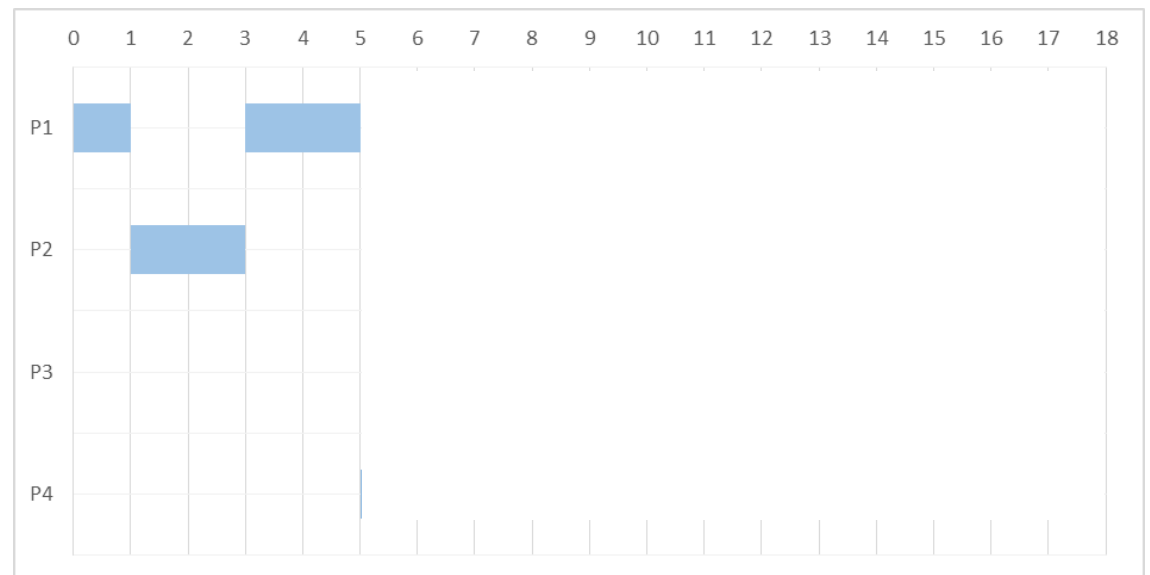
Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	6
P4	5	1



PSJF – Preemptive Shortest Job First

- Parte P1
- P2 parte, dura 2 T..
- Riparte P1 (non ho altri in coda..)

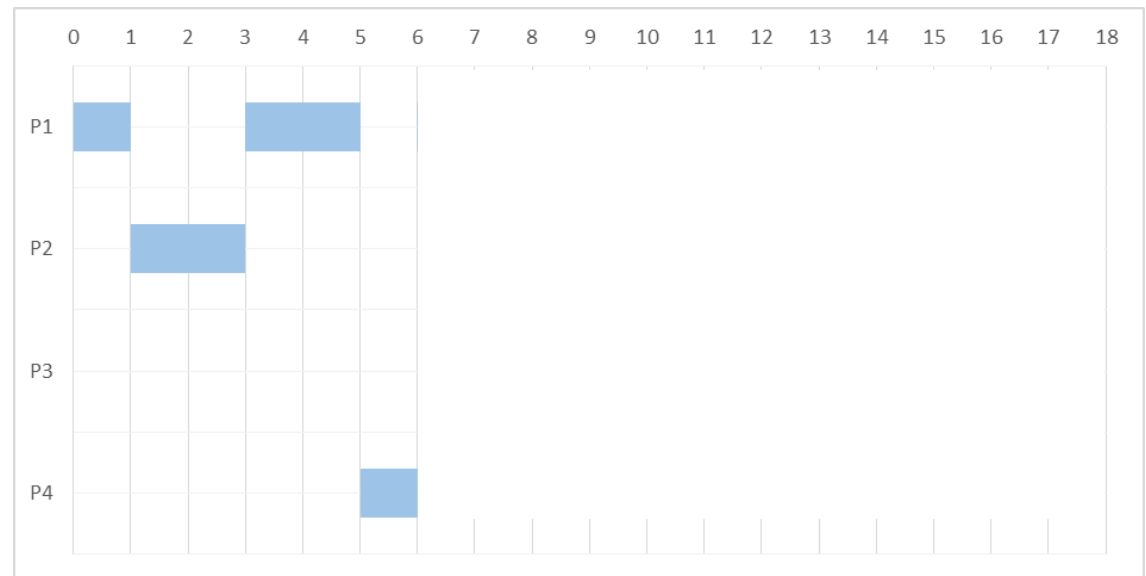
Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	6
P4	5	1



PSJF – Preemptive Shortest Job First

- Parte P1
- P2 parte, dura 2 T..
- Riparte P1 (non ho altri in coda..)
- Arriva P3 a $T = 3$,
ma sta runnando P1 (tengo in coda..)
- $T = 5$ arriva P4 che ha burst = 1:
"sorpassa" P3... e termina..

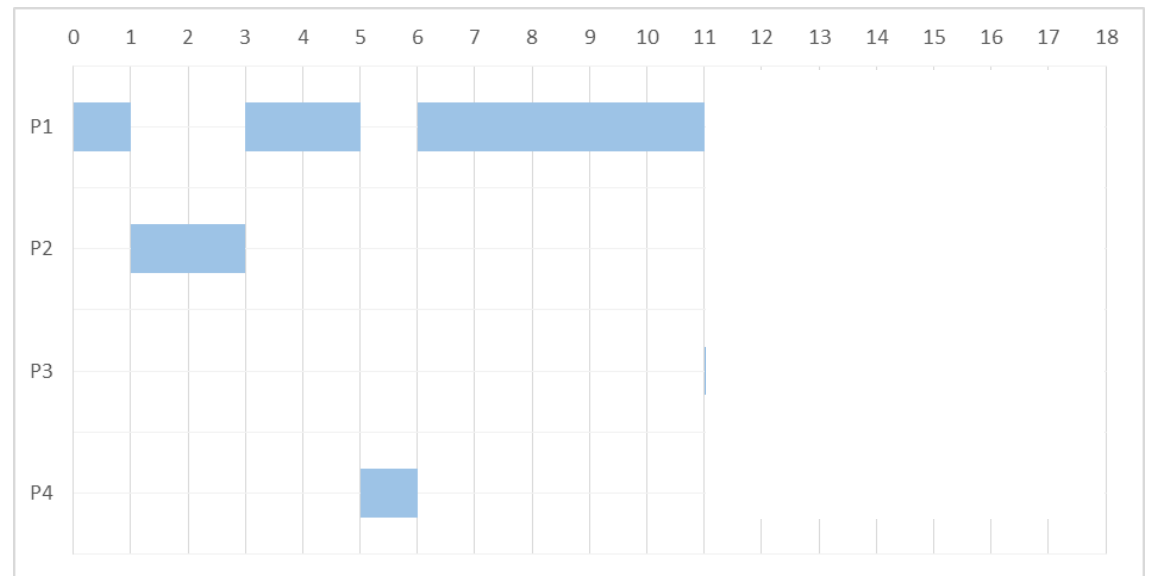
Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	6
P4	5	1



PSJF – Preemptive Shortest Job First

- P1 riparte, ha solo 5 T da fare,
- P3 ne ha 6...tengo in coda..)
-

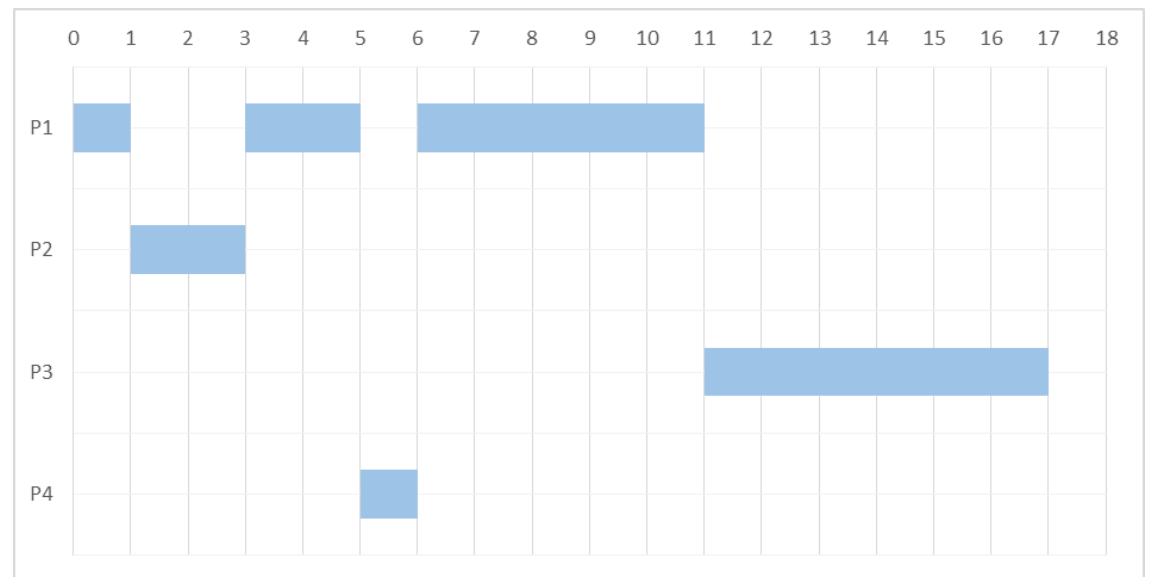
Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	6
P4	5	1



