HỆ ĐIỀU HÀNH
(Operation System)

Th.S Nguyễn Hoàng Thành
Email: thanhnh@ptithcm.edu.vn
Tel: 0909 682 711

QUẢN LÝ TIẾN TRÌNH

- **Các khái niệm về tiến trình**
- **Điều phối các tiến trình**
- **Liên lạc giữa các tiến trình**
- **Đồng bộ các tiến trình**
- **Tính trạng tắc nghẽn (deadlock)**

Tiến trình (Process)

- Tiến trình là một chương trình đang xử lý
- Mỗi tiến trình có một không gian địa chỉ, một con trỏ lệnh, một tập các thanh ghi và stack riêng.
- Tiến trình có thể cần đến một số tài nguyên như CPU, bộ nhớ chính, các tập tin và thiết bị nhập/xuất.
- Hệ điều hành sử dụng bộ điều phối (scheduler) để quyết định thời điểm cần dừng hoạt động của tiến trình đang xử lý và lựa chọn tiến trình tiếp theo cần thực hiện.
- Trong hệ thống có những tiến trình của hệ điều hành và tiến trình của người dùng.

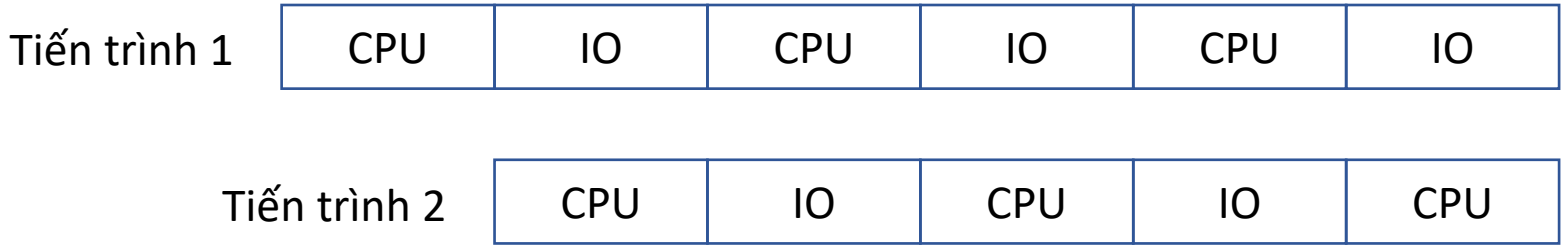
Mục đích cho nhiều tiến trình hoạt động đồng thời

- Tăng hiệu suất sử dụng CPU (tăng mức độ đa chương)
- Tăng mức độ đa nhiệm
- Tăng tốc độ xử lý

Tăng hiệu suất sử dụng CPU (tăng mức độ đa chương)

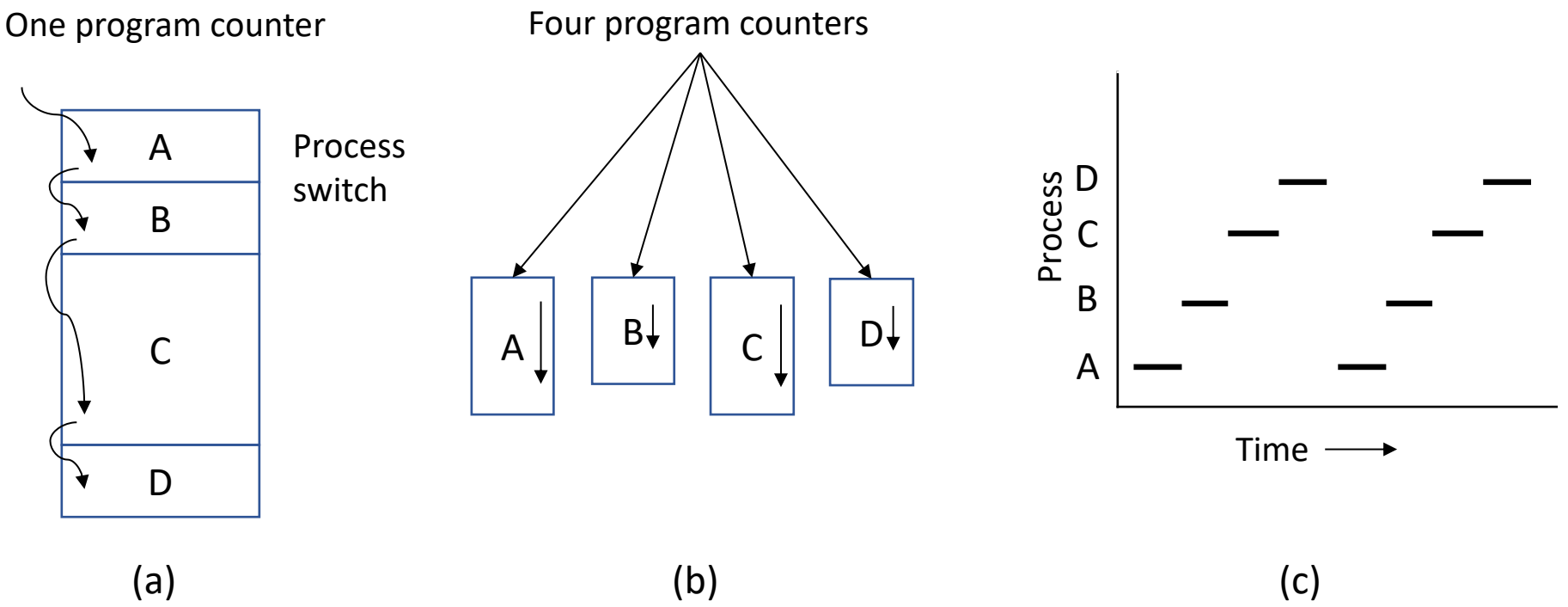
- Phần lớn các tiến trình khi thi hành đều trải qua nhiều chu kỳ xử lý (sử dụng CPU) và chu kỳ nhập xuất (sử dụng các thiết bị nhập xuất) xen kẽ như sau:

- Nếu chỉ có 1 tiến trình duy nhất trong hệ thống, thì vào các chu kỳ IO của tiến trình, CPU sẽ hoàn toàn nhàn rỗi. Ý tưởng tăng cường số lượng tiến trình trong hệ thống là để tận dụng CPU: nếu tiến trình 1 xử lý IO, thì hệ điều hành có thể sử dụng CPU để thực hiện tiến trình 2...



Tăng mức độ đa nhiệm

- Cho mỗi tiến trình thực thi luân phiên trong một thời gian rất ngắn, tạo cảm giác là hệ thống có nhiều tiến trình thực thi đồng thời.



Tăng tốc độ xử lý

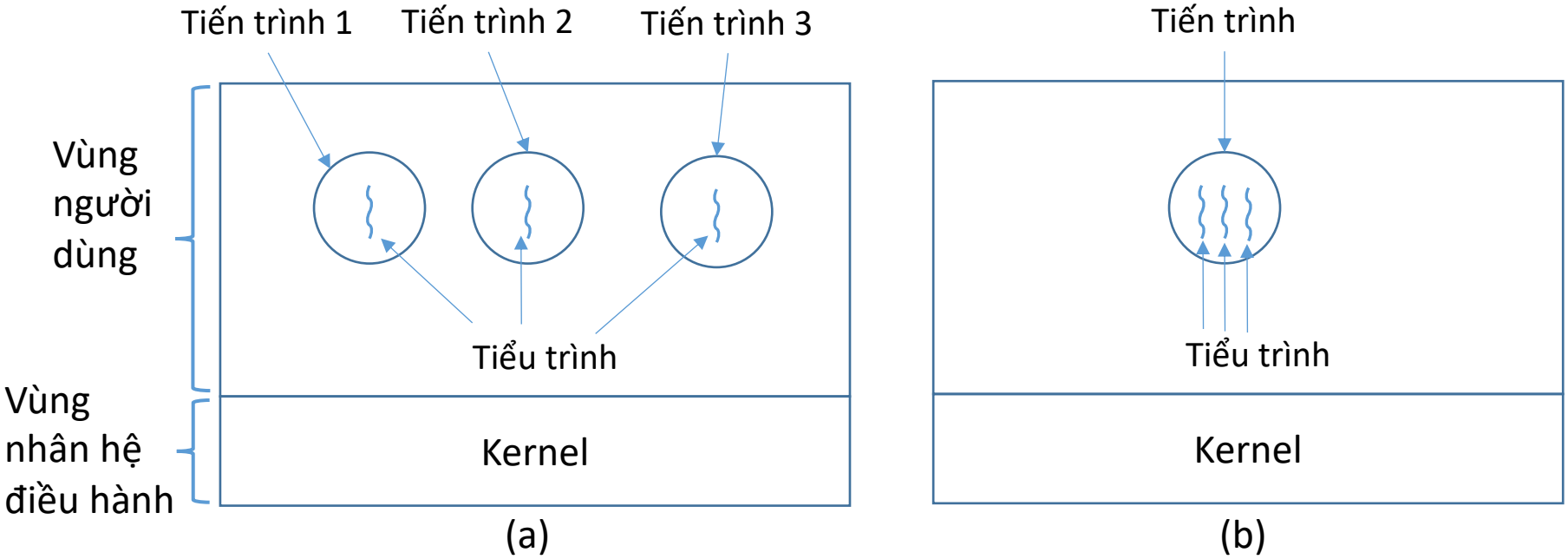
- Một số bài toán có thể xử lý song song nếu được xây dựng thành nhiều đơn thể hoạt động đồng thời thì sẽ tiết kiệm được thời gian xử lý.
- Ví dụ xét bài toán tính giá trị biểu thức $kq = a \times b + c \times d$. Nếu tiến hành tính đồng thời $(a \times b)$ và $(c \times d)$ thì thời gian xử lý sẽ ngắn hơn là thực hiện tuần tự.

Tiểu trình (thread)

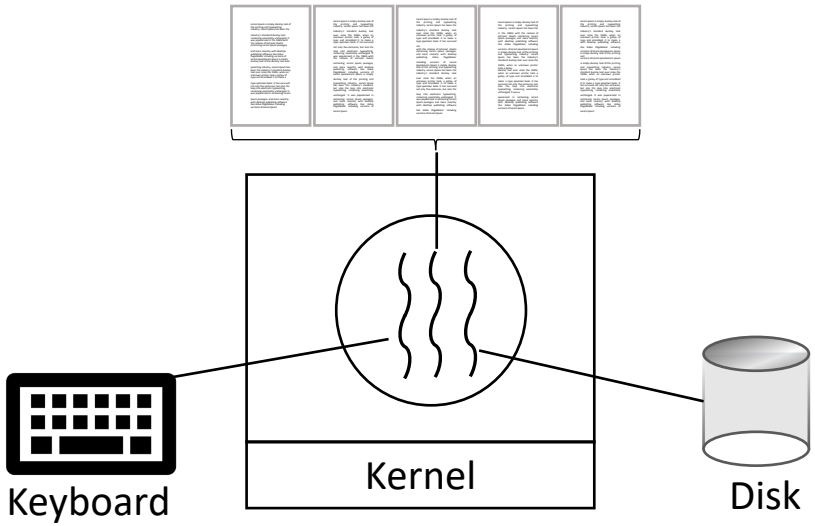
- Một tiến trình có thể tạo nhiều tiểu trình.
- Mỗi tiểu trình thực hiện một chức năng nào đó và thực thi đồng thời cũng bằng cách chia sẻ CPU.
- Các tiểu trình trong cùng một tiến trình dùng chung không gian địa chỉ tiến trình nhưng có con trỏ lệnh, tập các thanh ghi và stack riêng.
- Một tiểu trình cũng có thể tạo lập các tiểu trình con, và nhận các trạng thái khác nhau như một tiến trình.

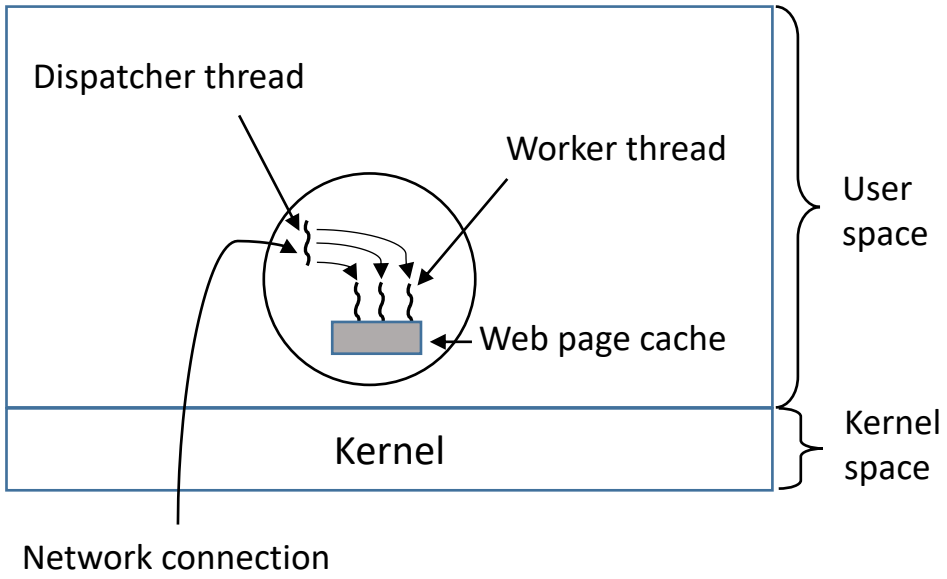
Liên lạc giữa các tiểu trình

- Các tiến trình chỉ có thể liên lạc với nhau thông qua các cơ chế do hệ điều hành cung cấp.
- Các tiểu trình liên lạc với nhau dễ dàng thông qua các biến toàn cục của tiến trình.
- Các tiểu trình có thể do hệ điều hành quản lý hoặc hệ điều hành và tiến trình cùng phối hợp quản lý



Ví dụ về tiểu trình






```
while (TRUE) {
  get_next_request(&buf);
  handoff_work(&buf);
}
```

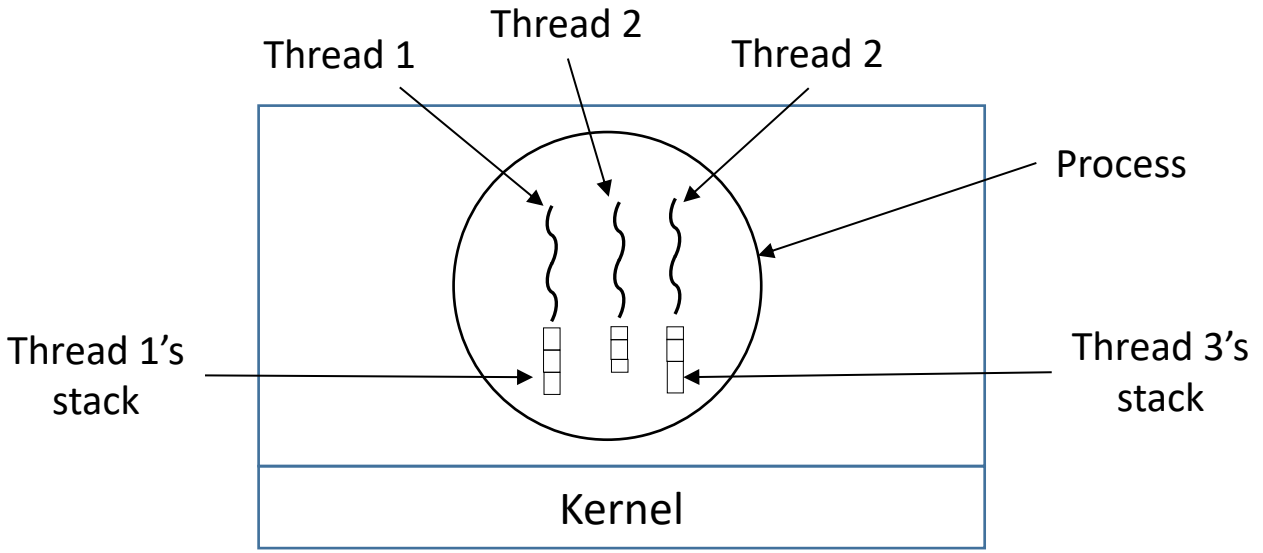
(a)

```
while (TRUE) {
  wait_for_work(&buf);
  look_for_page_in_cache(&buf, &page);
  if (page_not_in_cache(&page)
    read_page_from_disk(&buf, &page);
  return_page(&page);
}
```

(b)