PSJF – Preemptive Shortest Job First

- P1 termina
- Parte P3

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	6
P4	5	1



PSJF – Preemptive Shortest Job First: Attesa

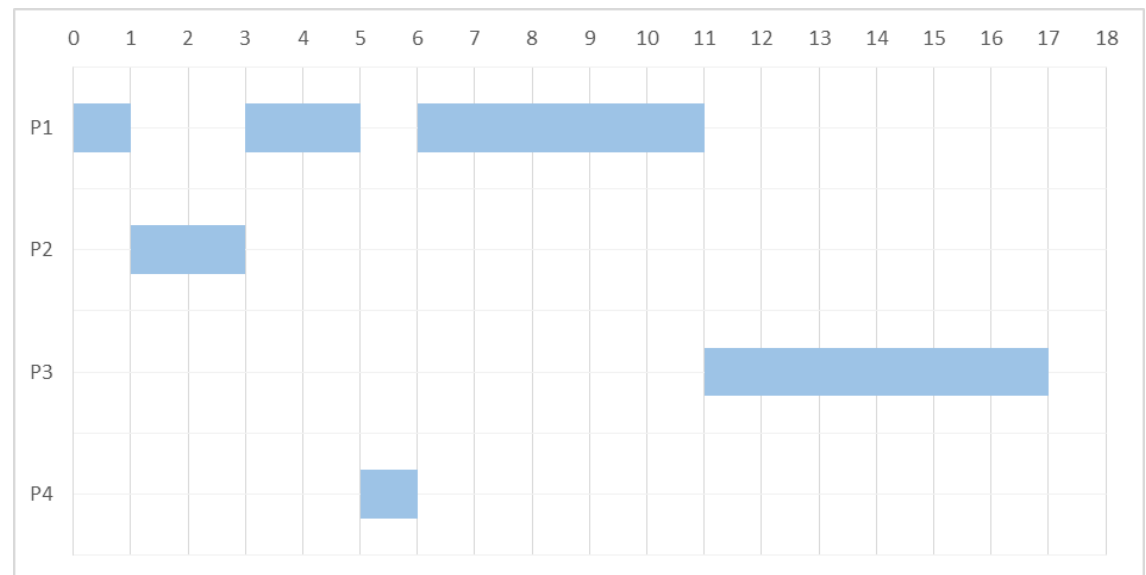
formula:

$$T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIV}$$

(In questo caso : $T_{servizio} = T_{burst}$)

$$T_{a P1} = 11 - 8 - 0 = 3$$

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	6
P4	5	1



PSJF – Preemptive Shortest Job First: Attesa

formula:

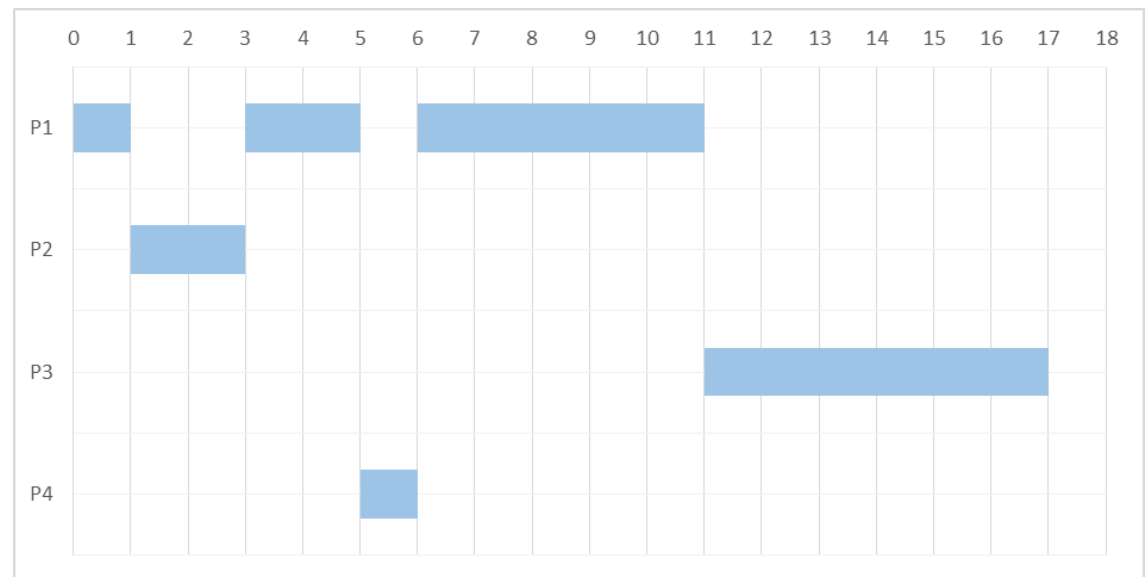
$$T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIV}$$

(In questo caso : $T_{servizio} = T_{burst}$)

$$T_{a P1} = 11 - 8 - 0 = 3$$

$$T_{a P2} = 3 - 2 - 1 = 0$$

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	6
P4	5	1



PSJF – Preemptive Shortest Job First: Attesa

formula:

$$T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVA}$$

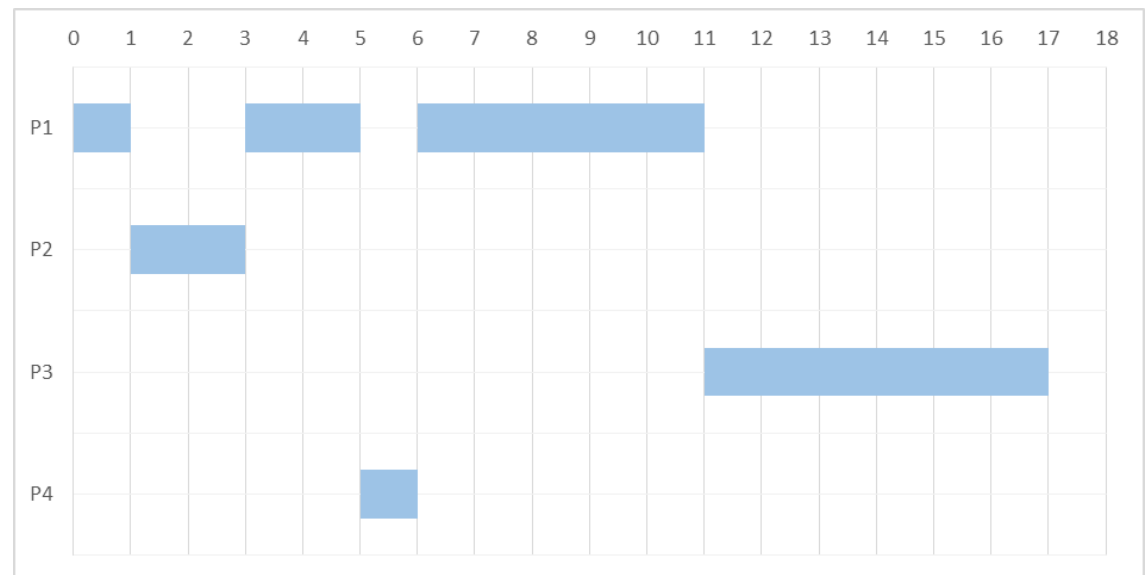
(In questo caso : $T_{servizio} = T_{burst}$)

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	6
P4	5	1

$$T_{a P1} = 11 - 8 - 0 = 3$$

$$T_{a P2} = 3 - 2 - 1 = 0$$

$$T_{a P3} = 17 - 6 - 3 = 8$$



PSJF – Preemptive Shortest Job First: Attesa

formula:

$$T_{ATTESA} = T_{FINE} - T_{SERVIZIO} - T_{ARRIVA}$$

(In questo caso : $T_{servizio} = T_{burst}$)

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	6
P4	5	1

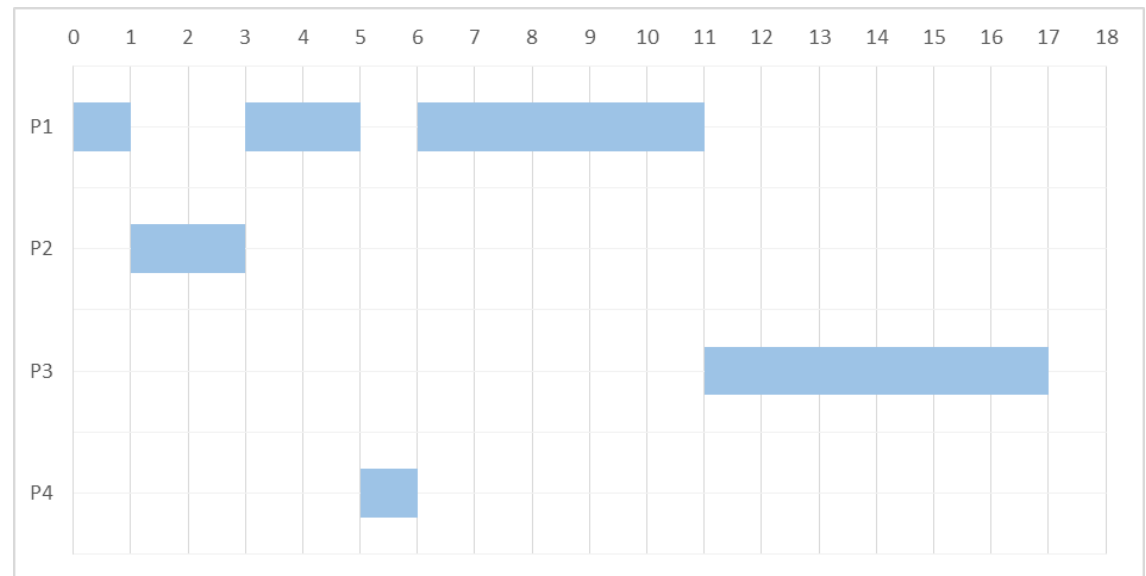
$$T_{a P1} = 11 - 8 - 0 = 3$$

$$T_{a P2} = 3 - 2 - 1 = 0$$

$$T_{a P3} = 17 - 6 - 3 = 8$$

$$T_{a P4} = 6 - 1 - 5 = 0$$

→ Attesa media = 2.75



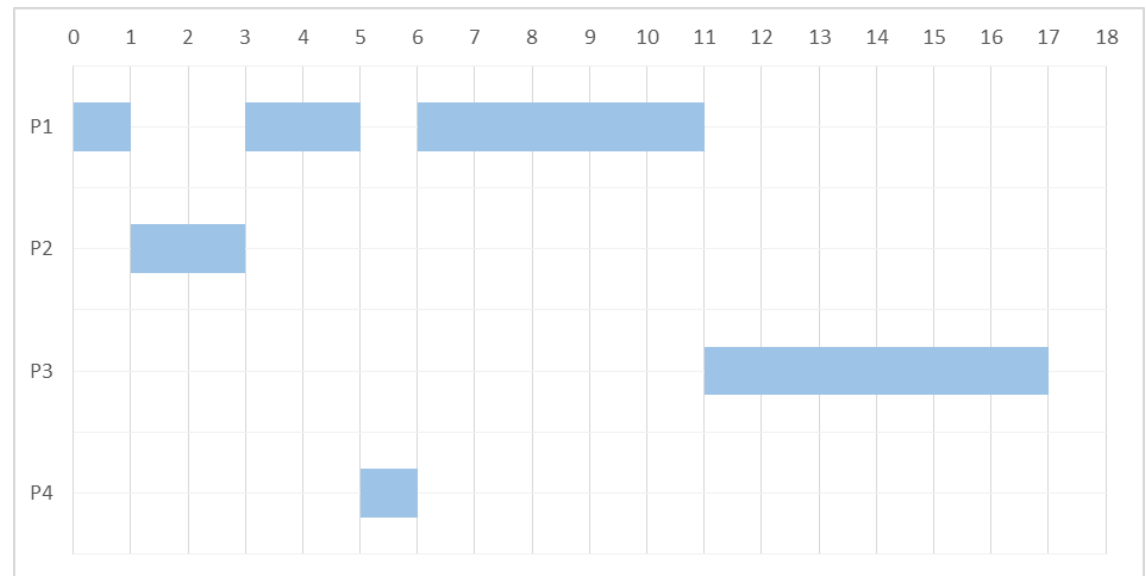
PSJF – Preemptive Shortest Job First: Prestazioni

formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

(In questo caso : $T_{servizio} = T_{burst}$)

– $R_{a P1} = 8 / (11 - 0) = 0.73$

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	6
P4	5	1



PSJF – Preemptive Shortest Job First: Prestazioni

formula: $R = \frac{T_{SERVIZIO}}{T_{FINE} - T_{ARRIVO}}$

(In questo caso : $T_{servizio} = T_{burst}$)

Processo	Tempo di arrivo	Tempo di burst
P1	0	8
P2	1	2
P3	3	6
P4	5	1

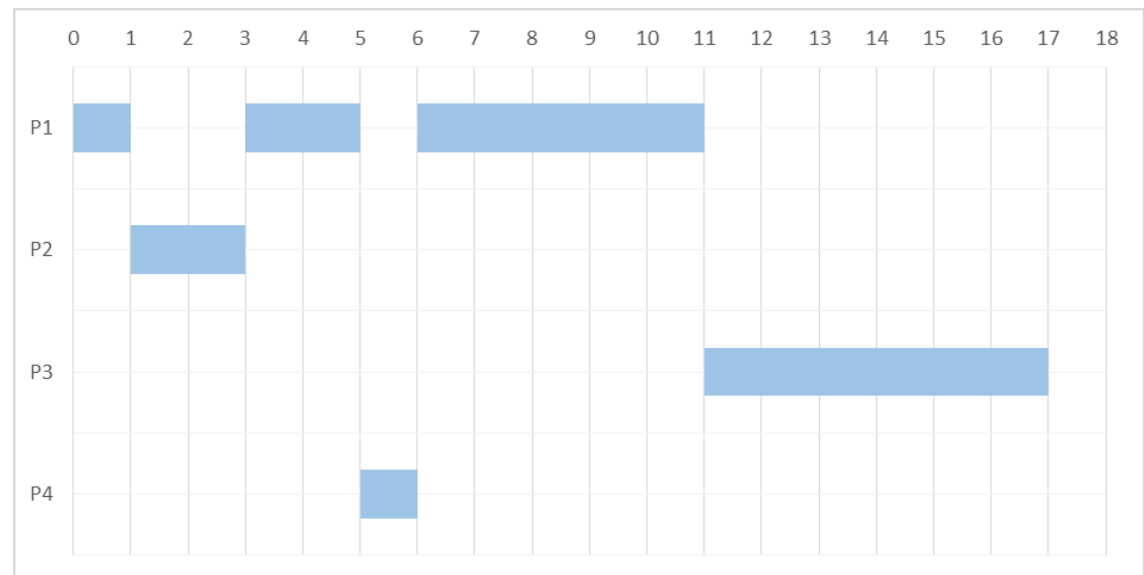
– $R_{a P1} = 8 / (11 - 0) = 0.73$

■ *Velocizzando:*

– $R_{a P2} = 2 / (3 - 1) = 1$

– $R_{a P3} = 6 / (17 - 3) = 0.43$

– $R_{a P4} = 1 / (6 - 5) = 1$



→ Prestazioni medie = 0.79

SPN – Shortest Process Next

- Dati i seguenti processi con i rispettivi tempo di arrivo, tempo di servizio

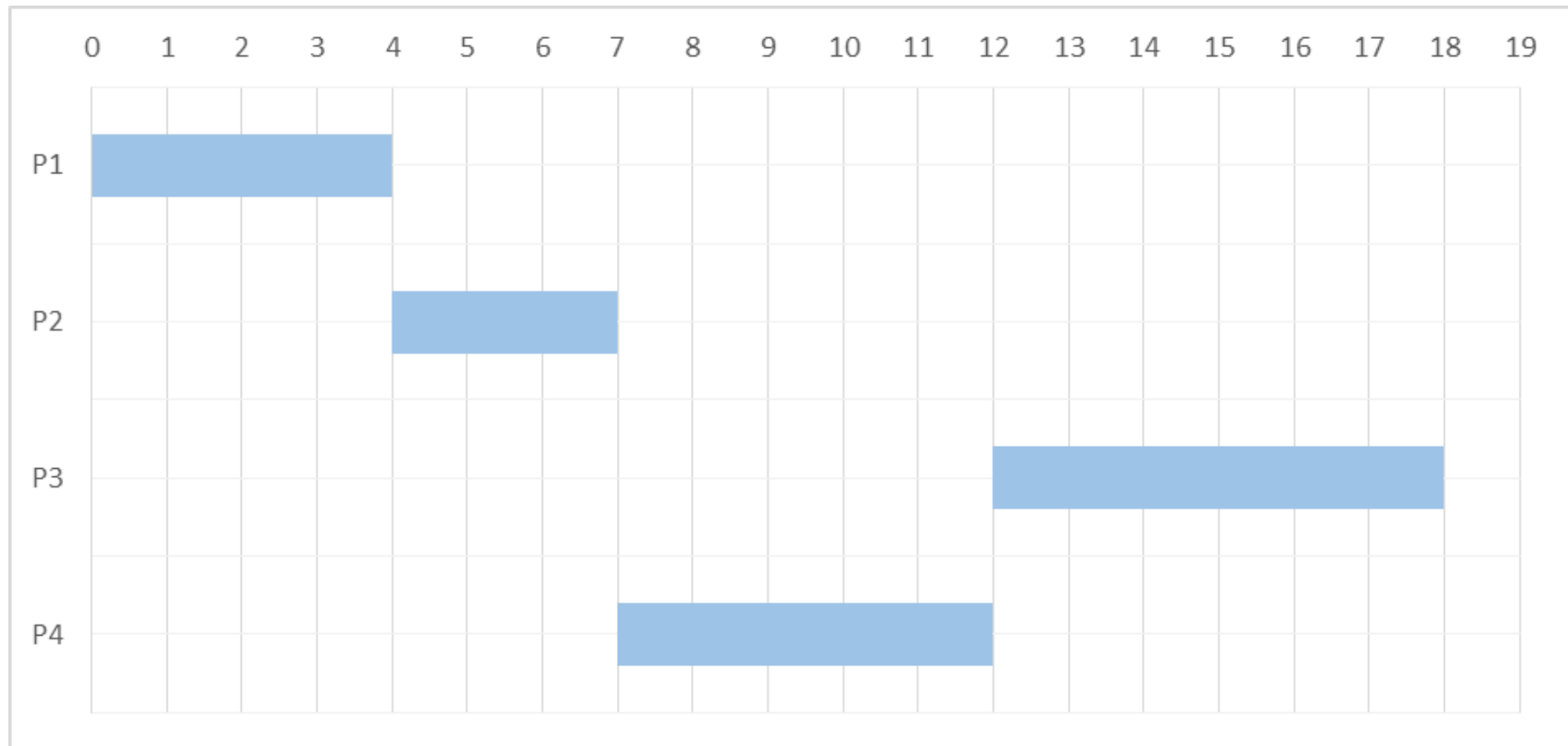
Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

- Disegnare il grafico temporale dello scheduling
- Calcolare il tempo medio di attesa
- Calcolare il rapporto di prestazioni