Ví dụ về tiểu trình

- Một process có ba thread, mỗi thread sẽ có stack riêng

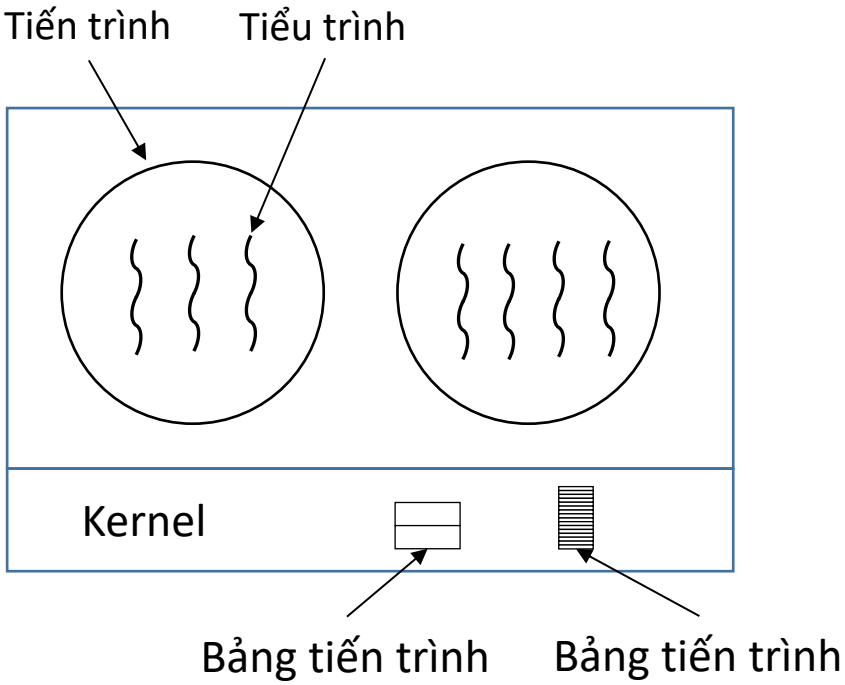


Per process items	**Per thread items**
Address space Global varibales Open files Child processes Pending alarms Signals and signal handlers Accounting information	Program counter Registers Stack State

Cài đặt tiểu trình (Threads)

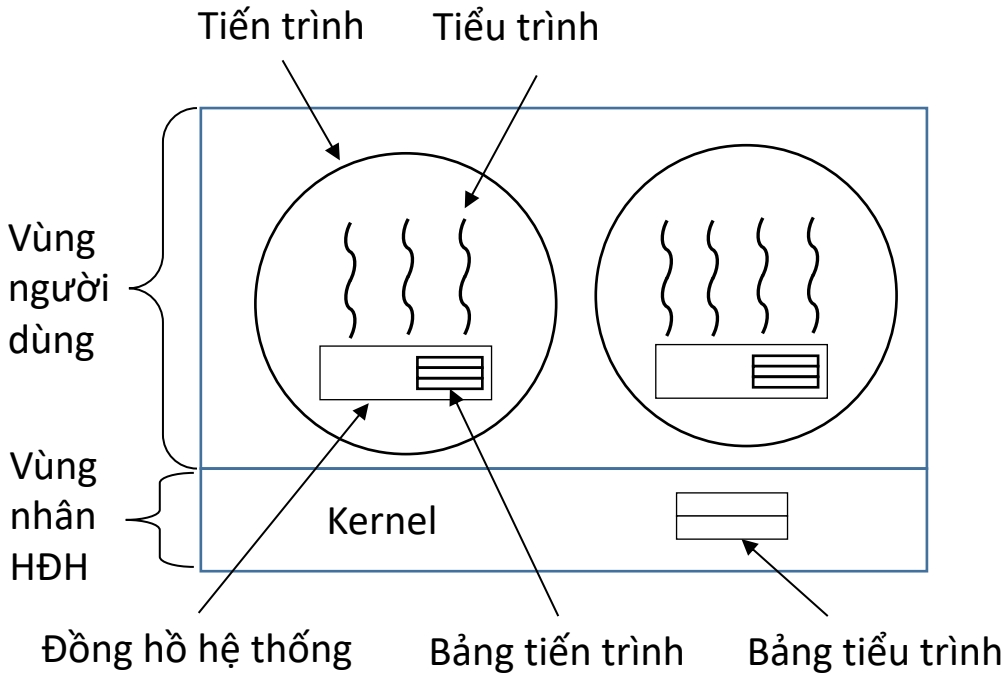
- Cài đặt trong Kernel-space
- Cài đặt trong User-space
- Cài đặt trong Kernel-space và User-space

Cài đặt tiểu trình trong Kernel-space



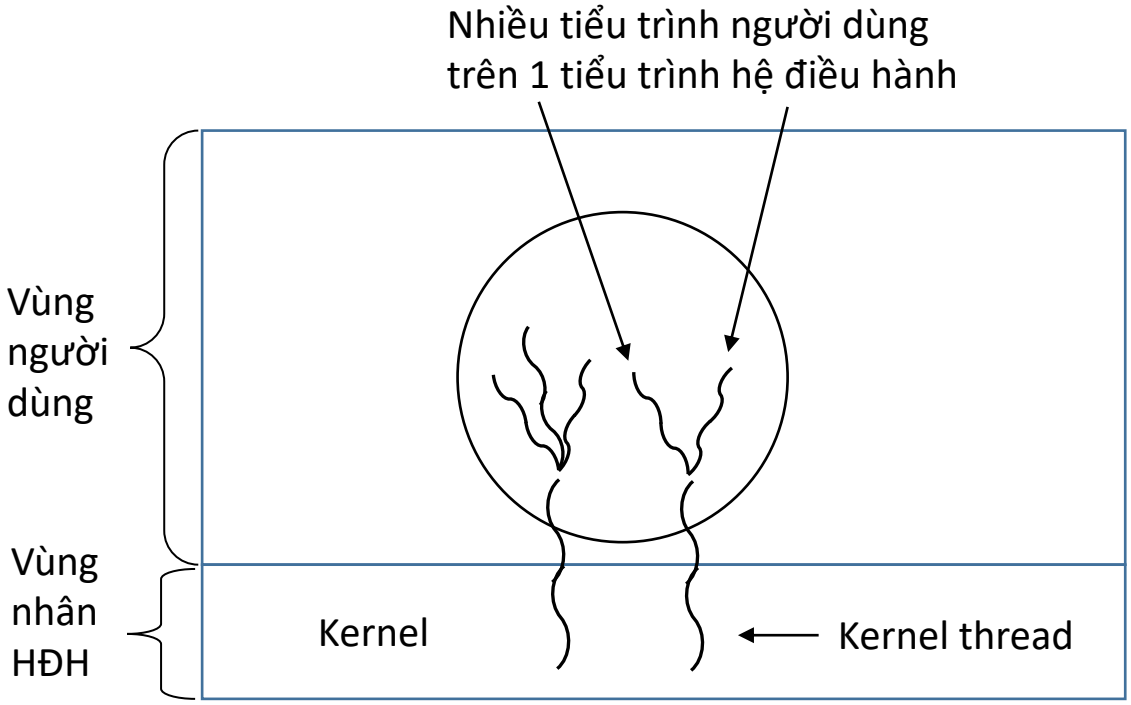
- Bảng quản lý thread lưu trữ ở phần kernel.
- Việc điều phối các thread là do hệ điều hành chịu trách nhiệm.

Cài đặt tiểu trình trong User-space



- Bảng quản lý thread lưu trữ ở phần user-space.
- Việc điều phối các thread là do tiến trình chịu trách nhiệm.

Cài đặt trong Kernel-space và User-space



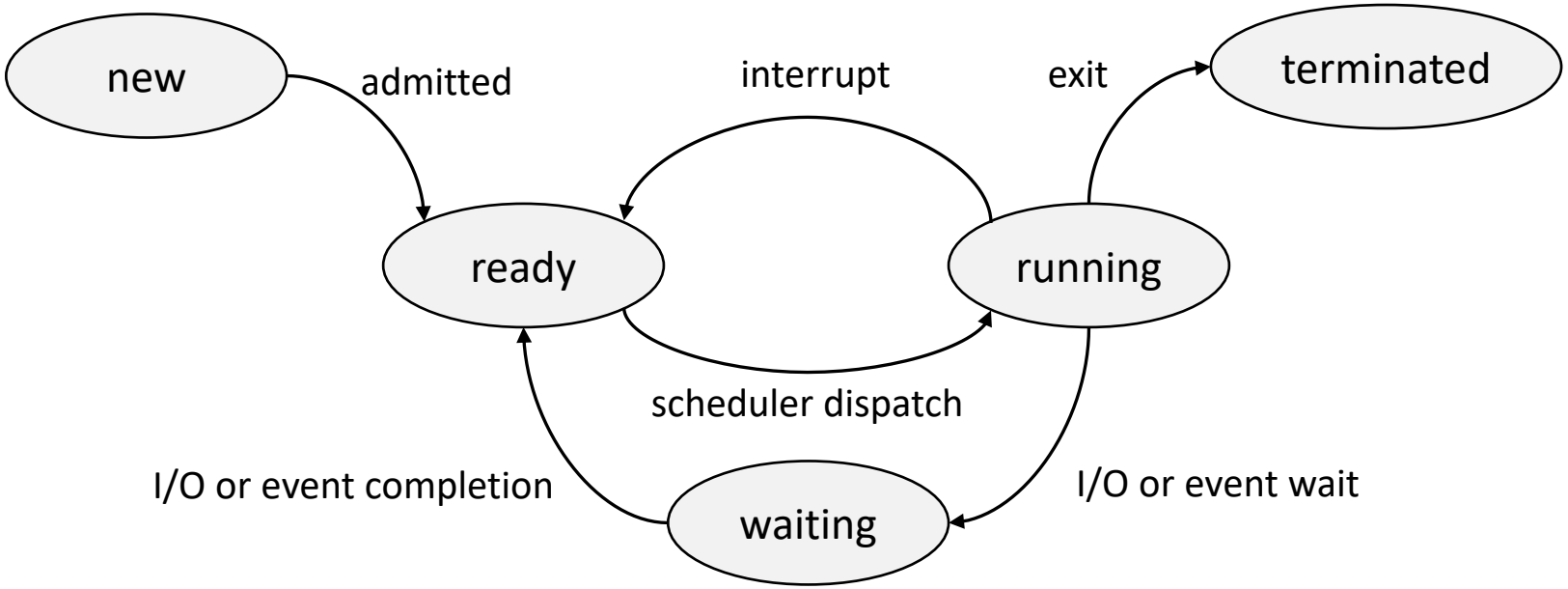
- Một thread của hệ điều hành quản lý một số thread của tiến trình

Ví dụ về điều phối tiểu trình

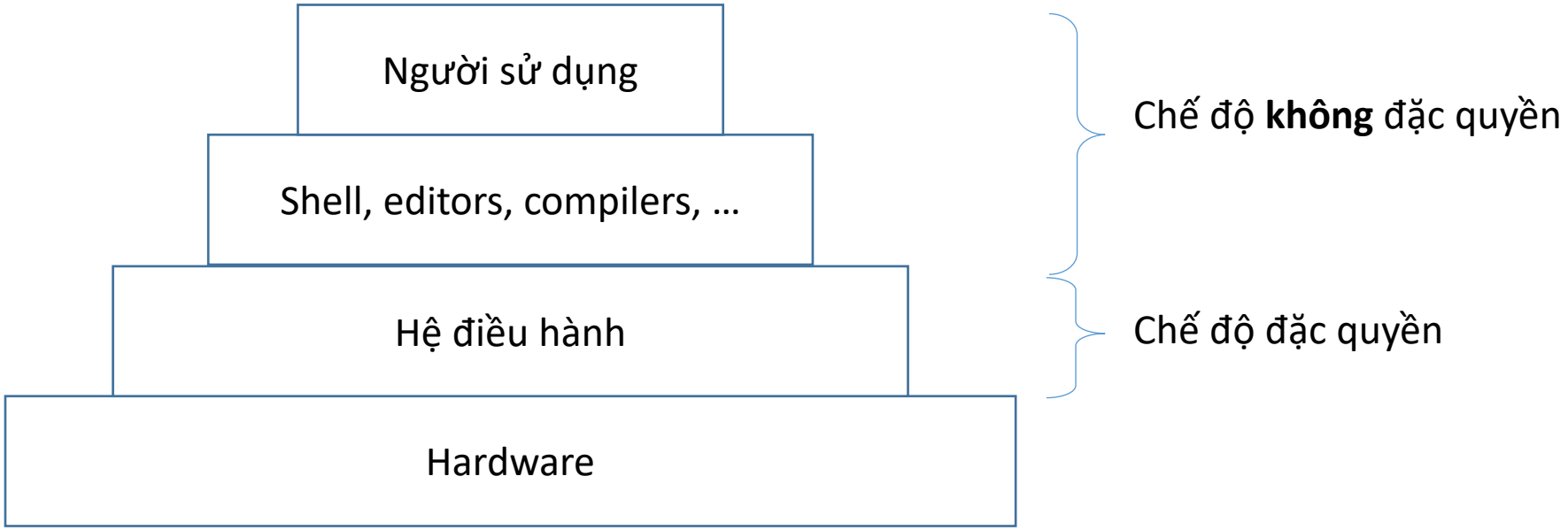
- quantum của process=50 msec
- quantum của thread=5 msec
- Tiến trình A có 3 thread, tiến trình B có 4 thread.

Các trạng thái của tiến trình

- **Mới tạo:** tiến trình đang được tạo lập.
- **Running:** các chỉ thị của tiến trình đang được xử lý.
- **Blocked:** tiến trình chờ được cấp phát một tài nguyên, hay chờ một sự kiện xảy ra.
- **Ready:** tiến trình chờ được cấp phát CPU để xử lý.
- **Kết thúc:** tiến trình hoàn tất xử lý.



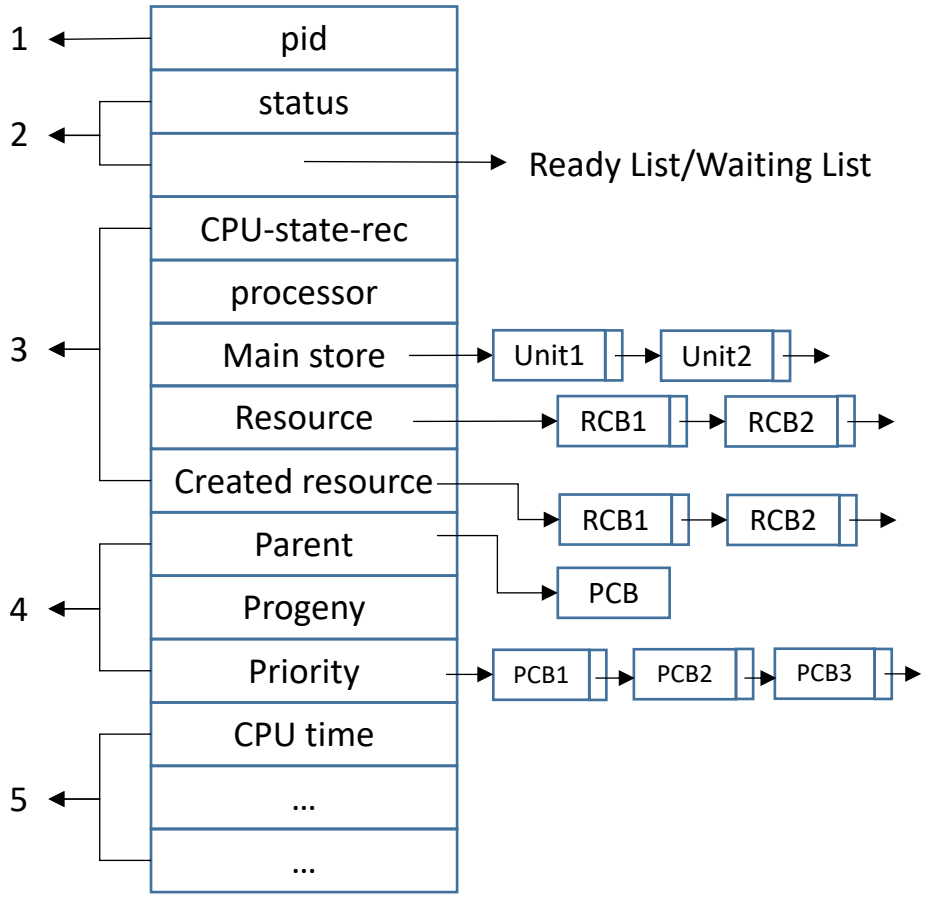
Chế độ xử lý của tiến trình



Hai chế độ xử lý

Cấu trúc dữ liệu khối quản lý tiến trình

- Khối quản lý tiến trình (PCB): là một vùng nhớ lưu trữ các thông tin mô tả cho tiến trình:
  - **Định danh của tiến trình (1)**
  - **Trạng thái tiến trình (2)**
  - **Ngữ cảnh của tiến trình (3)**
    - *Trạng thái CPU, Bộ xử lý, Bộ nhớ chính,Tài nguyên sử dụng,Tài nguyên tạo lập*
  - **Thông tin giao tiếp (4)**
    - *Tiến trình cha,Tiến trình con, Độ ưu tiên*
  - **Thông tin thống kê (5)**