SPN – Shortest Process Next

■ Lo scheduling risultante è



■ Attesa

– $P1 = 0$, $P2 = 3$, $P3 = 9$, $P4 = 2$

→ Attesa media = 3.5

■ Prestazioni

– $P1 = 1$, $P2 = 0.5$, $P3 = 0.40$, $P4 = 0.71$

→ Prestazioni medie = 0.65

HRRN – High Response Ratio Next

- Dati i seguenti processi con i rispettivi tempo di arrivo, tempo di servizio

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

- Priorità del Processo

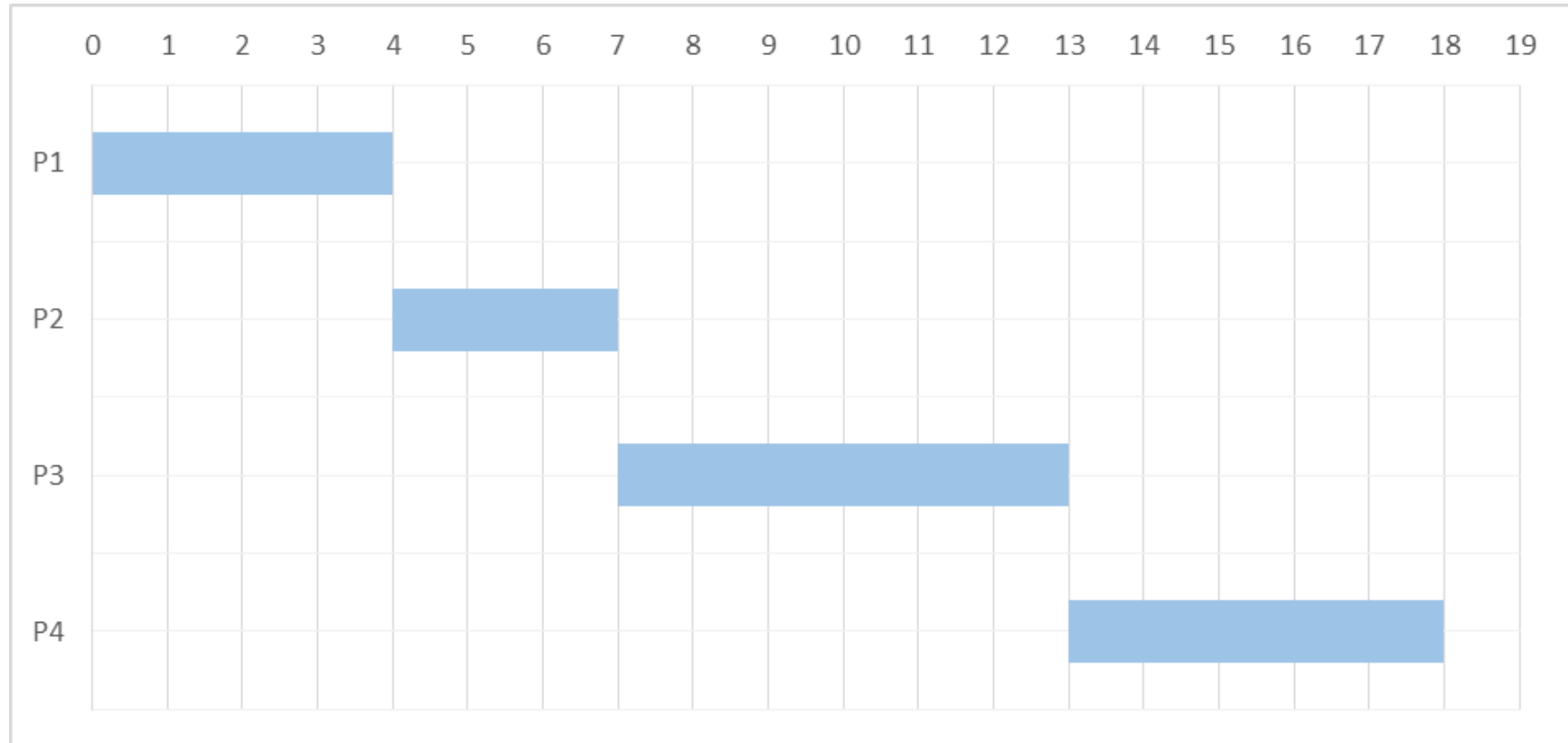
$$P = \frac{T_{ATTESA} + T_{EXEC}}{T_{EXEC}}$$

- Disegnare il grafico temporale dello scheduling
- Calcolare il tempo medio di attesa
- Calcolare il rapporto di prestazioni



HRRN – High Response Ratio Next

■ Lo scheduling risultante è



■ Attesa

– $P1 = 0, P2 = 3, P3 = 4, P4 = 8$

→ Attesa media = 3.75

■ Prestazioni

– $P1 = 1, P2 = 0.5, P3 = 0.6, P4 = 0.38$

→ Prestazioni medie = 0.62

SRT – Shortest Remaining Time

- Dati i seguenti processi con i rispettivi tempo di arrivo, tempo di servizio

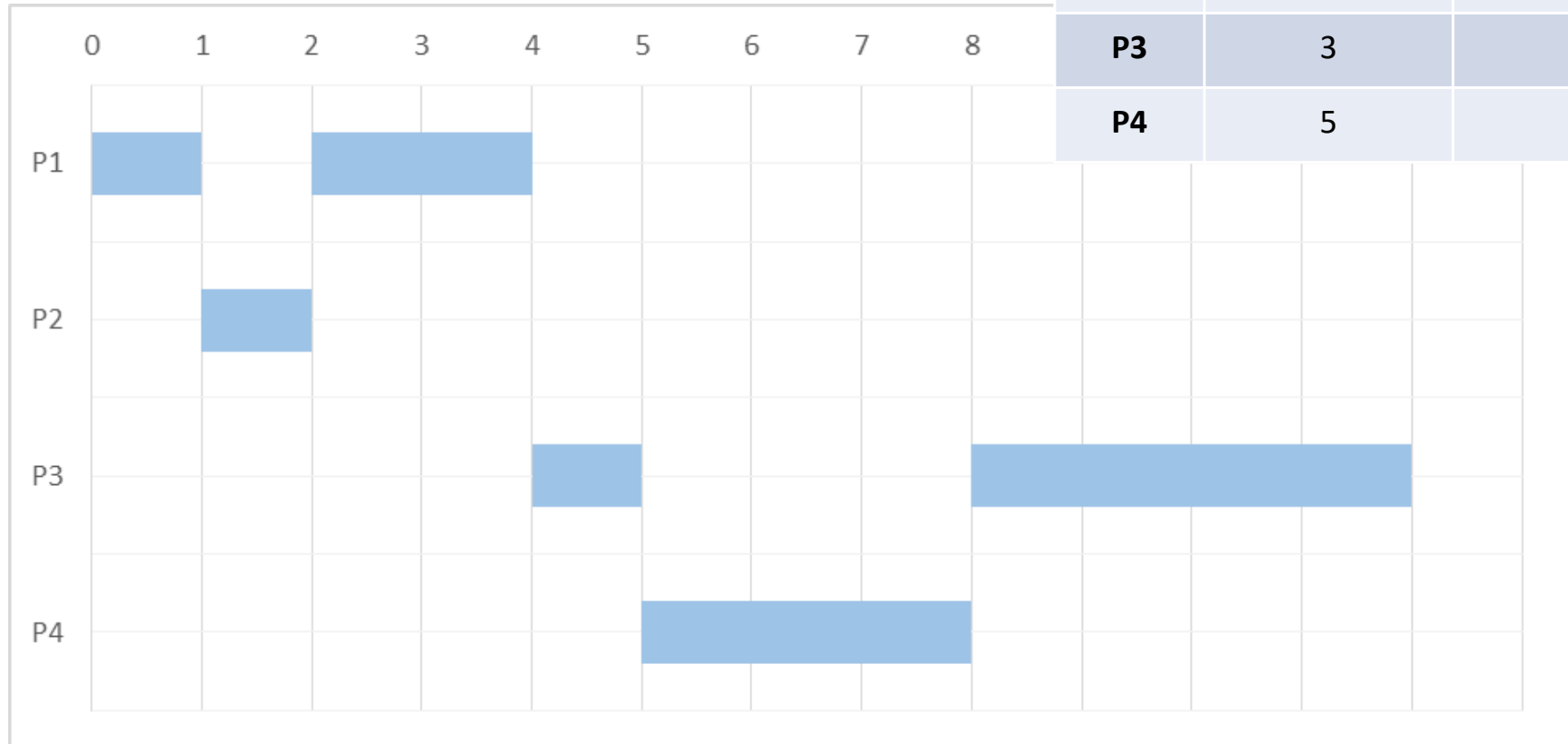
Processo	Tempo di arrivo	Tempo di servizio
P1	0	3
P2	1	1
P3	3	5
P4	5	3

- Disegnare il grafico temporale dello scheduling
- Calcolare il tempo medio di attesa
- Calcolare il rapporto di prestazioni