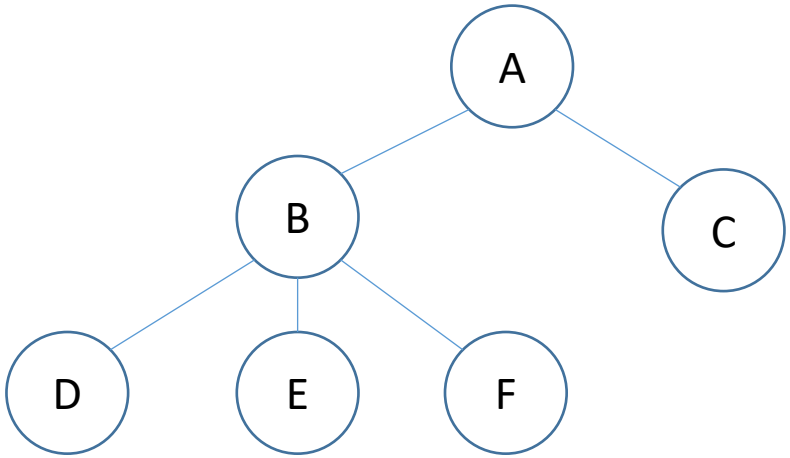
Khối mô tả tiến trình

Thao tác trên tiến trình

- Tạo lập tiến trình (create)
- Kết thúc tiến trình (destroy)
- Tạm dừng tiến trình (suspend)
- Tái kích hoạt tiến trình (resume)
- Thay đổi độ ưu tiên tiến trình (change priority)

Tạo lập tiến trình

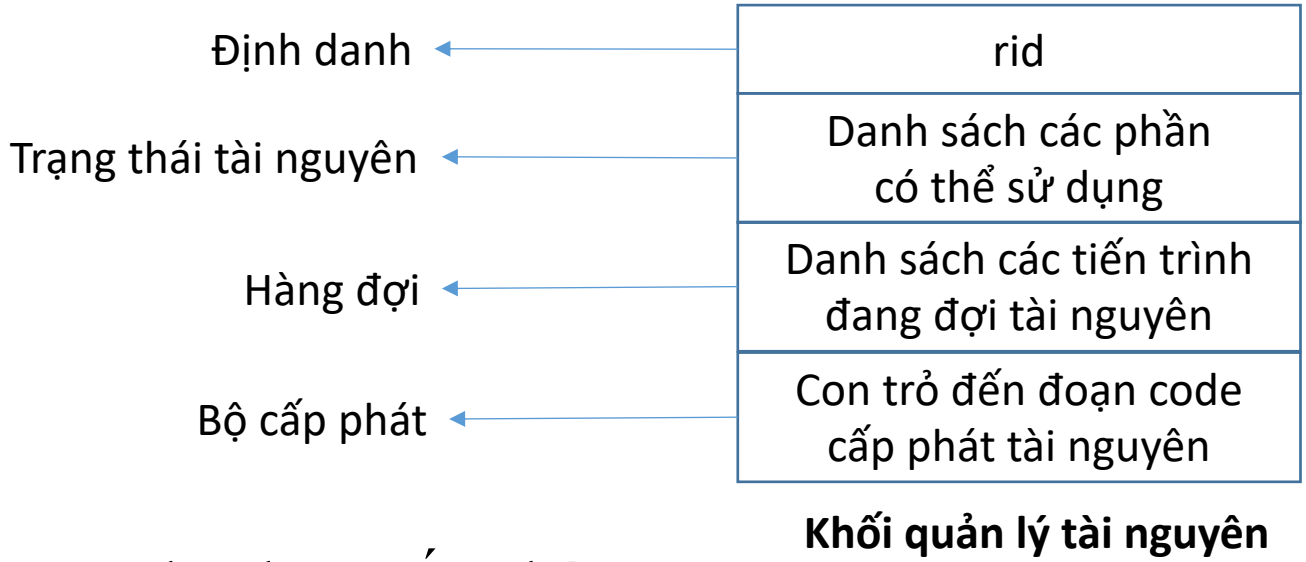
- Định danh cho tiến trình mới phát sinh
- Đưa tiến trình vào danh sách quản lý của hệ thống
- Xác định độ ưu tiên cho tiến trình
- Tạo PCB cho tiến trình
- Cấp phát các tài nguyên ban đầu cho tiến trình



Kết thúc tiến trình

- Thu hồi các tài nguyên hệ thống đã cấp phát cho tiến trình
- Hủy tiến trình khỏi tất cả các danh sách quản lý của hệ thống
- Hủy bỏ PCB của tiến trình

Cấp phát tài nguyên cho tiến trình



Khối quản lý tài nguyên

- Các mục tiêu của kỹ thuật cấp phát :
  - Bảo đảm một số lượng hợp lệ các tiến trình truy xuất đồng thời đến các tài nguyên không chia sẻ được.
  - Cấp phát tài nguyên cho tiến trình có yêu cầu trong một khoảng thời gian trì hoãn có thể chấp nhận được.
  - Tối ưu hóa sự sử dụng tài nguyên.

Điều phối tiến trình

- Hệ điều hành điều phối tiến trình thông qua bộ điều phối (scheduler) và bộ phân phối (dispatcher).
- Bộ điều phối sử dụng một giải thuật thích hợp để lựa chọn tiến trình được xử lý tiếp theo.
- Bộ phân phối chịu trách nhiệm cập nhật ngữ cảnh của tiến trình bị tạm ngưng và trao CPU cho tiến trình được chọn bởi bộ điều phối để tiến trình thực thi.
- **Mục tiêu điều phối**
  - Sự công bằng (Fairness)
  - Tính hiệu qủa (Efficiency)
  - Thời gian đáp ứng hợp lý (Response time)
  - Thời gian lưu lại trong hệ thống (TurnaroundTime)
  - Thông lượng tối đa (Throughput)

Các đặc điểm của tiến trình

- Tính hướng xuất / nhập của tiến trình (I/O-boundedness):
  - Nhiều lượt sử dụng CPU, mỗi lượt dùng thời gian ngắn.
- Tính hướng xử lý của tiến trình (CPU-boundedness):
  - Ít lượt sử dụng CPU, mỗi lượt dùng thời gian dài.
- Tiến trình tương tác **hay** xử lý theo lô
- Độ ưu tiên của tiến trình
- Thời gian đã sử dụng CPU của tiến trình
- Thời gian còn lại tiến trình cần để hoàn tất

Các nguyên lý điều phối

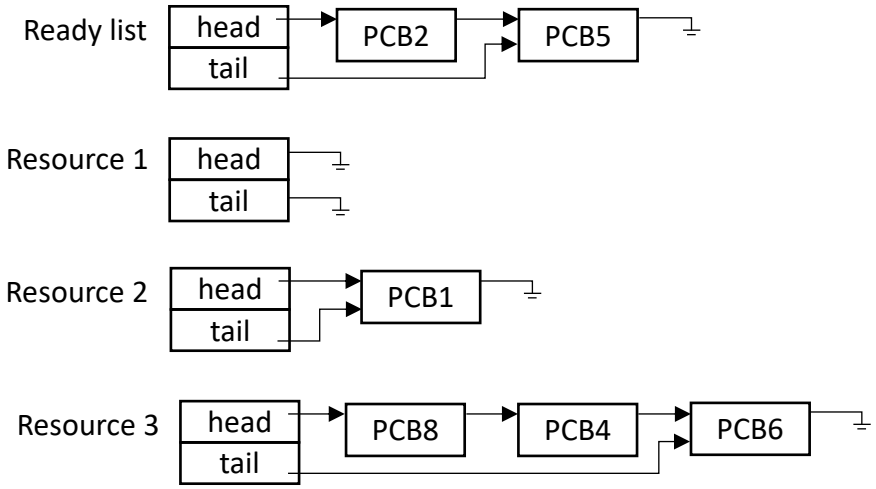
- Điều phối độc quyền (preemptive)
  - độc chiếm CPU
  - không thích hợp với các hệ thống nhiều người dùng
- Điều phối không độc quyền (nopreemptive)
  - tránh được tình trạng một tiến trình độc chiếm CPU
  - có thể dẫn đến các mâu thuẫn trong truy xuất-> cần phương pháp đồng bộ hóa thích hợp để giải quyết.
  - phức tạp trong việc phân định độ ưu tiên
  - phát sinh thêm chi phí khi chuyển đổi CPU qua lại giữa các tiến trình

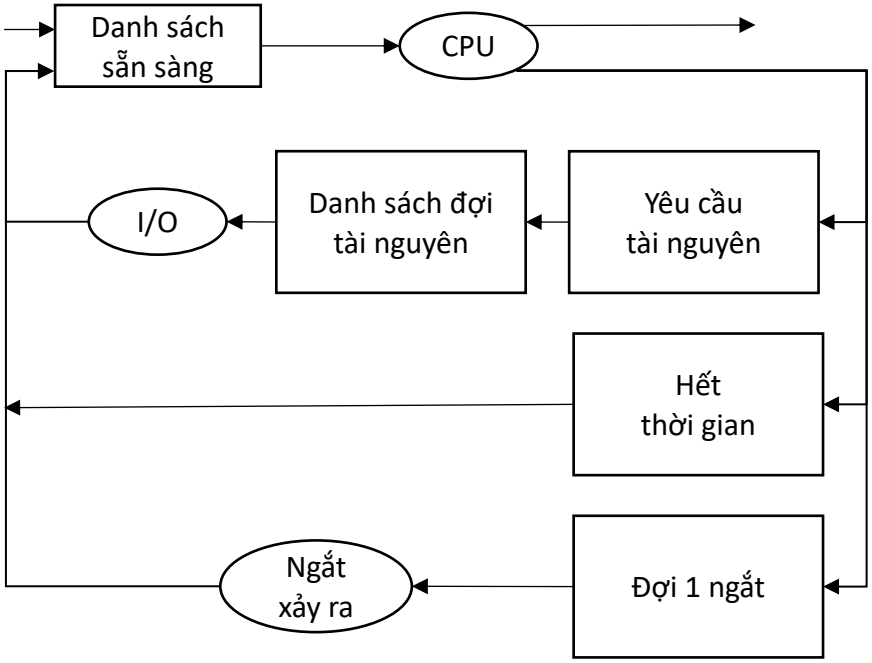
Thời điểm thực hiện điều phối

- running -> blocked
  - ví dụ chờ một thao tác nhập xuất hay chờ một tiến trình con kết thúc…
- running -> ready
  - ví dụ xảy ra một ngắt.
- blocked -> ready
  - ví dụ một thao tác nhập/xuất hoàn tất.
- Tiến trình kết thúc.
- Tiến trình có độ ưu tiên cao hơn xuất hiện
  - chỉ áp dụng đối với điều phối không độc quyền

Tổ chức điều phối - Các danh sách điều phối

- danh sách tác vụ (job list)
- danh sách sẵn sàng (ready list)
- danh sách chờ đợi (waiting list)





Các loại điều phối

- **Điều phối tác vụ (job scheduling)**
  - chọn tác vụ nào được đưa vào bộ nhớ chính để thực hiện
  - quyết định mức độ đa chương
  - tần suất hoạt động thấp
- **Điều phối tiến trình (process scheduling)**
  - Chọn một tiến trình ở trạng thái sẵn sàng (đã được nạp vào bộ nhớ chính, và có đủ tài nguyên để hoạt động) để cấp phát CPU cho tiến trình đó thực hiện
  - có tần suất hoạt động cao(1 lần/100 ms).
  - sử dụng các thuật toán tốt nhất

Các thuật toán điều phối

- Thuật toán FIFO
- Thuật toán phân phối xoay vòng (Round Robin)
- Thuật toán độ ưu tiên
- Thuật toán công việc ngắn nhất (Shortest-job-first SJF)
- Thuật toán nhiều mức độ ưu tiên
- Chiến lược điều phối xổ số (Lottery)

Thuật toán FIFO

Ready List

Điều phối theo nguyên tắc độc quyền

Tiến trình	Thời điểm vào RL	Thời gian xử lý
P_1	0	24
P_2	1	3
P_3	2	3

P_1	P_2	P_3
0	24	27
		30

- Thời gian chờ đợi được xử lý là 0 đối với P1, (24 -1) với P2 và (27-2) với P3.
- Thời gian chờ trung bình là $\frac{0+23+25}{3} = 16$ miliseconds.

Thuật toán FIFO

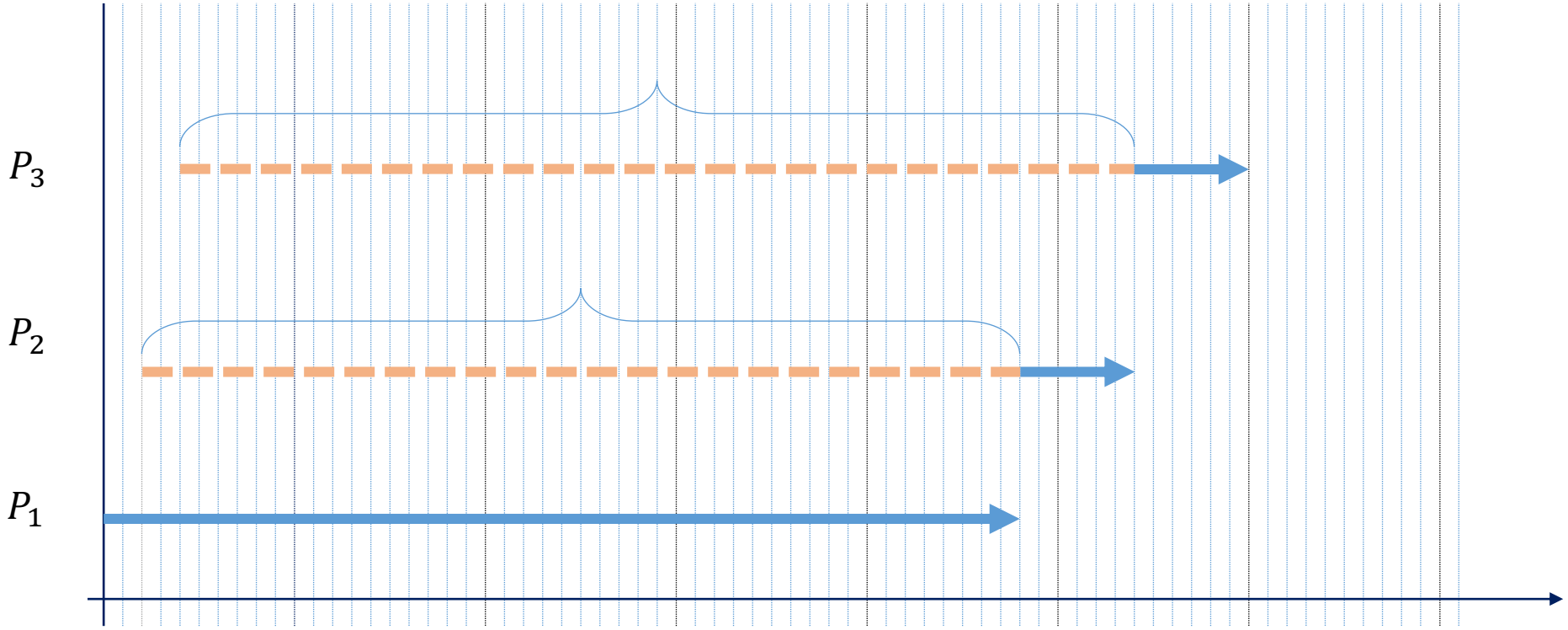
Tiến trình	Thời điểm vào RL	Thời gian xử lý
P_1	0	24
P_2	1	3
P_3	2	3

$$wait_{P_1} = 0$$

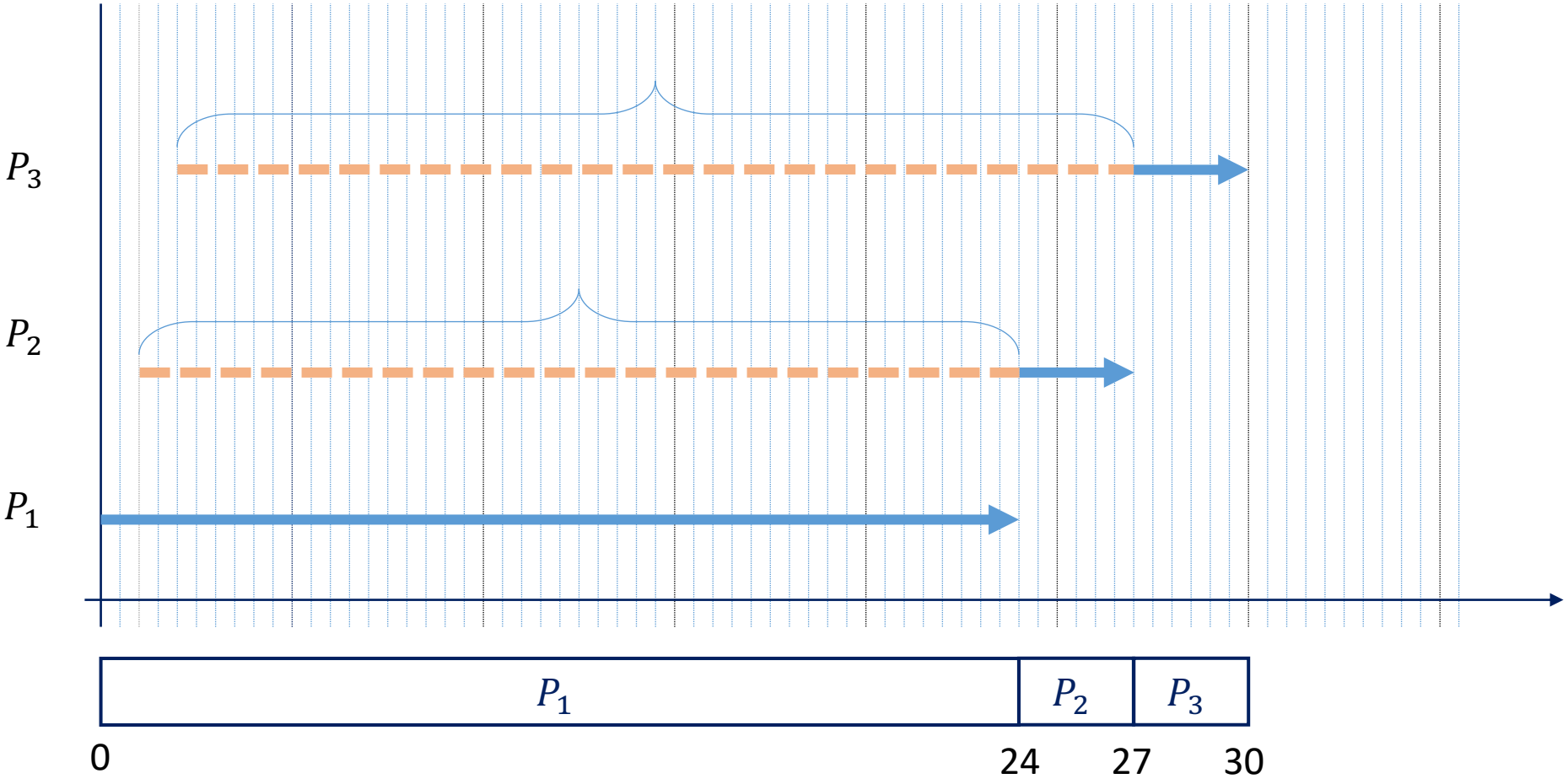
$$wait_{P_2} = 24 - 1 = 23$$

$$wait_{P_3} = (24 + 3) - 2 = 25$$

$$wait_{avg} = (0 + 23 + 25)/3 = 16$$



Thuật toán FIFO



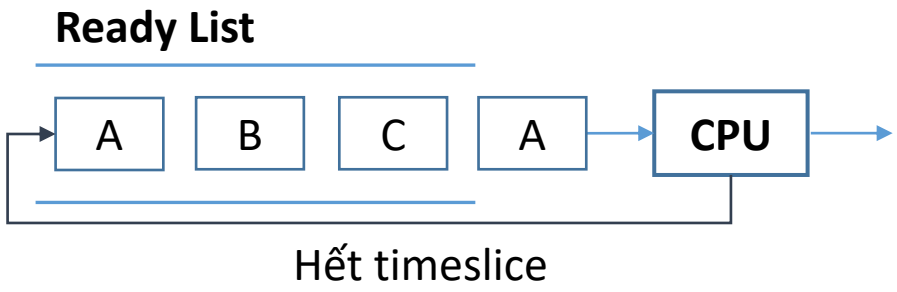
Thuật toán FIFO

Tiến trình	Thời điểm vào RL (1)	Thời gian xử lý (2)	Completion Time (3)	Turn around time (4)=(3)-(1)	Thời gian chờ (5)=(4)-(2)
P_1	0	24	24	24-0=24	24-24=0
P_2	1	3	27	27-1=26	26-3=23
P_3	2	3	30	30-2=28	28-3=25

Turn Around Time: thời gian kể từ lúc đưa vào (submission) đến lúc kết thúc

Thời gian chờ trung bình là $\frac{0+23+25}{3} = 16$ miliseconds.

Thuật toán phân phối xoay vòng (Round Robin)



Tiến trình	Thời điểm vào RL	Thời gian xử lý
P_1	0	24
P_2	1	3
P_3	2	3

P_1	P_2	P_3	P_1	P_1	P_1	P_1	P_1	
0	4	7	10	14	18	22	26	30

- Thời gian chờ đợi trung bình sẽ là $\frac{(6+3+5)}{3} = 4.66$ milisecondes.

Thuật toán phân phối xoay vòng (Round Robin)

Tiến trình	Thời điểm vào RL	Thời gian xử lý
P_1	0	24
P_2	1	3
P_3	2	3

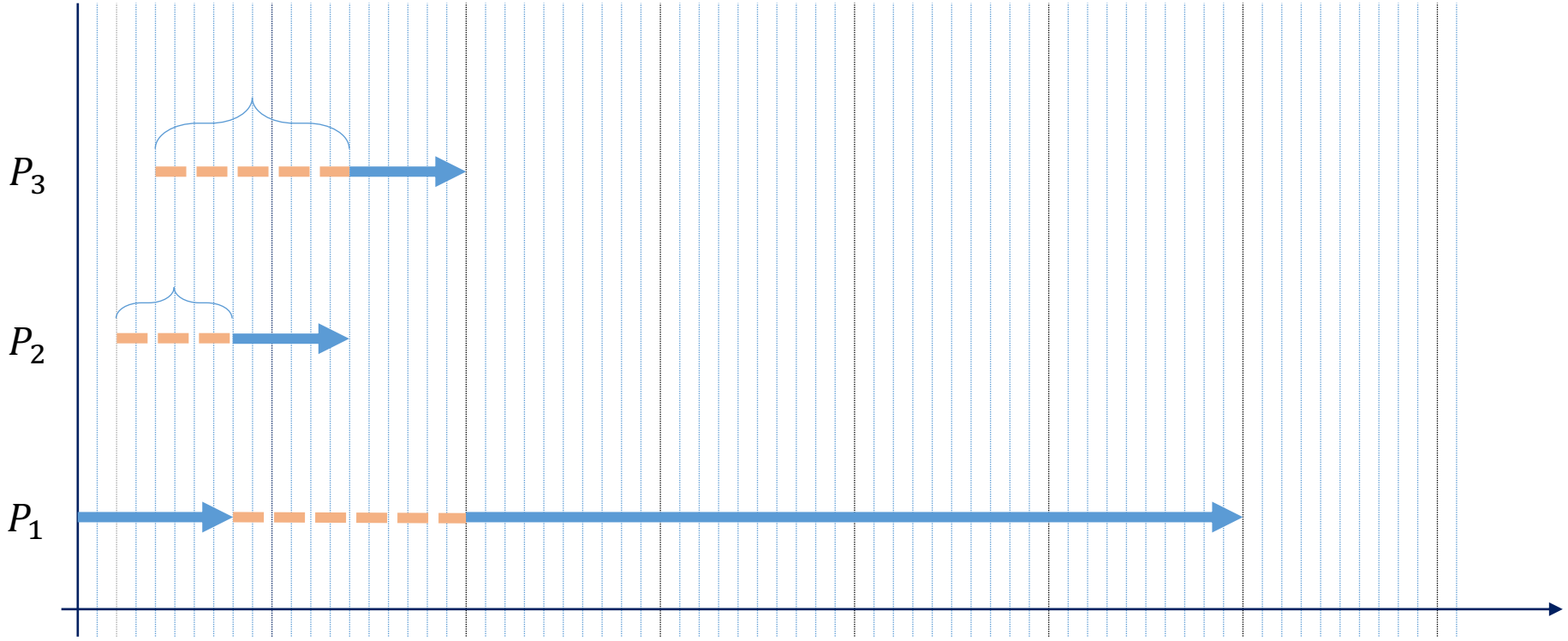
$$wait_{P_1} = 3 + 3 = 6$$

$$wait_{P_2} = 3$$

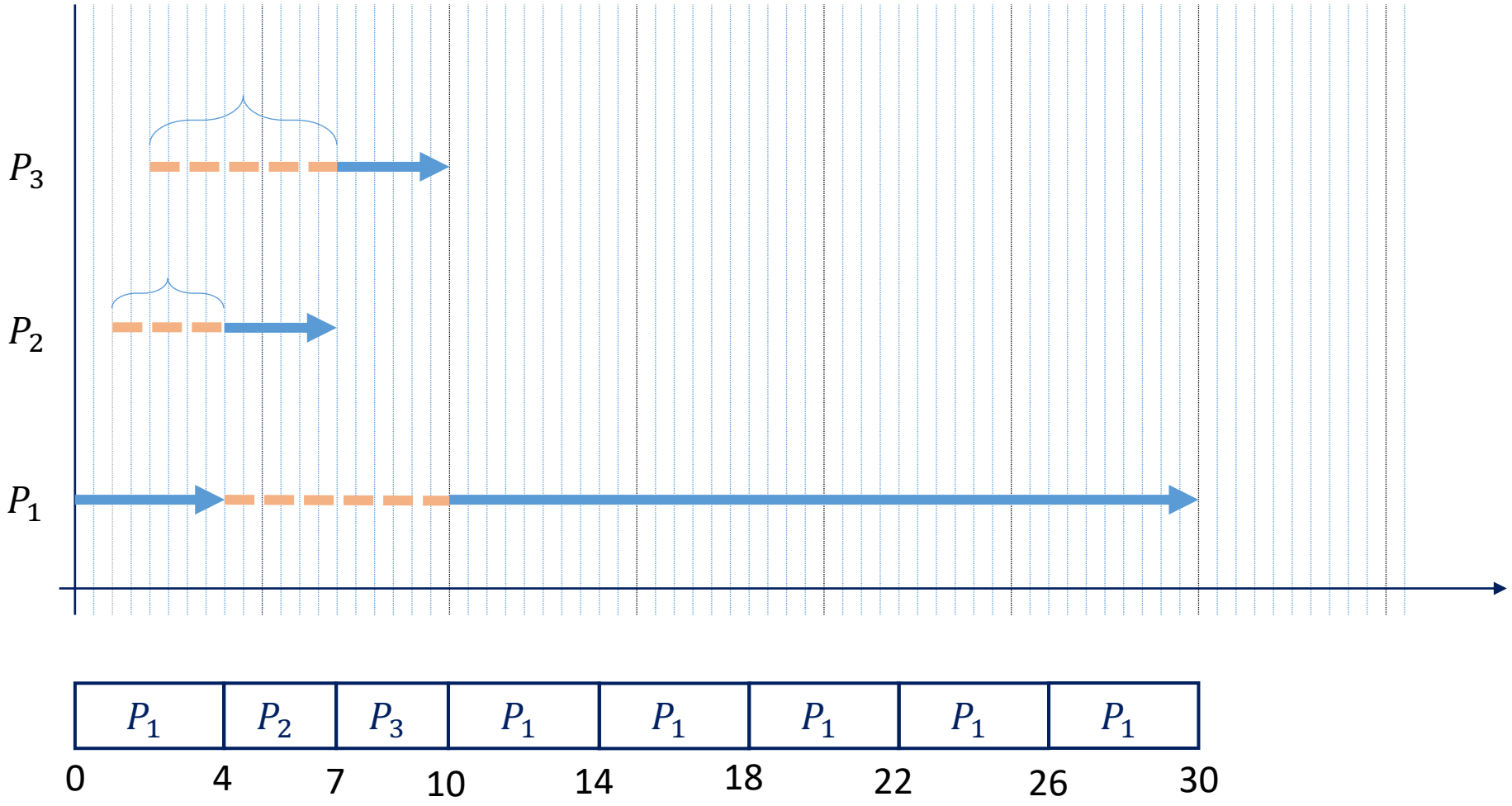
$$wait_{P_3} = 2 + 3 = 5$$

$$wait_{avg} = (6 + 3 + 5)/3 = 14/3$$

quantum là 4 miliseconds



Thuật toán phân phối xoay vòng (Round Robin)

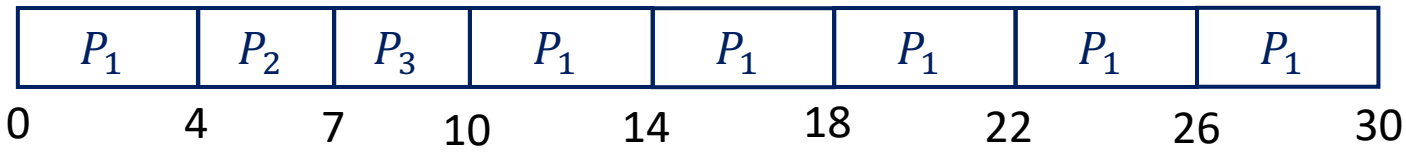


Thuật toán phân phối xoay vòng (Round Robin)

Tiến trình	Thời điểm vào RL (1)	Thời gian xử lý (2)	Complete time (3)	Turn around time (4)=(3)-(1)	Thời gian chờ (5)=(4)-(2)
P_1	0	24	30	30-0=30	30-24=6
P_2	1	3	7	7-1=6	6-3=3
P_3	2	3	10	10-2=8	8-3=5

Turn Around Time: thời gian kể từ lúc đưa vào (submission) đến lúc kết thúc

Thời gian chờ trung bình là $\frac{6+3+5}{3} = \frac{14}{3}$ miliseconds.



Thuật toán độ ưu tiên

- Độ ưu tiên $P_2 > P_3 > P_1$

Tiến trình	Thời điểm vào RL	Độ ưu tiên	Thời gian xử lý
P_1	0	3	24
P_2	1	1	3
P_3	2	2	3