SRT – Shortest Remaining Time

■ Lo scheduling risultante è



■ Attesa

– $P1 = 1, P2 = 0, P3 = 3, P4 = 0$

→ Attesa media = 1.25

■ Prestazioni

– $P1 = 0.75, P2 = 1, P3 = 0.56, P4 = 1$

→ Prestazioni medie = 0.83

MLQ – Multilevel Queue

- Dati i seguenti processi con i rispettivi tempo di arrivo, tempo di servizio

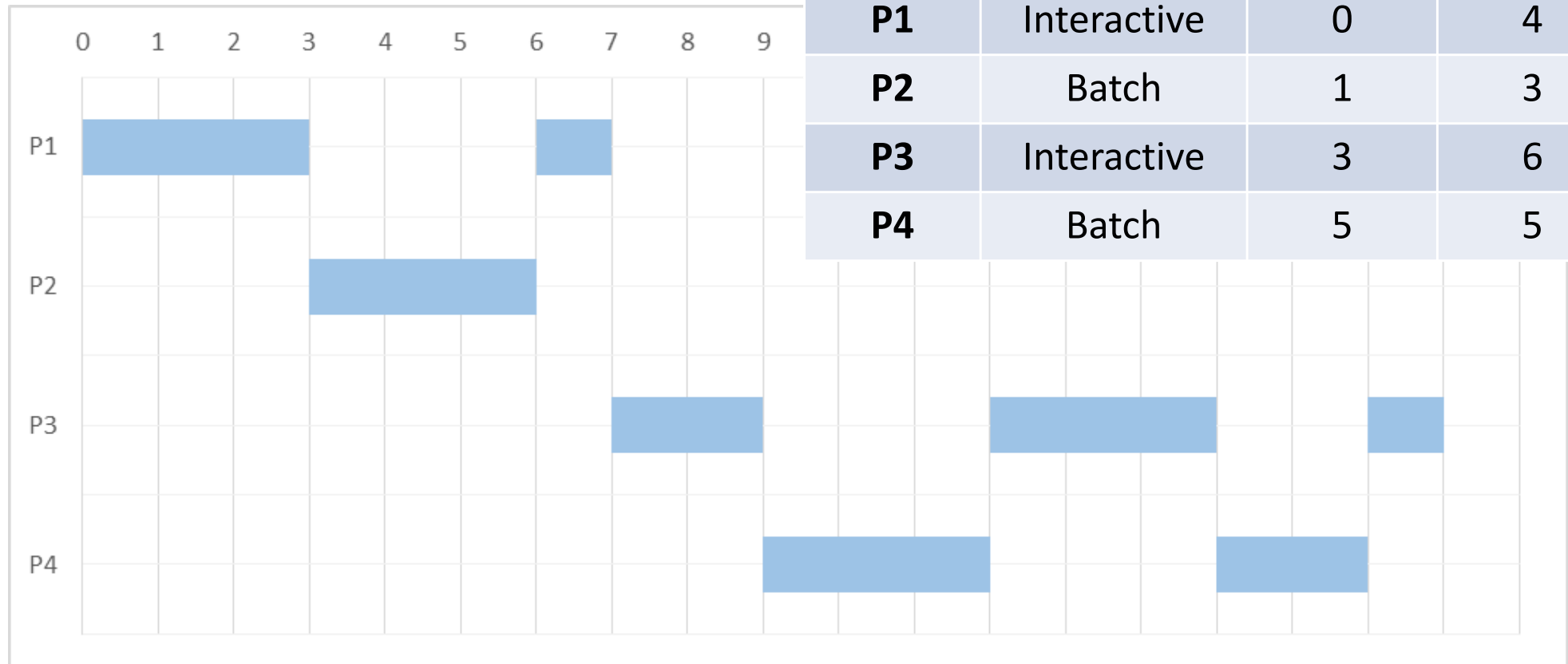
Processo	Tipo	Tempo di arrivo	Tempo di servizio
P1	Interactive	0	4
P2	Batch	1	3
P3	Interactive	3	6
P4	Batch	5	5

- Quanto di tempo 3 cicli
- Due code
 - Coda Interactive FCFS
 - Coda Batch FCFS
- Politica intercoda RR
 - Disegnare il grafico temporale dello scheduling
 - Calcolare il tempo medio di attesa
 - Calcolare il rapporto di prestazioni



MLQ – Multilevel Queue

■ Lo scheduling risultante è



■ Attesa

– P1 = 3, P2 = 2, P3 = 9, P4 = 7

→ Attesa media = 5.25

■ Prestazioni

– P1 = 0.57, P2 = 0.60, P3 = 0.40, P4 = 0.42

→ Prestazioni medie = 0.50

MLFQ – Multilevel Feedback Queue

- Dati i seguenti processi con i rispettivi tempo di arrivo, tempo di servizio

Processo	Tempo di arrivo	Tempo di servizio
P1	0	4
P2	1	3
P3	3	6
P4	5	5

- Tre code

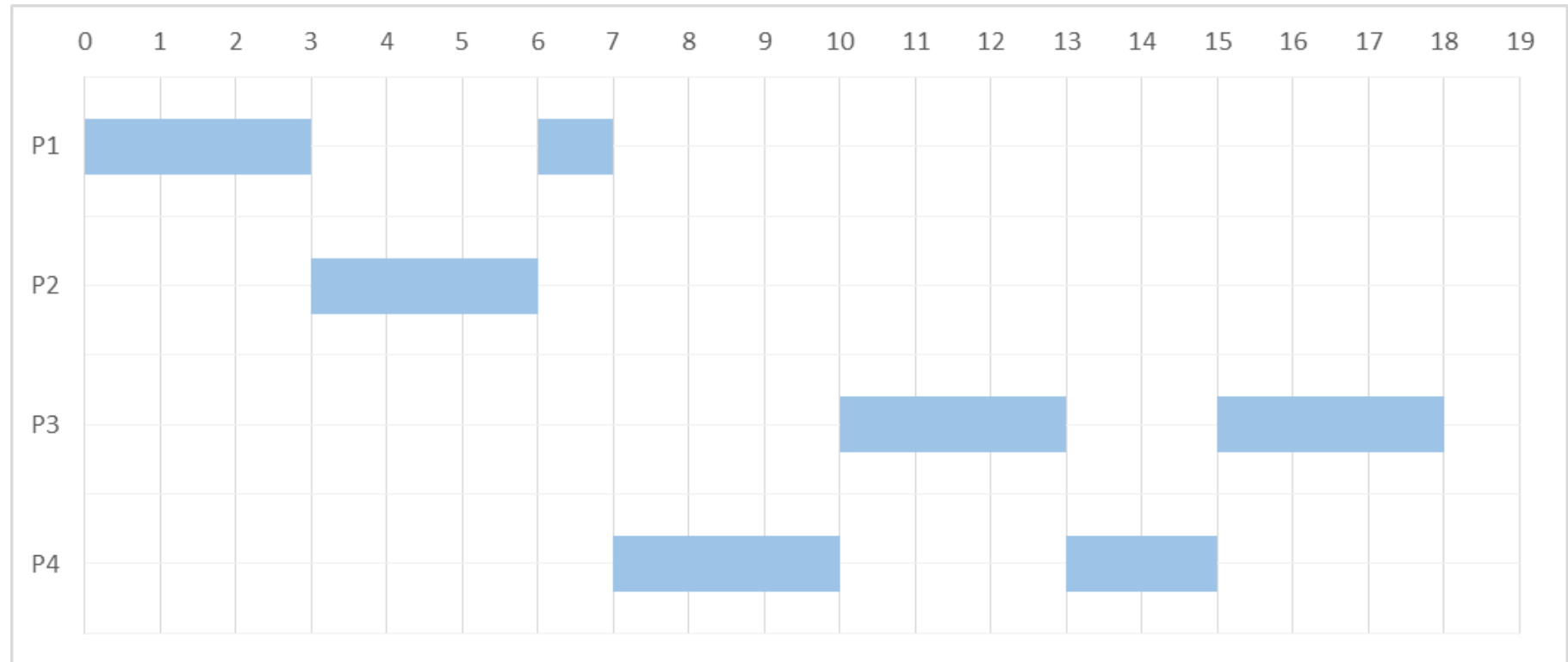
- Q1 RR Quanto = 3
- Q2 RR Quanto = 5
- Q3 FCFS

- Disegnare il grafico temporale dello scheduling
- Calcolare il tempo medio di attesa
- Calcolare il rapporto di prestazioni



MLFQ – Multilevel Feedback Queue

■ Lo scheduling risultante è



■ Attesa

– $P1 = 3, P2 = 2, P3 = 9, P4 = 5$

→ Attesa media = 4.75

■ Prestazioni

– $P1 = 0.57, P2 = 0.60, P3 = 0.40, P4 = 0.50$

→ Prestazioni medie = 0.52

Memoria virtuale: recap

- Riduce costi
- Spazio Logico (virtuale) \geq spazio fisico
- Pagine stessa grandezza
- (Stesso offset)
- $NPV \geq NPF$
- Page fault
- Bit x validita'
- Mapping fatto in HW (con supporto nel .S.O. ...)
- Tabella della pagine

Memoria virtuale: esempio 1

- Es1

32 bit spazio logico (virtuale)

30 bit spazio fisico

Calcoliamo NPF NPV bit necessari.

Memoria virtuale: esempio 1

32 bit spazio logico (virtuale)

4 GB

30 bit spazio fisico

1 GB

Supponiamo page Size 4 K

...

Memoria virtuale: esempio 1

32 bit spazio logico (virtuale) 4 GB

30 bit spazio fisico 1 GB

Page Size 4 K

$$\text{NPL} = 4\text{GB} / 4\text{KB} = 2^{32} / 2^{12} = 2^{20} \rightarrow 20\text{ BIT x individuarle}$$

..

Memoria virtuale: esempio 1

32 bit spazio logico (virtuale) 4 GB

30 bit spazio fisico 1 GB

Page Size 4 K

$NPL = 4GB / 4 KB = 2^{32} / 2^{12} = 2^{20} \rightarrow 20 \text{ BIT x individuarle}$

$NPF = 2^{30} / 2^{12} = 2^{18} \rightarrow 18 \text{ bit}$

La Page Table 20 bit

Memoria virtuale: esempio 1

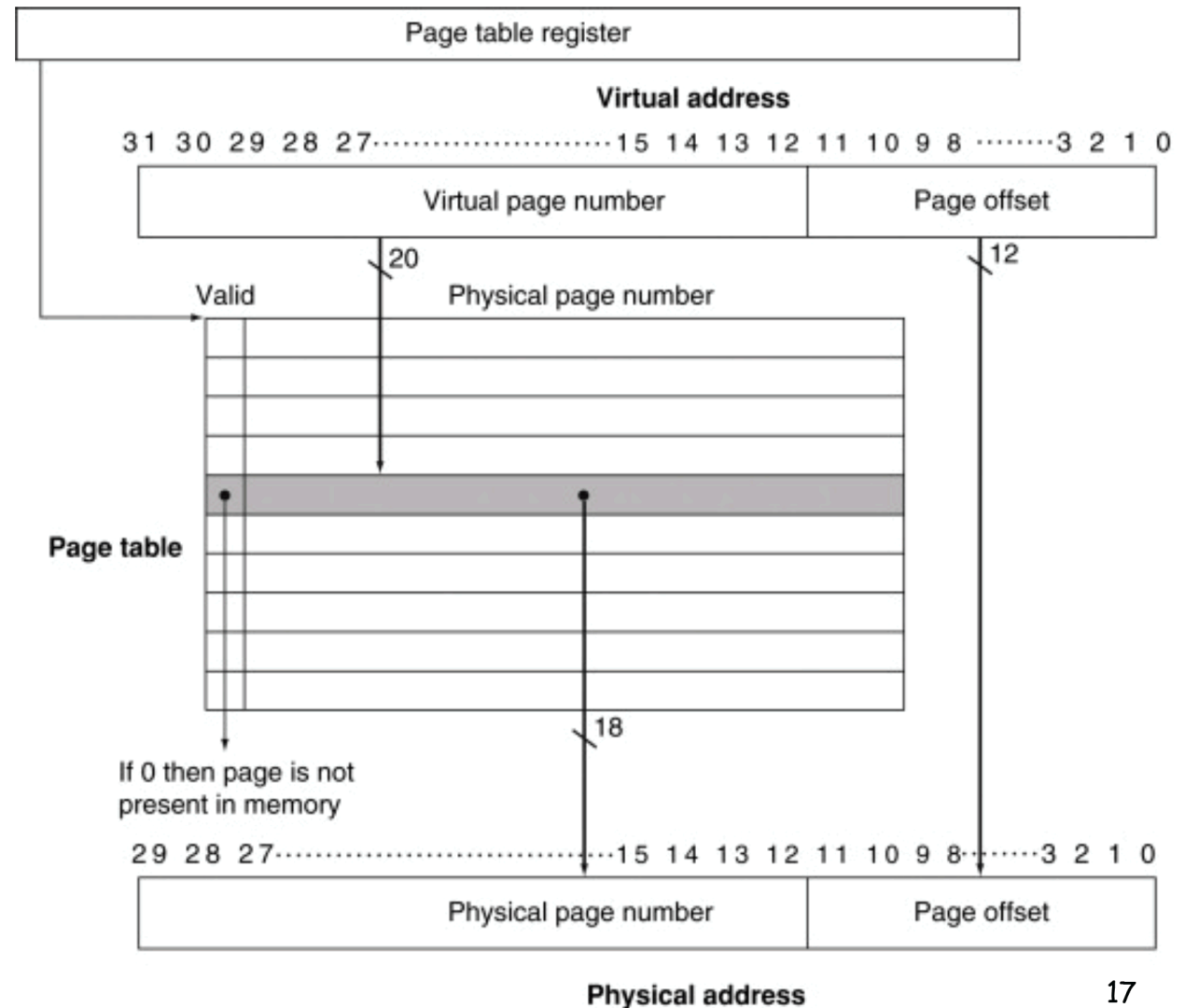
$NPL = 4GB / 4 KB = 2^{32} / 2^{12} = 2^{20} \rightarrow 20 \text{ BIT x individuarle}$

$NPF = 2^{30} / 2^{12} = 2^{18} \rightarrow 18 \text{ bit}$

La Page Table 20 righe

VPN fa da indice

Bit di validita': 0 == page fault



Memoria virtuale: esempio 2

48 bit spazio logico (virtuale)

40 bit spazio fisico

page Size 4 K

Calcolare NPF NPV e offset.

Memoria virtuale: esempio 2

48 bit spazio logico (virtuale) (256 TB) 40 bit spazio fisico (1TB)

page Size 4 K (12 bit)

48 - 12 36 bit x NPV

40 - 12 = 28 bit x NPF

Offset 12 bit (4K pages)

Notazioni x esercizi

La pagina virtuale n dell'area virtuale A del programma o processo P è indicata con la notazione AP_n , dove:

- A indica un tipo di area virtuale secondo la convenzione:

C (codice)

D (dati),

P (pila)

$COND$ (area dati condivisa)

- P indica il programma o il processo

- n indica il numero di pagina nell'ambito dell'area virtuale

quindi

$\langle \text{tipo} \rangle \langle \text{processo} \rangle \langle n.\text{pagina} \rangle$

Notazioni x esercizi: <tipo><processo><n.pagina>

- A indica un tipo di area virtuale secondo la convenzione:

C (codice)

D (dati),

P (pila)

COND (area dati condivisa)

- P indica il programma o il processo

- n indica il numero di pagina nell'ambito dell'area virtuale

Esempio:

DQ3 indica la pagina 3 dell'area dati del processo Q

Nota:

non è la pagina con $NPV = 3$,

perchè in generale l'area DP non inizia con $NPV = 0$

VM: Esercizio 1

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

la memoria centrale fisica ha capacità di 32 Kbyte,
quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte.

Si chiede di svolgere i punti seguenti...

A b c d e, li vedremo passo passo..

(esercizio-memvirt-1.pdf)

VM: Esercizio 1 richiesta a

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

la memoria centrale fisica ha capacità di 32 Kbyte,
quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte.

A) Si definisca la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi

NPF, Spiazzamento fisico, NPL, Spiazzamento logico

VM: Esercizio 1 richiesta a: Soluzione

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

la memoria centrale fisica ha capacità di 32 Kbyte,
quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte.

A) Si definisca la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi:

NPF, Spiazzamento fisico, NPL, Spiazzamento logico

NPF:

32 KByte / 4 KByte = 8, di solito si scrivono i Bit necessari quindi

NPF $2^3 \rightarrow 3$ bit

Nota: talora la soluzione dice semplicemente NPF = 3...

VM: Esercizio 1 richiesta a: Soluzione

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

la memoria centrale fisica ha capacità di 32 Kbyte,
quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte.

A) Si definisca la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi:

NPF, Spiazzamento fisico, NPL, Spiazzamento logico

NPF: 3,

Spiazzamento fisico: $4K = 2^{12}$ quindi bastano 12 bit.

VM: Esercizio 1 richiesta a: Soluzione

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

la memoria centrale fisica ha capacità di 32 Kbyte,
quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte.

A) Si definisca la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi:

NPF, Spiazzamento fisico, NPL, Spiazzamento logico

NPF: 3,

Spiazzamento fisico: 12,

NPL: identico al NPF, 32K /4 K bastano 3 bit.

VM: Esercizio 1 richiesta a: Soluzione

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

la memoria centrale fisica ha capacità di 32 Kbyte,
quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte.

A) Si definisca la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi:

NPF, Spiazzamento fisico, NPL, Spiazzamento logico

NPF: 3,

Spiazzamento fisico: 12,

NPL: 3,

Spiazzamento logico: non cambi nulla rispetto a quello fisico:

12

VM: Esercizio 1 richiesta b

Nel sistema vengono creati alcuni processi, indicati nel seguito con P, Q, R, S.

I programmi eseguiti da tali processi sono due: X e Y.

La dimensione iniziale dei segmenti dei programmi è la seguente:

CX: 8K DX: 4K PX:4K

CY:12K DY: 8K PY:4K

Si inserisca in tabella 1

La struttura in pagine della memoria virtuale

indir. virtuale	prog. X	prog. Y
0		
1		
2		
3		
4		
5		
6		
7		

(mediante la notazione definita sopra: CX0 CX1 DX0 PX0 ... CY0 ...).

VM: Esercizio 1 richiesta b: Soluzione

CX: 8K DX: 4K PX:4K

CY:12K DY: 8K PY:4K

- Tutti i processi P, Q, R, S eseguono X e Y
- Quindi verranno caricati:
 - Prima codice
 - poi dati ed
 - IN FONDO Lo stack..

Tutto nella VM e ovviamente UN SOLA copia di X e di Y....

VM: Esercizio 1 richiesta b: Soluzione

CX: 8K DX: 4K PX:4K

CY:12K DY: 8K PY:4K

CX 2 pagine...

indir. virtuale	prog. X	prog. Y
0	CX0	
1	CX1	
2		
3		
4		
5		
6		
7		

1) *memoria logica*

VM: Esercizio 1 richiesta b: Soluzione

CX: 8K DX: 4K PX:4K

CY:12K DY: 8K PY:4K

Idem

CY 3 pagine...

indir. virtuale	prog. X	prog. Y
0	CX0	CY0
1	CX1	CY1
2		CY2
3		
4		
5		
6		
7		

1) *memoria logica*

VM: Esercizio 1 richiesta b: Soluzione

CX: 8K DX: 4K PX:4K

CY:12K DY: 8K PY:4K

Data:

X 1 pagina,

Y 2 pagine

Subito adiacenti..

indir. virtuale	prog. X	prog. Y
0	CX0	CY0
1	CX1	CY1
2	DX0	CY2
3		DY0
4		
5		
6		
7		

1) *memoria logica*

VM: Esercizio 1 richiesta b: Soluzione

CX: 8K DX: 4K PX:4K

CY:12K DY: 8K PY:4K

STACK x definizione "in fondo"

indir. virtuale	prog. X	prog. Y
0	CX0	CY0
1	CX1	CY1
2	DX0	CY2
3		DY0
4		DY1
5		
6		
7	PX0	PY0

1) *memoria logica*

VM: Esercizio 1 richiesta c

A un certo istante di tempo t_0 sono terminati, nell'ordine, gli eventi seguenti:

1. creazione del processo P e lancio del programma Y (“fork” di P ed “exec” di Y)
2. creazione del processo Q e lancio del programma X (“fork” di Q ed “exec” di X)
3. accesso a 1 pagina dati e creazione di una nuova pagina di pila da parte di P
4. accesso a 1 pagina dati da parte di Q
5. creazione del processo R come figlio di P (“fork” eseguita da P)
6. creazione di 1 pagina di pila da parte di R

Con le seguenti ipotesi...