Thuật giải độ ưu tiên độc quyền

P_1	P_2	P_3
0	24	27
		30

Thời gian chờ đợi trung bình sẽ là:
(0+23+25)/3=16 miliseconds

Thuật giải độ ưu tiên không độc quyền

P_1	P_2	P_3	P_1	
0	1	4	7	30

Thời gian chờ đợi trung bình sẽ là:
(6+0+2)/3=2.7 miliseconds

Thuật toán độ ưu tiên

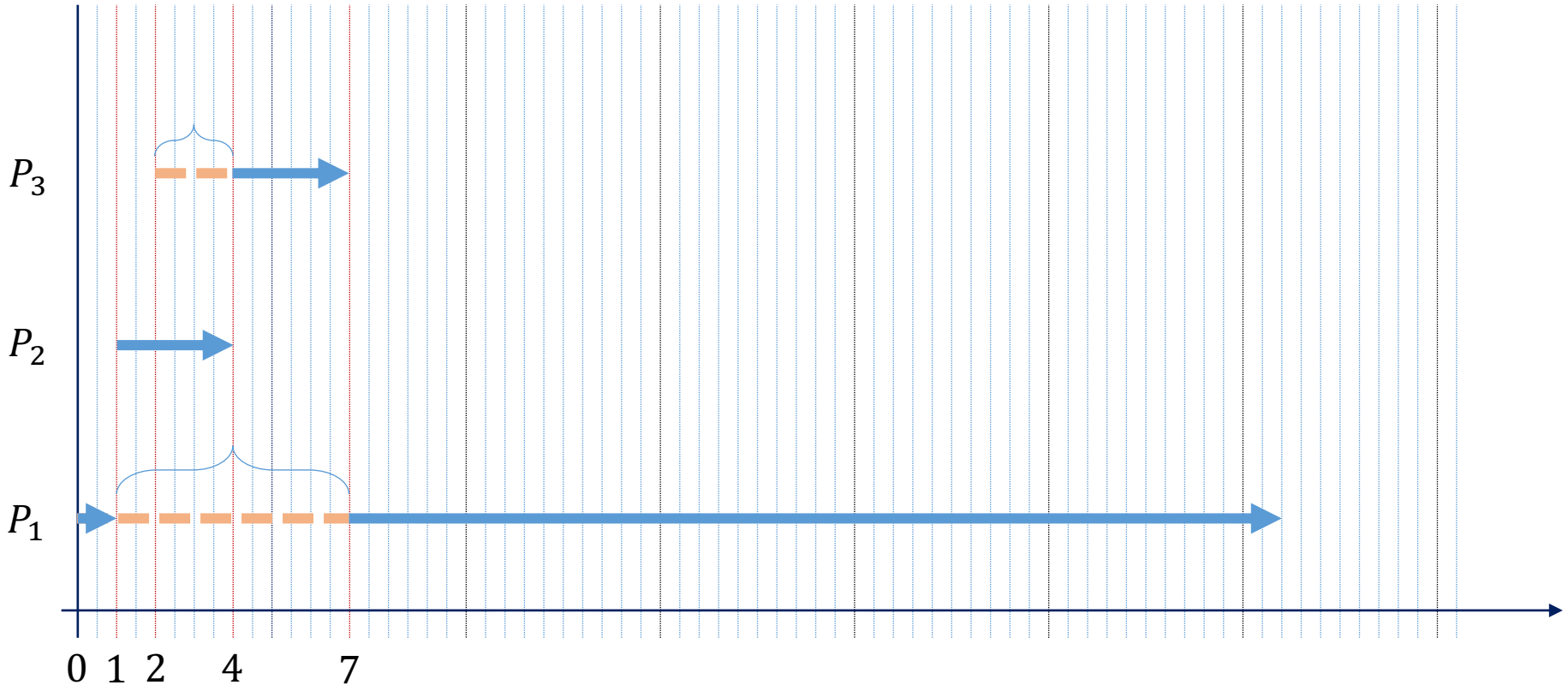
Tiến trình	Thời điểm vào RL	Độ ưu tiên	Thời gian xử lý
P_1	0	3	24
P_2	1	1	3
P_3	2	2	3

$$wait_{P_1} = 3 + 3 = 6$$

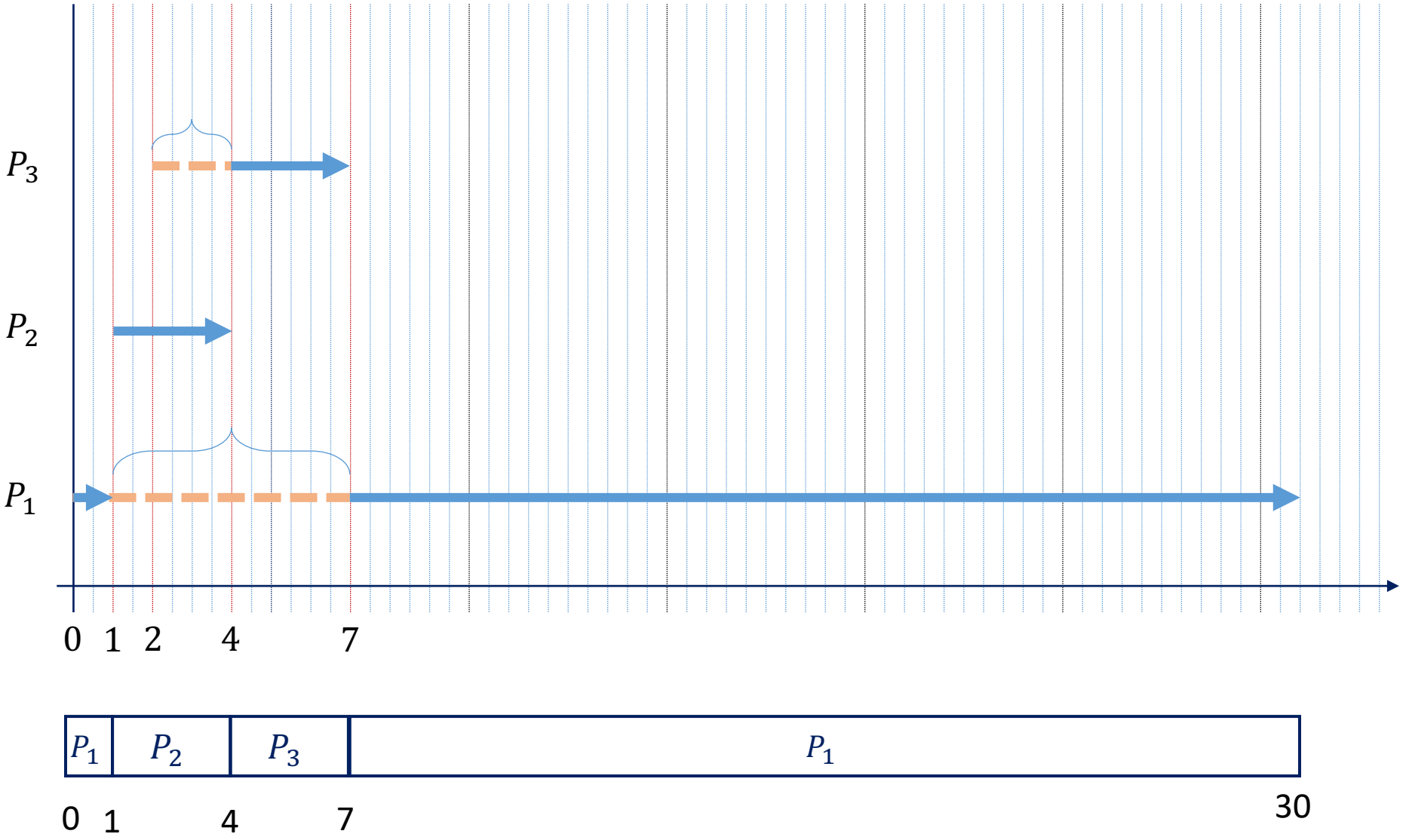
$$wait_{P_2} = 0$$

$$wait_{P_3} = 2$$

$$wait_{avg} = (6 + 0 + 2)/3 = 8/3$$



Thuật toán độ ưu tiên



Thuật toán độ ưu tiên

Tiến trình	Thời điểm vào RL (1)	Thời gian xử lý (2)	Complete time (3)	Turn around time (4)=(3)-(1)	Thời gian chờ (5)=(4)-(2)
P_1	0	24	30	30-0=30	30-24=6
P_2	1	3	4	4-1=3	3-3=0
P_3	2	3	7	7-2=5	5-3=2

Turn Around Time: thời gian kể từ lúc đưa vào (submission) đến lúc kết thúc

Thời gian chờ trung bình là $\frac{6+0+2}{3} = \frac{8}{3}$ miliseconds.

Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

- t: thời gian xử lý mà tiến trình còn yêu cầu
- Độ ưu tiên p = 1/t

Tiến trình	Thời điểm vào RL	Thời gian xử lý
P_1	0	6
P_2	1	8
P_3	2	4
P_4	3	2

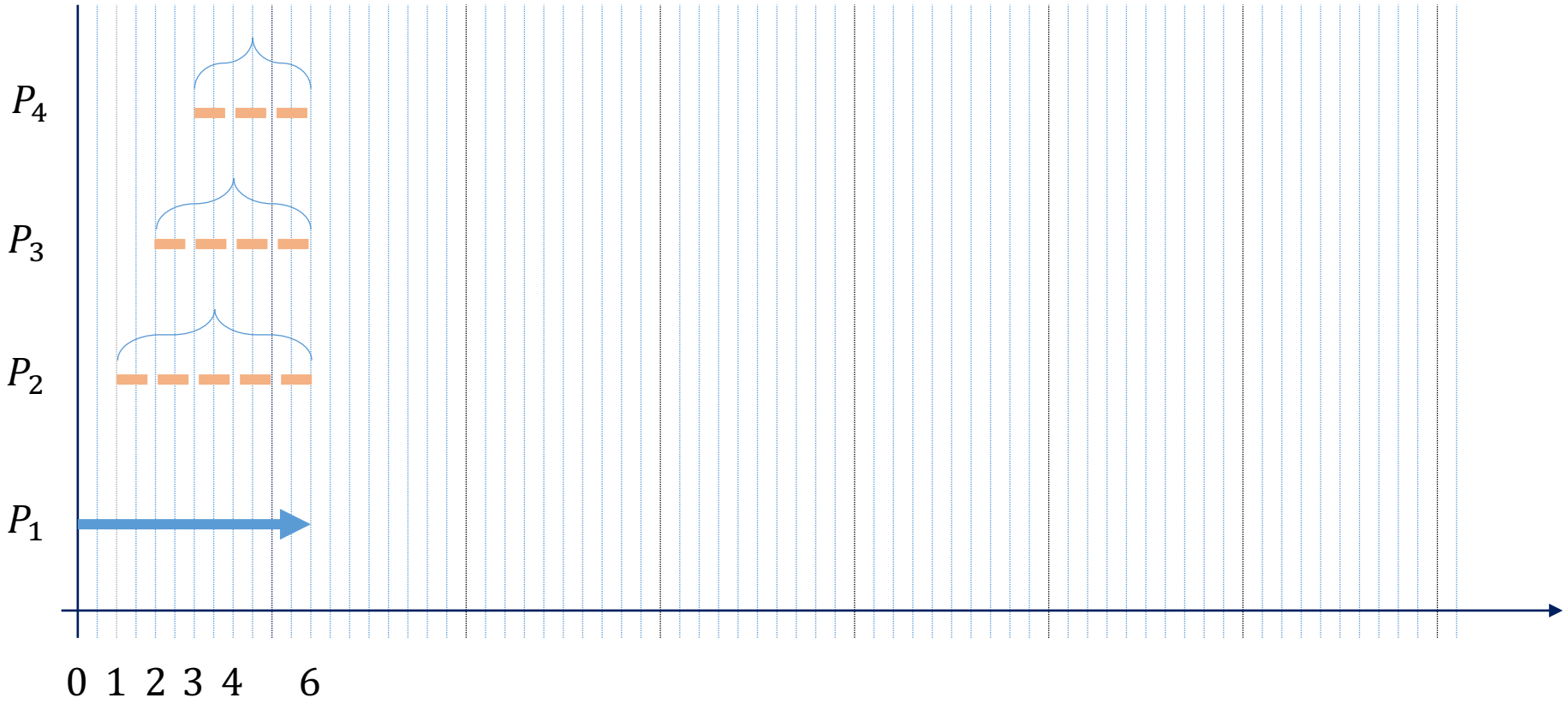
Thuật giải SJF độc quyền

Thuật giải SJF không độc quyền

Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Tiến trình	Thời điểm vào RL	Thời gian xử lý	Độ ưu tiên
P_1	0	6	1/6
P_2	1	8	1/8
P_3	2	4	1/4
P_4	3	2	1/2

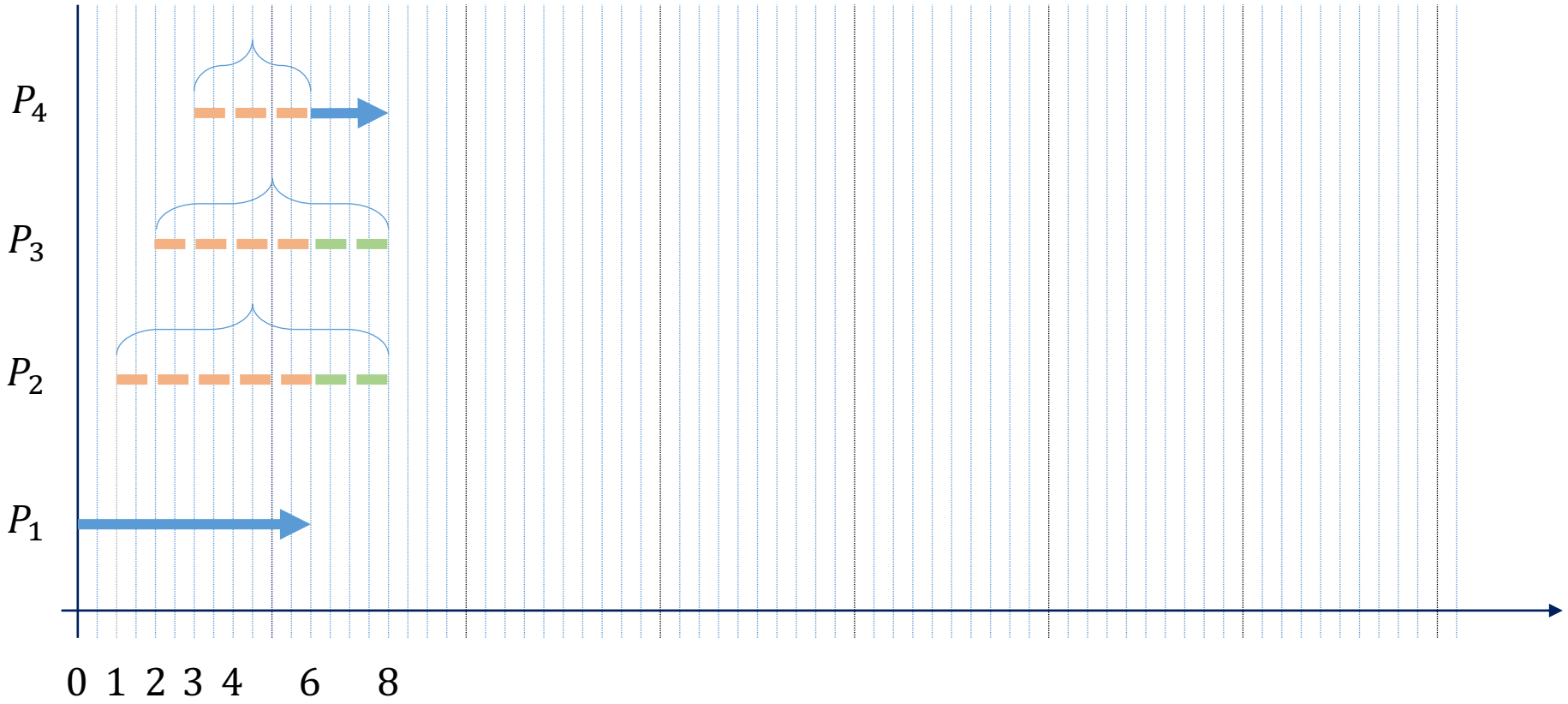
Thuật giải SJF **độc quyền**



Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Tiến trình	Thời điểm vào RL	Thời gian xử lý	Độ ưu tiên
P_1	0	6	1/6
P_2	1	8	1/8
P_3	2	4	1/4
P_4	3	2	1/2

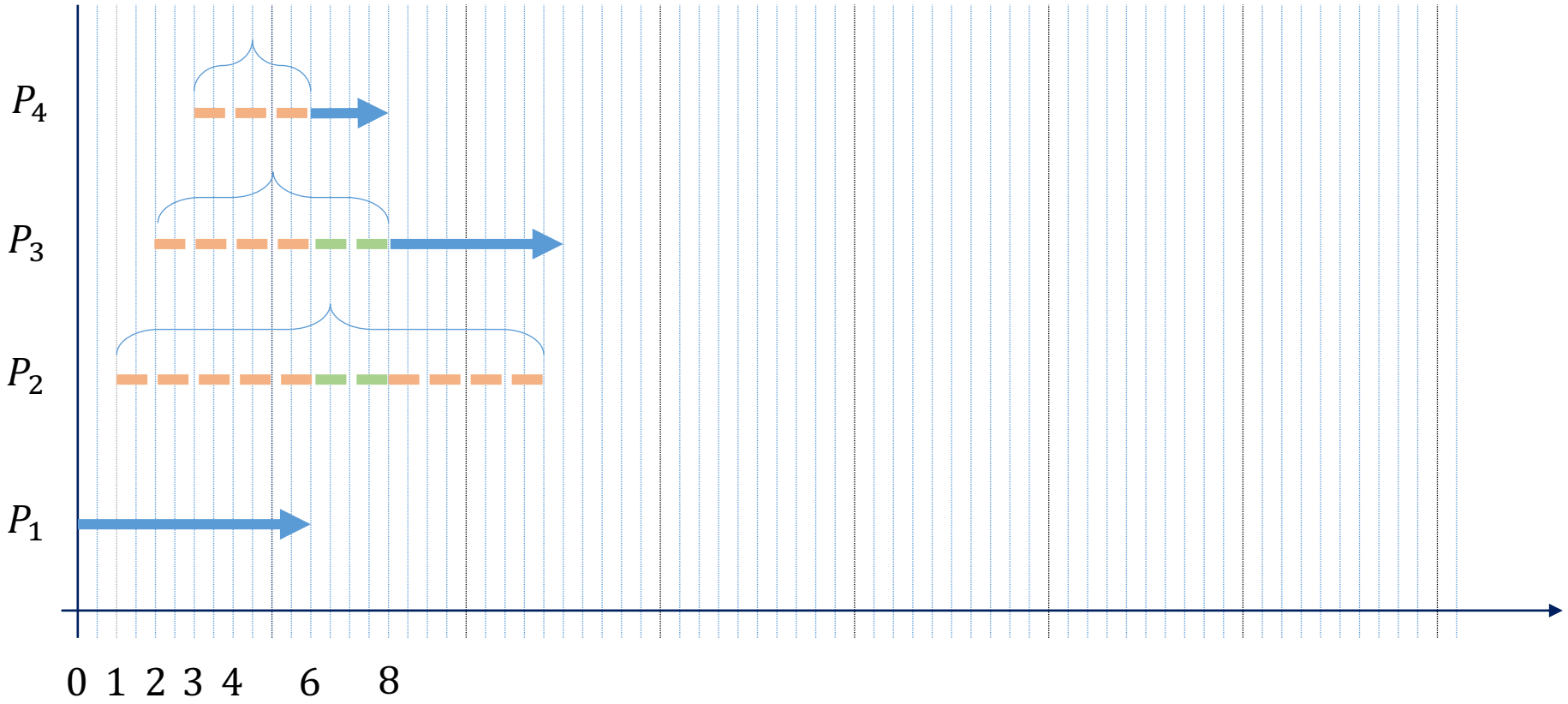
Thuật giải SJF **độc quyền**



Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Tiến trình	Thời điểm vào RL	Thời gian xử lý	Độ ưu tiên
P_1	0	6	1/6
P_2	1	8	1/8
P_3	2	4	1/4
P_4	3	2	1/2