VM: Esercizio 1 richiesta c: ipotesi

- il lancio di una programma avviene caricando solamente la pagina di codice con l'istruzione di partenza e una sola pagina di pila
- il caricamento di pagine ulteriori è in Demand Paging (cioè le pagine si caricano su richiesta senza scaricare le precedenti fino al raggiungimento del numero massimo di pagine residenti)
- l'indirizzo (esadecimale) dell'istruzione di partenza di X è 14AF
- l'indirizzo (esadecimale) dell'istruzione di partenza di Y è 0231
- il numero di pagine residenti R vale 3 (tre)
- viene utilizzato l'algoritmo LRU (ove richiesto prima si dealloca una pagina di processo e poi si procede alla nuova assegnazione)
- le pagine meno utilizzate in ogni processo sono quelle caricate da più tempo, con la sola eccezione seguente: se è residente una sola pagina di codice, quella è certamente stata utilizzata recentemente
- al momento di una fork viene duplicata solamente la pagina di pila caricata più recentemente
- dopo la fork le pagine di codice possono essere condivise tra i processi padre e figlio, se ambedue i processi usano la stessa pagina virtuale...

VM: Esercizio 1 richiesta c: ipotesi II

- ipotizzando che l'allocazione delle pagine virtuali nelle pagine fisiche avvenga sempre in sequenza, senza buchi, a partire dalla pagina fisica 0,

si indichi, completando tabella 2, l'allocazione fisica delle pagine dei tre processi all'istante t_0 (notazione CP0 CP1 DP0 PP0 ... CQ0 ...).

indir. fisico	pagine allocate al tempo t_0
0	
1	
2	
3	
4	
5	
6	
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

1 step:

il processo P parte caricando la pagina CP0
(perché il programma Y ha istruzione di
partenza in pagina 0) e una pagina di pila PP0:

(No data, vedi ipotesi)

....

indir. fisico	pagine allocate al tempo t_0
0	
1	
2	
3	
4	
5	
6	
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

1 step:

il processo P parte caricando la pagina CP0
(perché il programma Y ha istruzione di
partenza in pagina **0**, addr. 0231)

e una pagina di pila PP0:

(No data, vedi ipotesi)

indir. fisico	pagine allocate al tempo t_0
0	CP0
1	PP0
2	
3	
4	
5	
6	
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

2 step:

successivamente Q carica CQ1 (perché il programma X ha istruzione di partenza in pagina 1) e PQ0

indir. fisico	pagine allocate al tempo t_0
0	CP0
1	PP0
2	CQ1
3	PQ0
4	
5	
6	
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

3 step:

P carica DP0, raggiungendo il limite delle pagine residenti (3), e quindi per caricare la pagina PP1 deve eliminare PP0 (vedi regole di utilizzazione indicate nel tema)

indir. fisico	pagine allocate al tempo t_0
0	CP0
1	PP0 PP1
2	CQ1
3	PQ0
4	DP0
5	
6	
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

4 step:

P carica DP0, raggiungendo il limite delle pagine residenti (3), e quindi per caricare la pagina PP1 deve eliminare PP0 (vedi regole di utilizzazione indicate nel tema)

Q carica DQ0 e raggiunge 3 pagine caricate

indir. fisico	pagine allocate al tempo t_0
0	CP0
1	PP0 PP1
2	CQ1
3	PQ0
4	DP0
5	DQ0
6	
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

5 step:

il nuovo processo R non ha bisogno di caricare CR0, perché esegue lo stesso programma di P e quindi CR0 è uguale a CP0,

quindi carica solamente PR1 (che è la copia della pagina PP1, ma conterrà un diverso pid)

indir. fisico	pagine allocate al tempo t_0
0	CP0 (= CR0)
1	PP0 PP1
2	CQ1
3	PQ0
4	DP0
5	DQ0
6	PR1
7	

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta c: soluzione

6 step:

R2 carica PR2

indir. fisico	pagine allocate al tempo t_0
0	CP0 (= CR0)
1	PP0 PP1
2	CQ1
3	PQ0
4	DP0
5	DQ0
6	PR1
7	PR2

2) memoria fisica, istante t_0

VM: Esercizio 1 richiesta d

- A un certo istante di tempo $t_1 > t_0$ sono terminati gli eventi seguenti:

7) terminazione del processo P (exit)

8) esecuzione della funzione “exec Y” (lancio di Y) nel processo Q e conseguente trasformazione di Q in processo S (si noti che Q si trasforma in S ma il pid resta lo stesso perché non c’è “fork”)

9) accesso a 2 pagine di dati per il processo S

.....

VM: Esercizio 1 richiesta d

Si completi la tabella 3 nelle medesime ipotesi delineate nel precedente punto (c) e supponendo che, dovendo utilizzare una pagina fisica libera, venga **sempre** utilizzata la pagina fisica libera avente indirizzo minore. Si aggiorni anche la tabella 1A (non è indispensabile).

indir. fisico	pagine allocate al tempo t_1
0	
1	
2	
3	
4	
5	
6	
7	

3) memoria fisica, istante t_1

VM: Esercizio 1 richiesta d

7 step:

P termina e libera le pagine fisiche 1 e 4
(non la 0, perché CR0 rimane necessaria)

indir. fisico	pagine allocate al tempo t_1
0	CP0 (= CR0)
1	
2	CQ1
3	PQ0
4	
5	DQ0
6	PR1
7	PR2

*3) memoria fisica,
istante t_1*

VM: Esercizio 1 richiesta d

8 step:

la exec Y da parte di Q non alloca il codice, perché CS0 risulta identica a CR0, ma alloca la pagina PS0 nella pagina fisica 1, già libera; vengono inoltre liberate le pagine fisiche 2 e 3

indir. fisico	pagine allocate al tempo t_1
0	CP0 (= CR0)
1	PS0
2	
3	
4	
5	DQ0
6	PR1
7	PR2

*3)memoria fisica,
istante t_1*

VM: Esercizio 1 richiesta d

9 step:

la nuova pagina DS0 viene allocata in pagina fisica 2, ma S raggiunge così il livello massimo di pagine residenti e quindi la successiva (DS1) deve essere allocata al posto di PS0

indir. fisico	pagine allocate al tempo t_1
0	CP0 (= CR0)
1	PS0 DS1
2	DS0
3	
4	
5	DQ0
6	PR1
7	PR2

*3) memoria fisica,
istante t_1*

VM: Esercizio 1 richiesta e

Si indichi il contenuto della tabella delle pagine della MMU all'istante di tempo t_1 completando la tabella 4. Si ipotizzi che le righe della tabella siano state allocate ordinatamente man mano che venivano allocate le pagine di memoria virtuale e che gli eventi di cui ai punti (c, d), influenzanti la MMU, partano da una situazione di tabella vergine, abbiano utilizzato le righe lasciate libere e che se è richiesta una nuova riga si utilizzi **sempre** la prima riga libera. **Si indichi** anche il valore assunto dal bit di validità di pagina (il valore 1 significa che la pagina è caricata). Il numero di righe nella tabella sotto non è significativo.

PID <i>indicare P Q R o S come pid oppure ns se la riga non è significativa</i>	NPV <i>utilizzare la notazione CP0/0 per indicare “segmento di codice di P numero 0 / pagina virtuale numero 0”, e similmente per le altre</i>	NPF	Bit di Validità

VM: Esercizio 1 richiesta e: Soluzione

Per permettere di interpretare il risultato riportato nella seguente tabella si indicano, oltre al contenuto della tabella all'istante finale, anche i contenuti precedenti.



PID <i>indicare P Q R o S come pid oppure ns se la riga non è significativa</i>	NPV <i>utilizzare la notazione CP0/0 per indicare "segmento di codice di P numero 0 / pagina virtuale numero 0", e similmente per le altre</i>	NPF	Bit di Validità
P _{ns} S	CP0 CS0 / 0	0	1
P _{ns} S	PP0 PP1 PS0 DS1 / 4	1	1
Q _{ns} S	CQ1 DS0 / 3	2	1
Q _{ns}	PQ0	3	0
P _{ns}	DP0	4	0
Q _{ns}	DQ0	5	0
R	CR0 / 0	0	1
R	PR1 / 6	6	1
R	PR2 / 5	7	1

VM: Esercizio 2

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 K byte, quella logica di 32 K byte e la pagina ha taglia di 4 K byte. Si chiede di svolgere i punti seguenti:

(a) **Si definisca** la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi



VM: Esercizio 2

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 K byte, quella logica di 32 K byte e la pagina ha taglia di 4 K byte. Si chiede di svolgere i punti seguenti:

(a) **Si definisca** la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi

NPF: 3 bit

spiazzamento fisico: 12 bit

NPL: 3 bit

spiazzamento logico: 12 bit

VM: Esercizio 2

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 K byte, quella logica di 32 K byte e la pagina ha taglia di 4 K byte. Si chiede di svolgere i punti seguenti:

(b) **Si inserisca** nella tabella a fianco la struttura in pagine della memoria virtuale di due programmi X e Y (mediante la notazione CX0 CX1 DX0 PX0 ... CY0 ...), sapendo che la dimensione iniziale dei segmenti di tali programmi è la seguente:

CX: 4 K CY:16K

DX:12 K DY: 4K

PX:4 K PY:4K

indir. virtuale	prog. X	prog. Y
0		
1		
2		
3		
4		
5		
6		
7		

VM: Esercizio 2

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 K byte, quella logica di 32 K byte e la pagina ha taglia di 4 K byte. Si chiede di svolgere i punti seguenti:

(b) **Si inserisca** nella tabella a fianco la struttura in pagine della memoria virtuale di due programmi X e Y (mediante la notazione CX0 CX1 DX0 PX0 ... CY0 ...), sapendo che la dimensione iniziale dei segmenti di tali programmi è la seguente:

CX: 4 K CY:16K

DX:12 K DY: 4K

PX:4 K PY:4K

indir. virtuale	prog. X	prog. Y
0	CX0	CY0
1	DX0	CY1
2	DX1	CY2
3	DX2	CY3
4		DY0
5		
6		
7	PX0	PY0

VM: Esercizio 2

C) Nel sistema vengono creati alcuni processi, indicati nel seguito con P, Q, R e S.