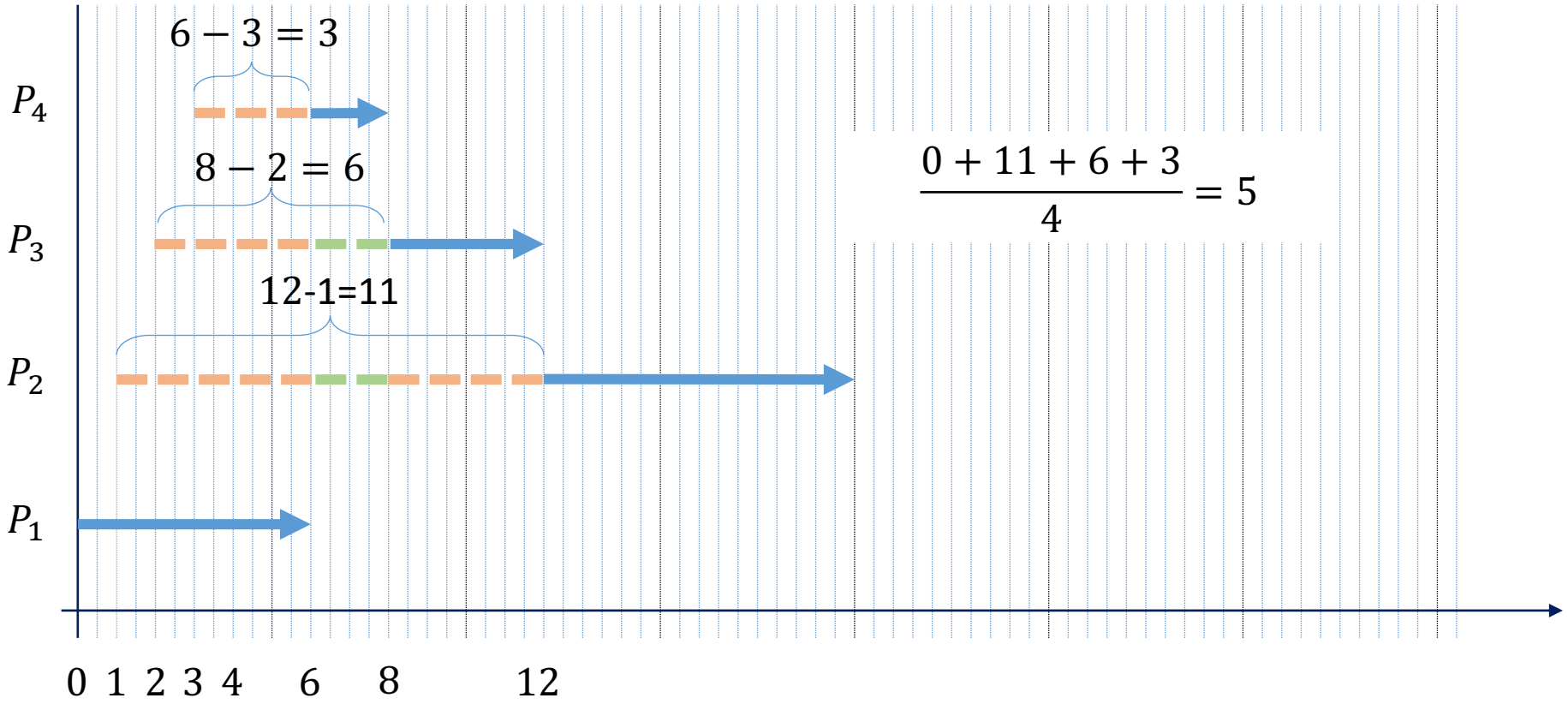
Thuật giải SJF **độc quyền**



Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

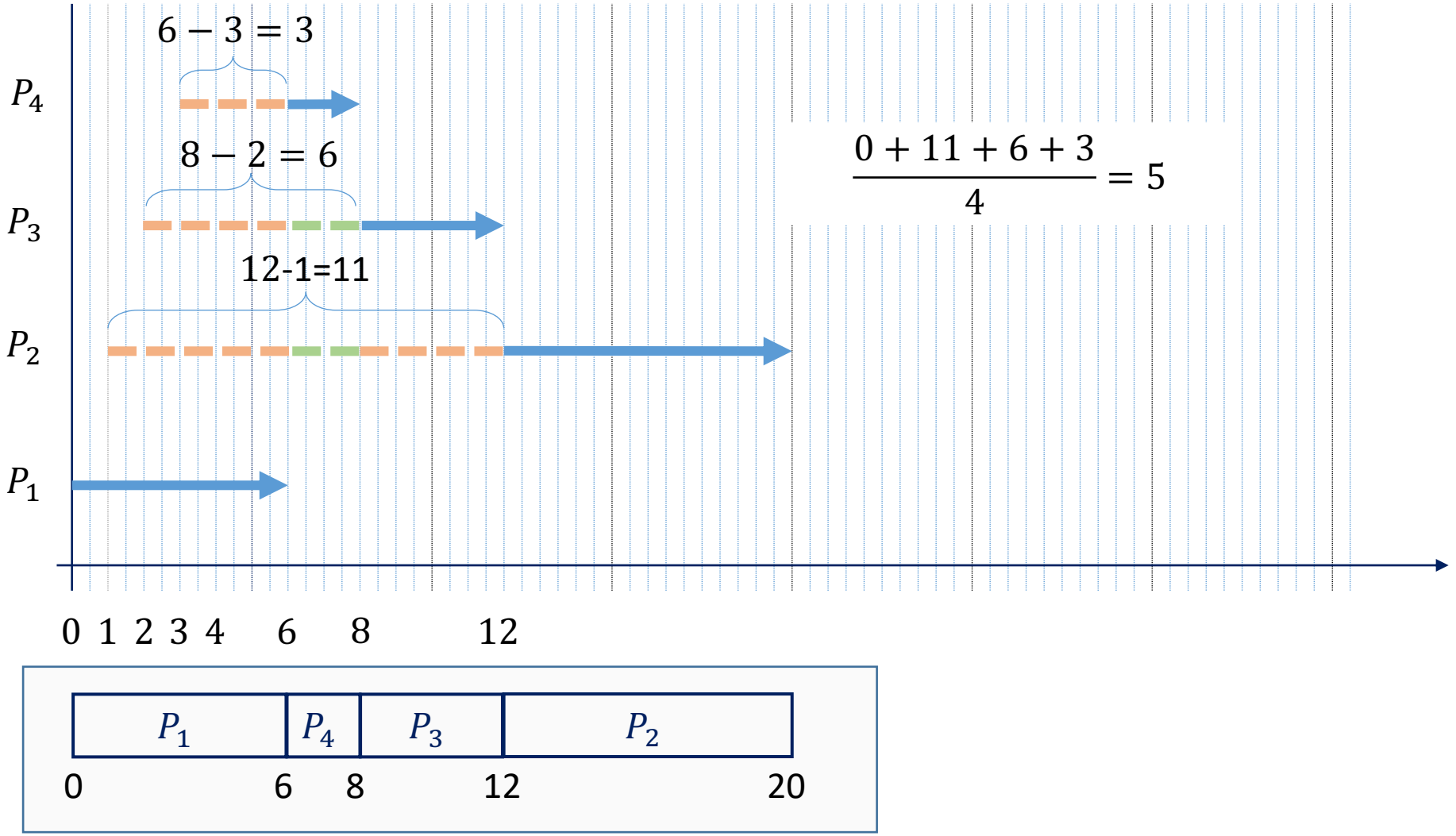
Tiến trình	Thời điểm vào RL	Thời gian xử lý	Độ ưu tiên
P_1	0	6	1/6
P_2	1	8	1/8
P_3	2	4	1/4
P_4	3	2	1/2

Thuật giải SJF **độc quyền**



Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Thuật giải SJF **độc quyền**



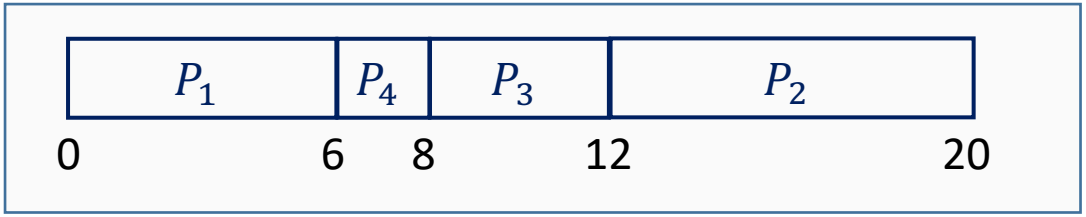
Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Thuật giải SJF **độc quyền**

Tiến trình	Thời điểm vào RL (1)	Thời gian xử lý (2)	Complete time (3)	Turn around time (4)=(3)-(1)	Thời gian chờ (5)=(4)-(2)
P_1	0	6	6	6-0=6	6-6=0
P_2	1	8	20	20-1=19	19-8=11
P_3	2	4	12	12-2=10	10-4=6
P_4	3	2	8	8-3=5	5-2=3

Turn Around Time: thời gian kể từ lúc đưa vào (submission) đến lúc kết thúc

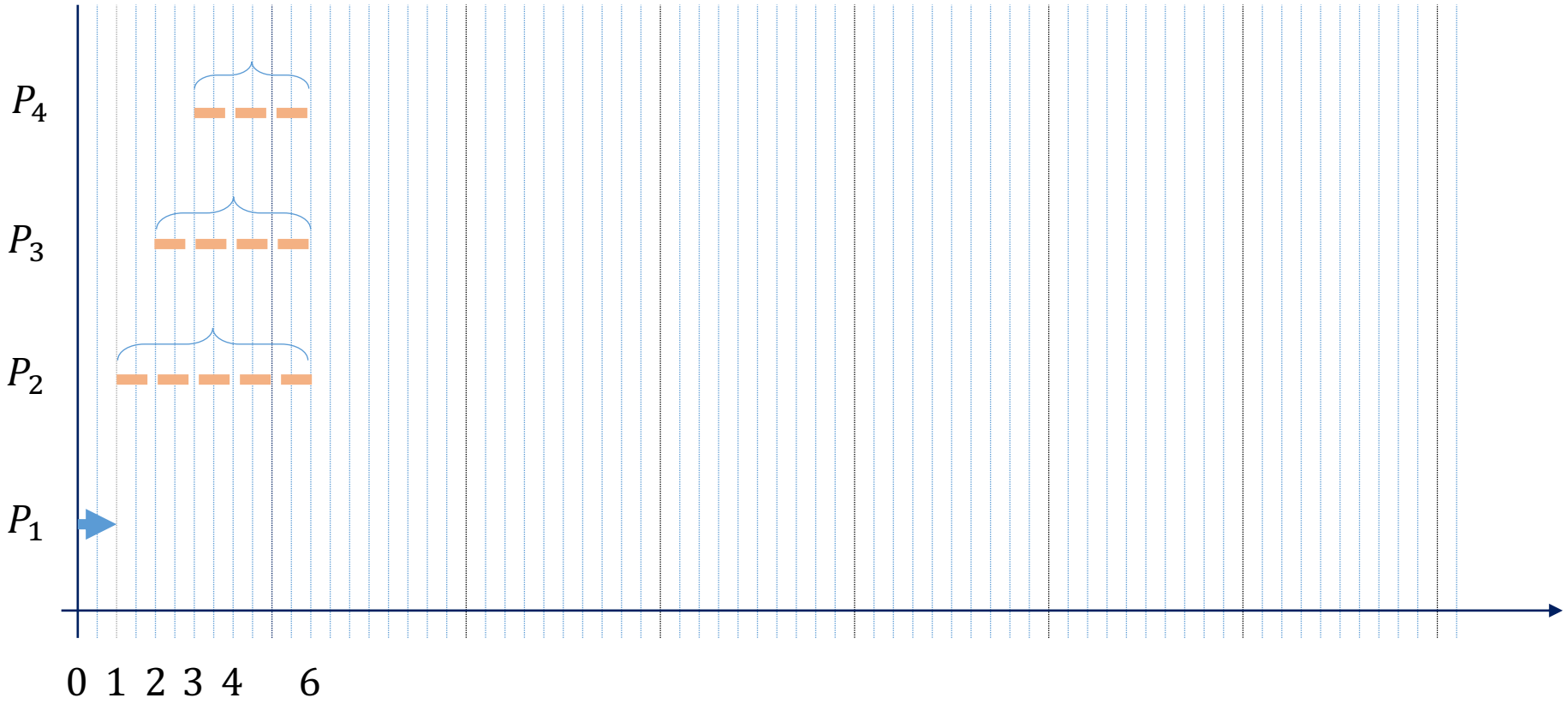
Thời gian chờ trung bình là $\frac{0+11+6+3}{4} = \frac{20}{4} = 5$ miliseconds.



Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Tiến trình	Thời điểm vào RL	Thời gian xử lý	Độ ưu tiên
P_1	0	6	1/6
P_2	1	8	
P_3	2	4	
P_4	3	2	

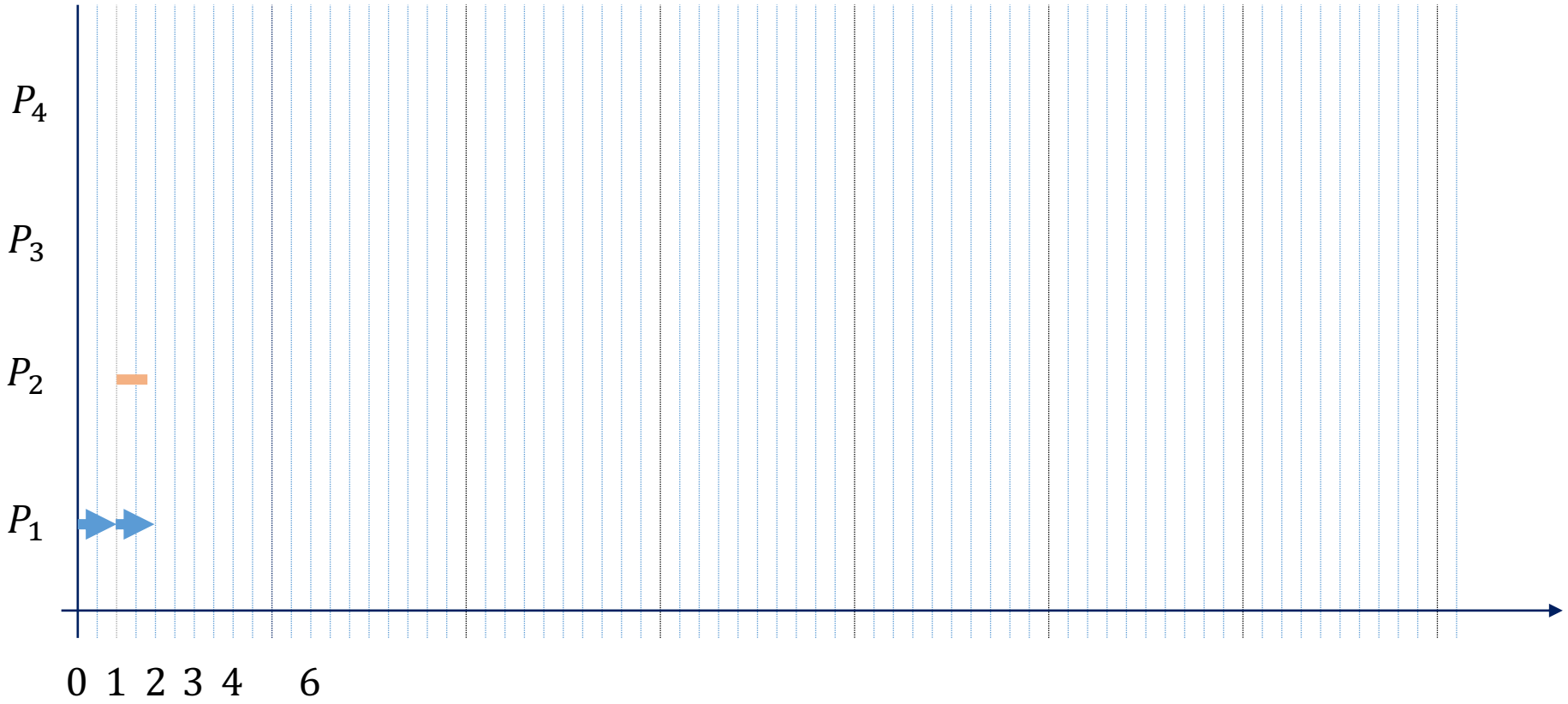
Thuật giải SJF không độc quyền



Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Tiến trình	Thời điểm vào RL	Thời gian xử lý	Độ ưu tiên
P_1	0	6⟶5	1/5
P_2	1	8	1/8
P_3	2	4	
P_4	3	2	