A pagina seguente sono indicate due serie di eventi; la prima serie termina all'istante t_0 , la seconda all'istante t_1 .

Si compilino le tabelle che descrivono i contenuti della memoria fisica e della MMU agli istanti t_0 e t_1 , utilizzando la notazione CP0 CP1 DP0 PP0 ... CQ0 ... per indicare le pagine virtuali dei processi; nelle tabelle della MMU si aggiunga anche il NPV effettivo con la notazione CP0/0 ecc ...; in tutte le tabelle si usi la notazione (CP0) ecc ... per indicare un dato non più valido e la notazione CP0 = CQ0 ecc ... per indicare che una pagina fisica contiene più di una pagina logica.

Si considerino valide le seguenti ipotesi relative al sistema:

- il lancio di un programma avviene caricando solamente la pagina di codice con l'istruzione di partenza e una sola pagina di pila
- il caricamento di pagine ulteriori è in Demand Paging (cioè le pagine si caricano su richiesta senza scaricare le precedenti fino al raggiungimento del numero massimo di pagine residenti)
- l'indirizzo (esadecimale) dell'istruzione di partenza di X è 0AAA
- l'indirizzo (esadecimale) dell'istruzione di partenza di Y è 39F2
- il numero di pagine residenti **R** vale **4**
- viene utilizzato l'algoritmo LRU (ove richiesto prima si dealloca una pagina di processo e poi si procede alla nuova assegnazione)
- in assenza di indicazioni esplicite relative all'accesso alle pagine, le pagine meno utilizzate in ogni processo sono quelle caricate da più tempo; inoltre la pagina di codice caricata più di recente è acceduta continuamente
- al momento di una "fork" viene duplicata solamente la pagina di pila caricata più recentemente, ma tutte le pagine virtuali del padre sono considerate condivise con il figlio
- dopo una "fork", se uno dei due processi padre o figlio scrive in una pagina condivisa, la nuova pagina fisica che viene allocata appartiene al processo che ha eseguito la scrittura
- l'allocazione delle pagine virtuali nelle pagine fisiche avviene **sempre** in sequenza, senza buchi, a partire dalla pagina fisica 0
- le righe della tabella della MMU vengono allocate ordinatamente man mano che vengono allocate le pagine di memoria virtuale
- gli eventi influenzanti la MMU partono da una situazione di tabella vergine
- se è richiesta una nuova riga di MMU, si utilizza **sempre** la prima riga libera

VM: Esercizio 2

(d) A un certo istante di tempo t_0 sono terminati, nell'ordine, gli eventi seguenti:

1. creazione del processo P e lancio del programma X ("fork" di P ed "exec" di X)
2. P accede a pagine nel seguente ordine: dati 1, dati 0, pila
3. P crea due pagine dati dinamiche tramite BRK
4. creazione del processo Q e lancio del programma Y ("fork" di Q ed "exec" di Y)
5. Q salta all'istruzione di indirizzo 2B35, poi accede alla pila
6. Q crea 2 pagine di pila

Si compilino le tabelle, al tempo t_0 .

memoria fisica	
indir. fisico	pagine allocate
0	
1	
2	
3	
4	
5	
6	
7	

MMU			
proc.	NP V	NP F	valid bit
P			
P			
P			
P			
Q			
Q			
Q			
Q			

VM: Esercizio 2

(d) A un certo istante di tempo t_0 sono terminati, nell'ordine, gli eventi seguenti:

1. creazione del processo P e lancio del programma X ("fork" di P ed "exec" di X)
2. P accede a pagine nel seguente ordine: dati 1, dati 0, pila
3. P crea due pagine dati dinamiche tramite BRK
4. creazione del processo Q e lancio del programma Y ("fork" di Q ed "exec" di Y)
5. Q salta all'istruzione di indirizzo 2B35, poi accede alla pila
6. Q crea 2 pagine di pila

Si compilino le tabelle, al tempo t_0 .

memoria fisica	
indir.fisico	pagine allocate
0	CP0
1	PP0
2	(DP1) DP4
3	(DP0) DP5
4	(CQ3) PQ2
5	PQ0
6	CQ2
7	PQ1

MMU			
proc.	NP V	NP F	valid bit
P	CP0/0	0	1
P	PP0/7	1	1
P	(DP1/2) DP4/5	2	1
P	(DP0/0) DP5/6	3	1
Q	(CQ3/3) PQ2/5	4	1
Q	PQ0/7	5	1
Q	CQ2/2	6	1
Q	PQ1/6	7	1
			0
			0
			0
			0

VM: Esercizio 2

A un certo istante di tempo $t_1 > t_0$ sono terminati, nell'ordine, gli eventi seguenti:

P termina ("exit" di P)

Q esegue una fork e crea il processo R

R esegue una fork e crea il processo S

Si compilino le tabelle sotto, al tempo t_1 .

memoria fisica	
indir. fisico	pagine allocate
0	
1	
2	
3	
4	
5	
6	
7	

MMU			
proc.	NP V	NP F	validbit
(P) R			
(P) R			
(P) R			
(P) R			
Q			
Q			
Q			
Q			
S			
S			
S			
S			

VM: Esercizio 2

A un certo istante di tempo $t_1 > t_0$ sono terminati, nell'ordine, gli eventi seguenti:

P termina ("exit" di P)

Q esegue una fork e crea il processo R

R esegue una fork e crea il processo S

Si compilino le tabelle sotto, al tempo t_1 .

memoria fisica	
indir. fisico	pagine allocate
0	(CP PR2 0)
1	(PP PS2 0)
2	(DP (DP4) 1)
3	(DP (DP5) 0)
4	(CQ PQ2 3)
5	PQ0 = PR0 = PS0
6	CQ2 = CR2 = CS2
7	PQ1 = PR1 = PS1

MMU				
proc.	NP V		NP F	validbit
(P) R	(CP0/ 0)	PR2/5	(0) 0	(1)(0)1
(P) R	(PP0/ 7)	PR0/7	(1) 5	(1)(0)1
(P) R	(DP1/ 2)	(DP4/5) CR2/2	(2) 6	(1)(0)1
(P) R	(DP0/ 0)	(DP5/6) PR1/6	(3) 7	(1)(0)1
Q	(CQ3/ 3)	PQ2/5	4	1
Q	PQ0/7		5	1
Q	CQ2/2		6	1
Q	PQ1/6		7	1
S	PS2/5		1	(0) 1
S	PS0/7		5	(0) 1
S	CS2/2		6	(0) 1
S	PS1/6		7	(0) 1

VM: Esercizio 3

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica è di 64 Kbyte, l'indirizzo logico è di 16 bit, la dimensione di pagina è di 4 Kbyte. Si svolgano i punti seguenti:

Si definisca la struttura dell'indirizzo fisico dando la lunghezza in bit dei campi costituenti:

NPF: 4 bit Spiazzamento fisico: 12 bit



SOLUZIONE:

NPF: 4 bit Spiazzamento fisico: 12 bit

VM: Esercizio 3

b) I tre programmi PGP, PGQ e PGR, che verranno eseguiti dai tre processi P, Q e R, rispettivamente, hanno la struttura di segmentazione iniziale seguente (e condividono il segmento COND):

CP: 8 K	DP: 4 K	PP: 8 K	COND: 4 K
CQ: 12 K	DQ: 4 K	PQ: 4 K	COND: 4 K
CR: 8 K	DR: 4 K	PR: 4 K	COND: 4 K

La dimensione complessiva dello spazio di indirizzi di ogni processo è di 64 K. Il segmento pila di ogni processo inizia dal fondo di tale spazio e cresce verso l'indirizzo iniziale. Onde permettere la crescita dello Heap (dati dinamici, funzione `malloc`), nei processi P, Q e R il segmento condiviso COND è allocato lasciando 2, 1 e 2 pagine libere dopo i segmenti dati DP, DQ e DR, rispettivamente.

■■■■

VM: Esercizio 3

Si definisca in tabella 1A il significato delle varie pagine di memoria logica, tramite la notazione usuale: CP0, CP1, DP0, PP0, ..., CQ0, ..., COND0, ... (ecc).

CP: 8 K	DP: 4 K	PP: 8 K	COND: 4 K
CQ: 12 K	DQ: 4 K	PQ: 4 K	COND: 4 K
CR: 8 K	DR: 4 K	PR: 4 K	COND: 4 K

Indirizzo di pagina virtuale	Processo P	Processo Q	Processo R
0	CP0	CQ0	CR0
1	CP1	CQ1	CR1
2	DP0	CQ2	DR0
3		DQ0	
4		DQ1	
5	COND0	COND0	COND0
6			
7			
8			
9			
A			
B	PP4		
C	PP3		
D	PP2		
E	PP1		
F	PP0	PQ0	PR0