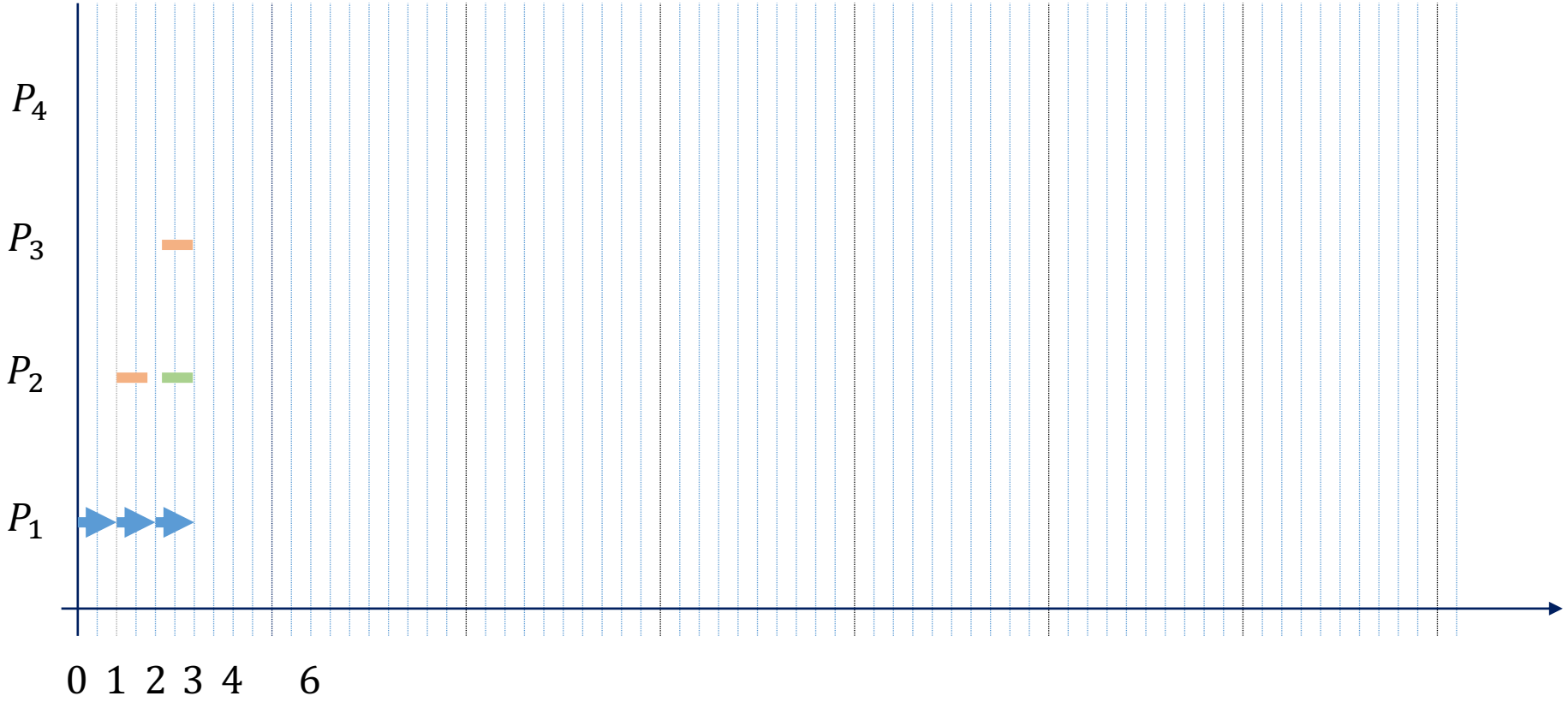
Thuật giải SJF không độc quyền



Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Tiến trình	Thời điểm vào RL	Thời gian xử lý	Độ ưu tiên
P_1	0	5⟶ 4	1/4
P_2	1	8	1/8
P_3	2	4	1/4
P_4	3	2	

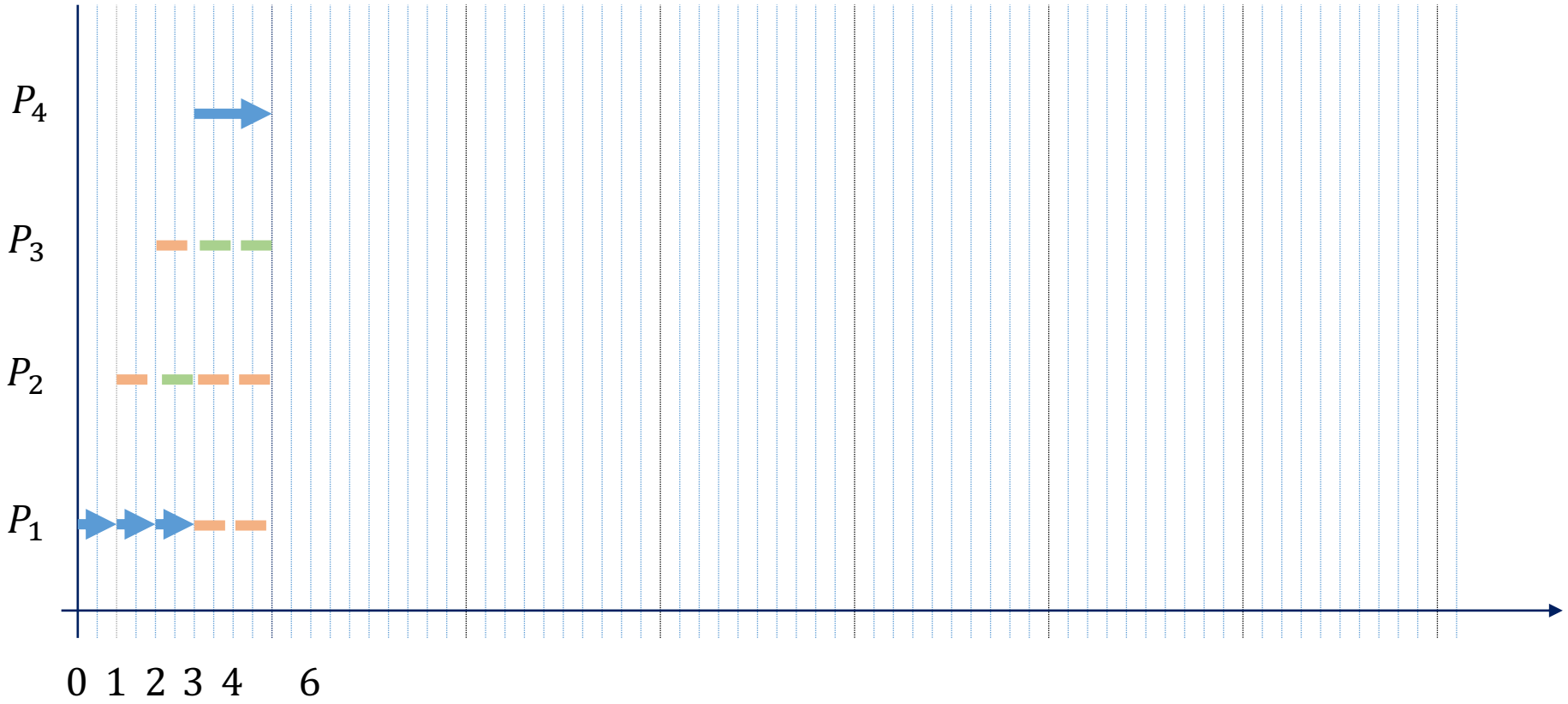
Thuật giải SJF không độc quyền



Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Tiến trình	Thời điểm vào RL	Thời gian xử lý	Độ ưu tiên
P_1	0	4⟶ 3	1/3
P_2	1	8	1/8
P_3	2	4	1/4
P_4	3	2	1/2

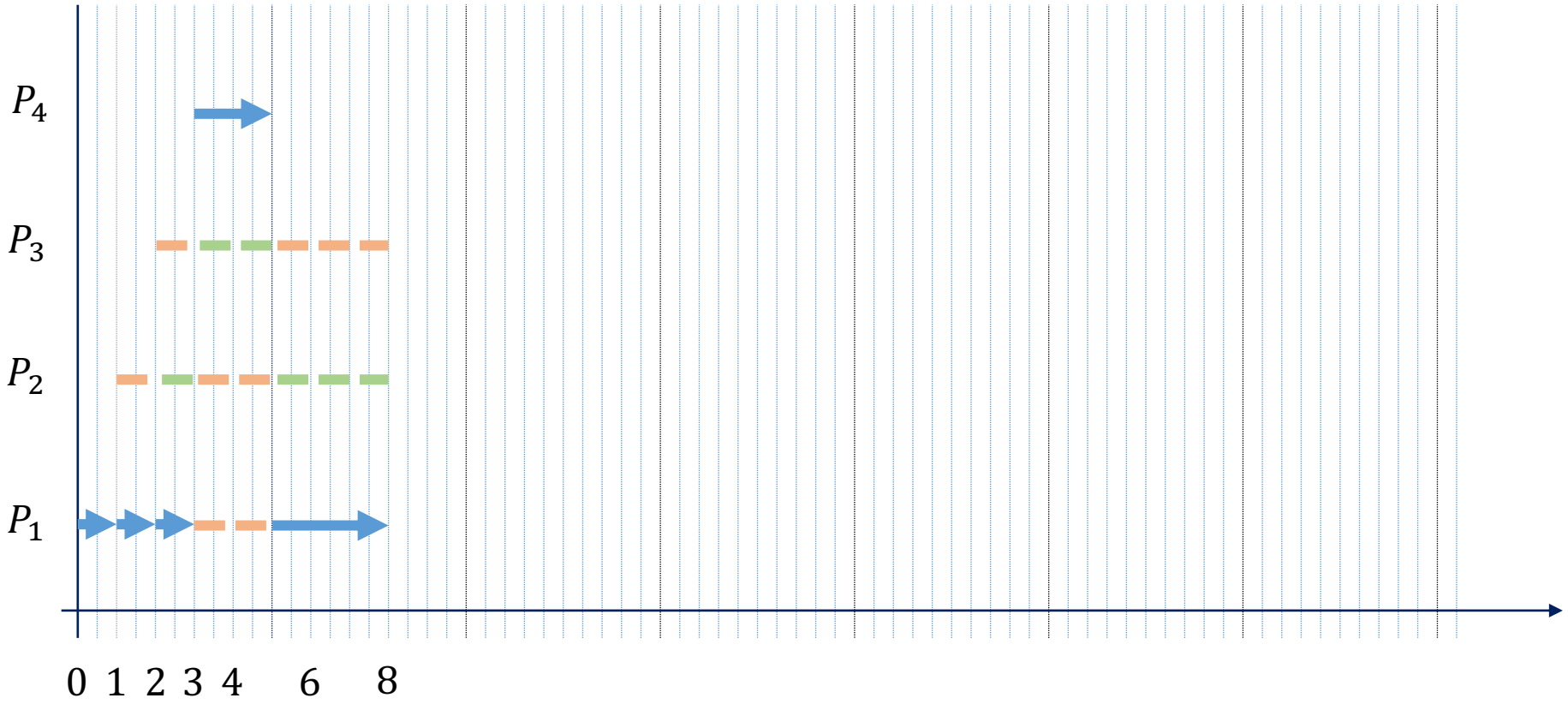
Thuật giải SJF không độc quyền



Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Tiến trình	Thời điểm vào RL	Thời gian xử lý	Độ ưu tiên
P_1	0	4⟶ 3	1/3
P_2	1	8	1/8
P_3	2	4	1/4
P_4	3	2	1/2

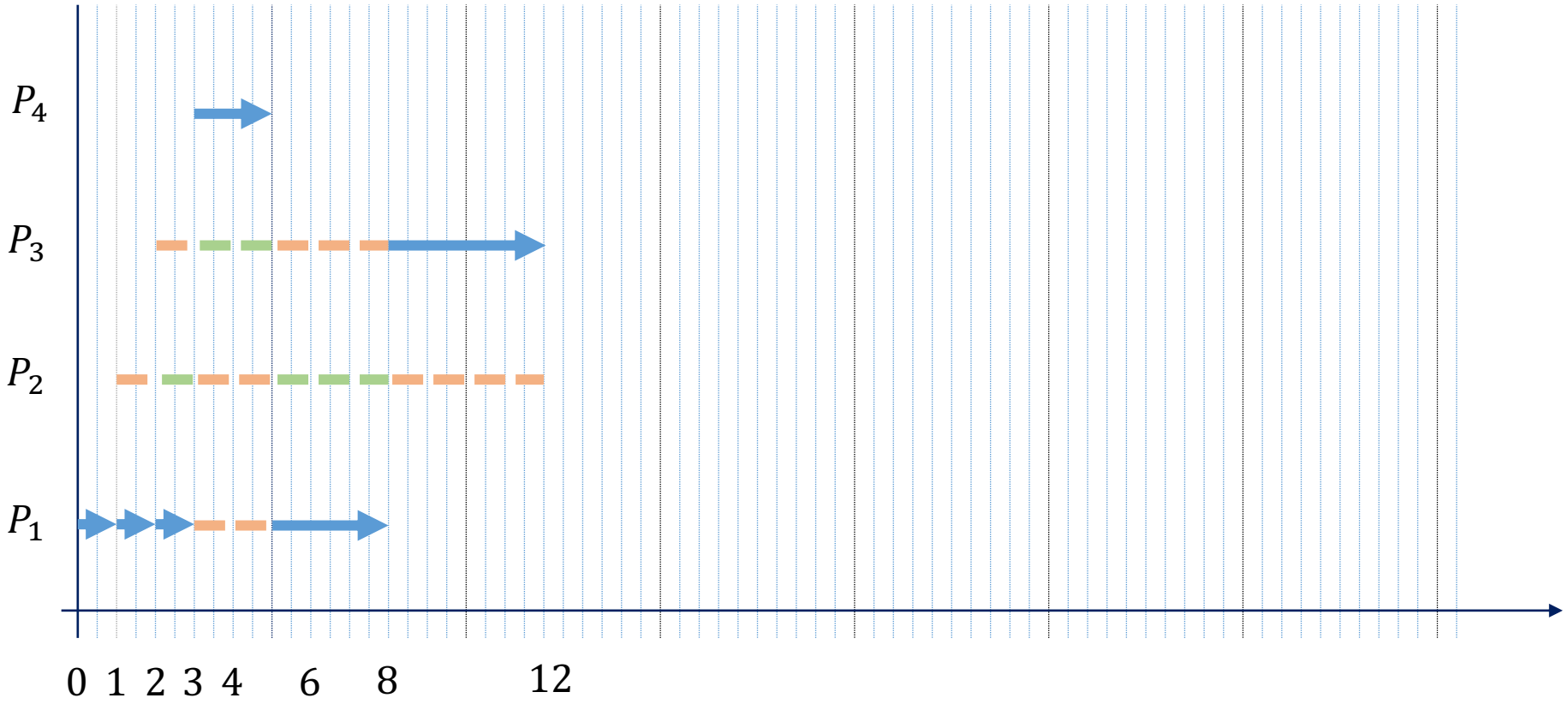
Thuật giải SJF không độc quyền



Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Tiến trình	Thời điểm vào RL	Thời gian xử lý	Độ ưu tiên
P_1	0	$4 \longrightarrow 3$	1/3
P_2	1	8	1/8
P_3	2	4	1/4
P_4	3	2	1/2

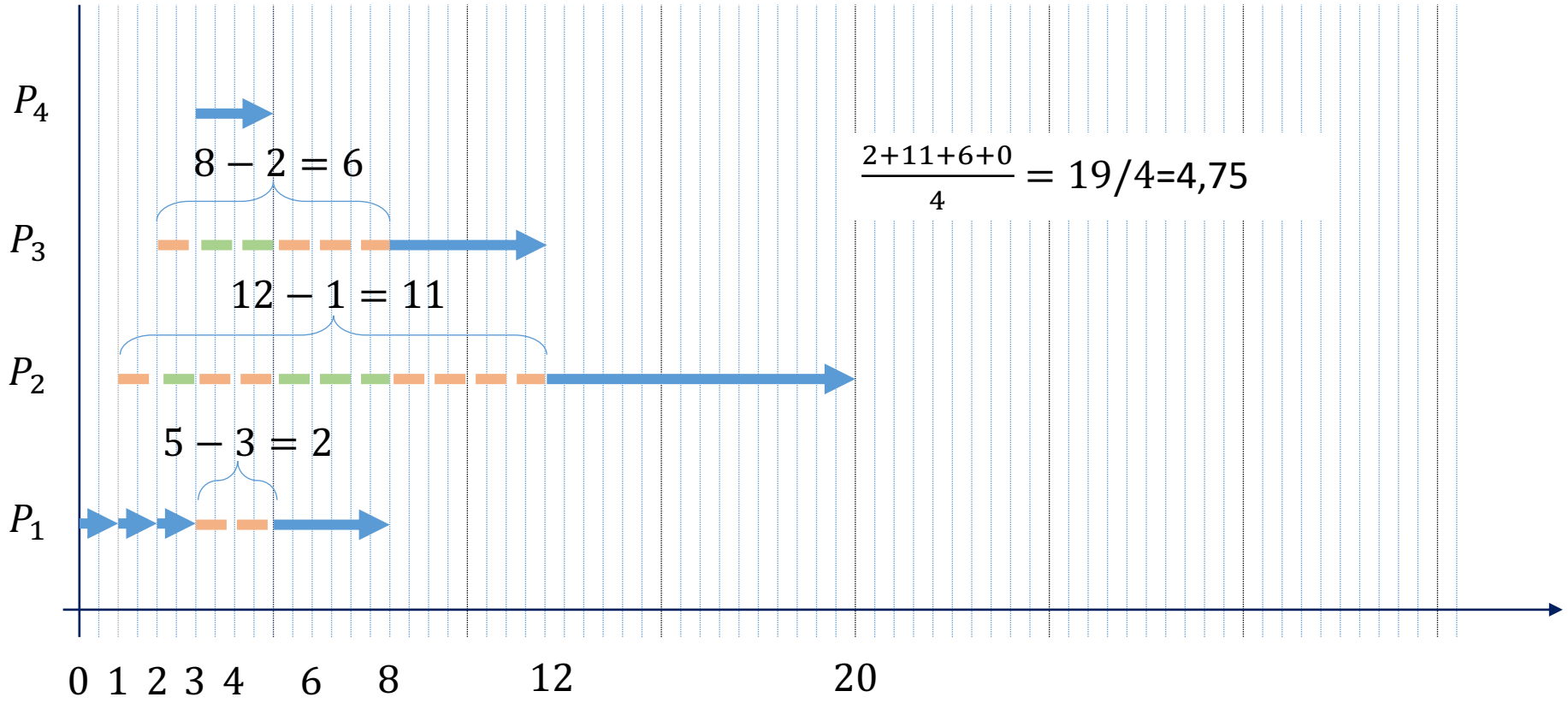
Thuật giải SJF không độc quyền



Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

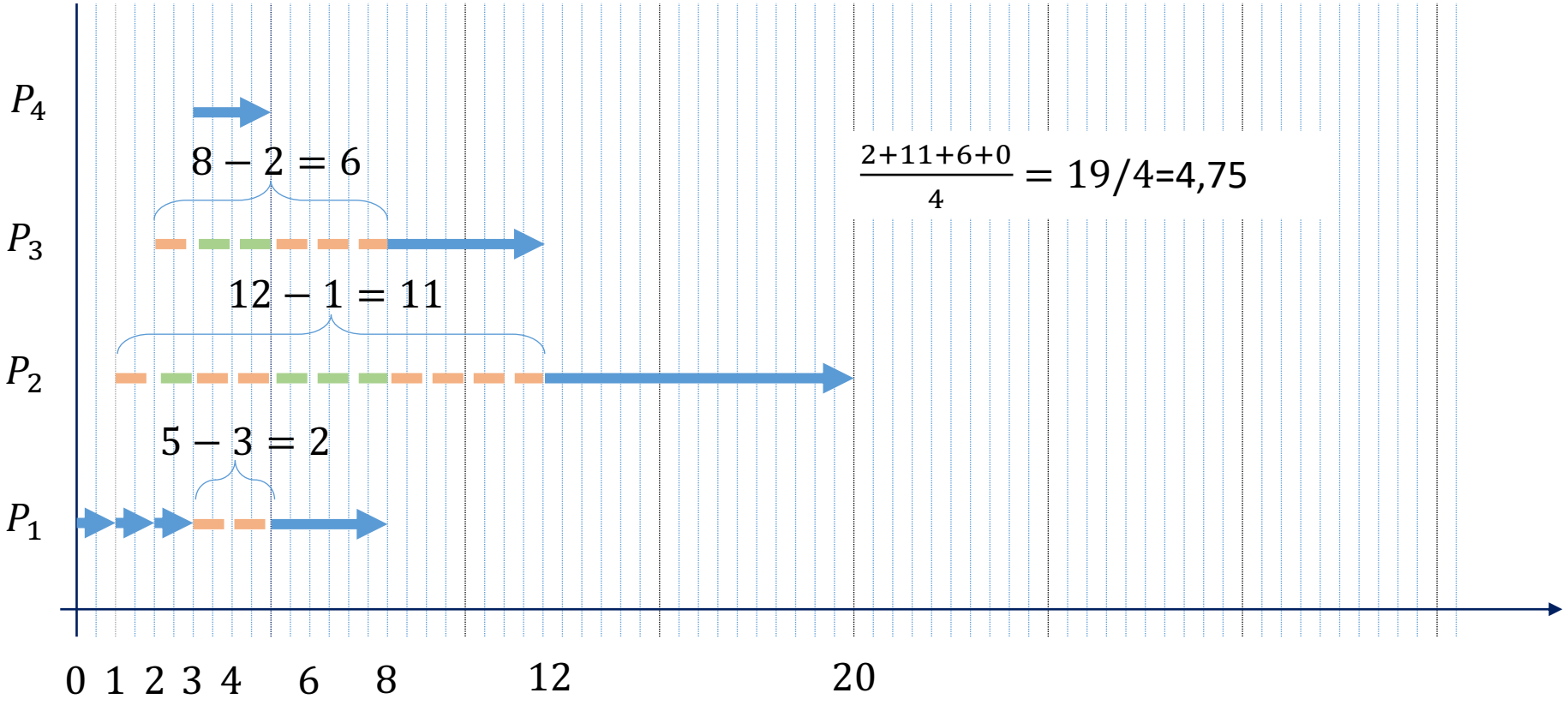
Tiến trình	Thời điểm vào RL	Thời gian xử lý	Độ ưu tiên
P_1	0	4⟶ 3	1/3
P_2	1	8	1/8
P_3	2	4	1/4
P_4	3	2	1/2

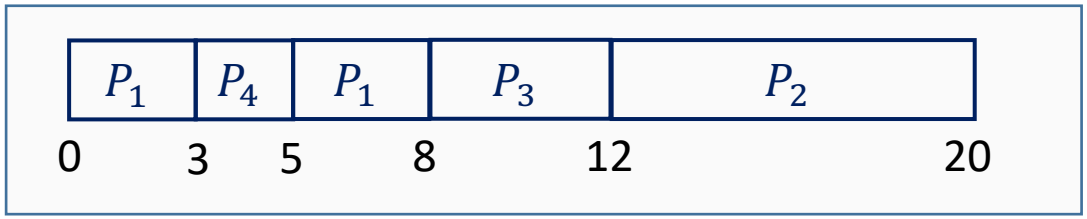
Thuật giải SJF không độc quyền



Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Thuật giải SJF không độc quyền





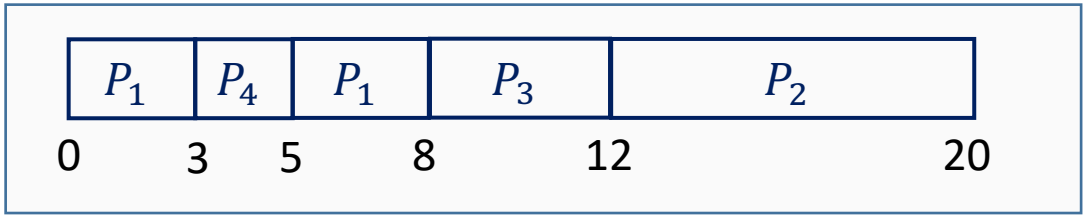
Thuật toán công việc ngắn nhất (Shortest-job-first SJF)

Thuật giải SJF không độc quyền

Tiến trình	Thời điểm vào RL	Thời gian xử lý	Complete time (3)	Turn around time (4)=(3)-(1)	Thời gian chờ (5)=(4)-(2)
P_1	0	6	8	8-0=8	8-6=2
P_2	1	8	20	20-1=19	19-8=11
P_3	2	4	12	12-2=10	10-4=6
P_4	3	2	5	5-3=2	2-2=0

Turn Around Time: thời gian kể từ lúc đưa vào (submission) đến lúc kết thúc

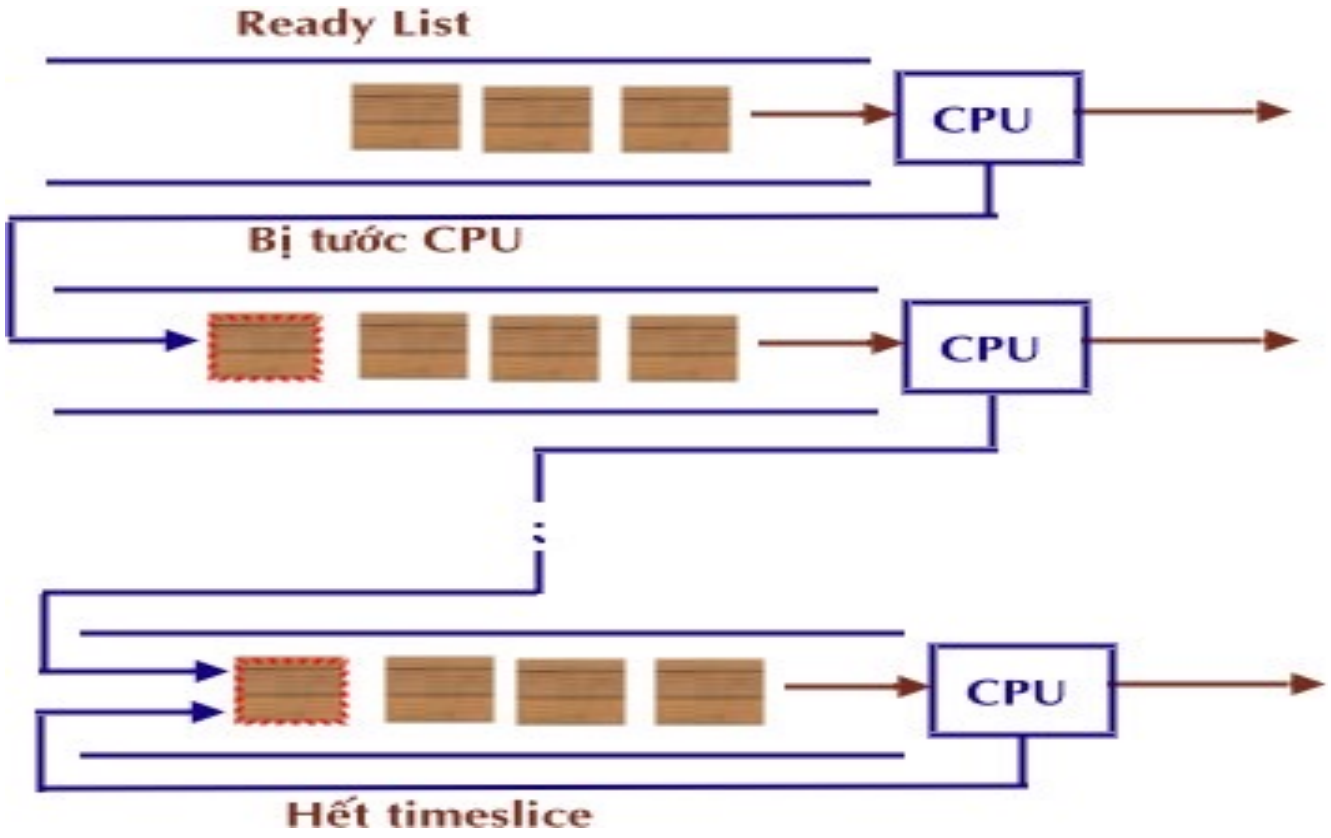
Thời gian chờ trung bình là $\frac{2+11+6+0}{4} = \frac{19}{4} = 4{,}75$ miliseconds.



Thuật toán nhiều mức độ ưu tiên

- Danh sách sẵn sàng được chia thành nhiều danh sách.
- Mỗi danh sách gồm các tiến trình có cùng độ ưu tiên và được áp dụng một giải thuật điều phối riêng

Điều phối theo nhiều mức ưu tiên xoay vòng (Multilevel Feedback)



Chiến lược điều phối xổ số (Lottery)

- Mỗi tiến trình được cấp một "vé số".
- HĐH chọn 1 vé "trúng giải", tiến trình nào sỡ hữu vé này sẽ được nhận CPU.
- Là giải thuật độc quyền.
- Đơn giản, chi phí thấp, bảo đảm tính công bằng cho các tiến trình.

LIÊN LẠC GIỮA CÁC TIẾN TRÌNH

- Mục đích:
  - để chia sẻ thông tin như dùng chung file, bộ nhớ,…
  - hoặc hợp tác hoàn thành công việc
- Các cơ chế:
  - Liên lạc bằng tín hiệu (Signal)
  - Liên lạc bằng đường ống (Pipe)
  - Liên lạc qua vùng nhớ chia sẻ (shared memory)
  - Liên lạc bằng thông điệp (Message)
  - Liên lạc qua socket

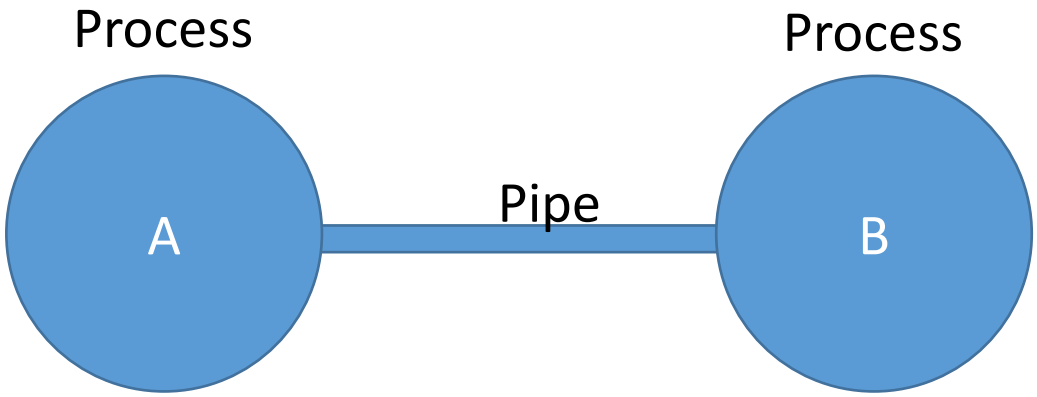
Liên lạc bằng tín hiệu (Signal)

Tín hiệu	Mô tả
SIGINT	Người dùng nhấn phím Ctl-C để ngắt xử lý tiến trình
SIGILL	Tiến trình xử lý một chỉ thị bất hợp lệ
SIGKILL	Yêu cầu kết thúc một tiến trình
SIGFPT	Lỗi chia cho 0
SIGSEGV	Tiến trình truy xuất đến một địa chỉ bất hợp lệ
SIGCLD	Tiến trình con kết thúc

Tín hiệu được gởi đi bởi:
- Phần cứng
- Hệ điều hành:
- Tiến trình:
- Người sử dụng:

Khi tiến trình nhận tín hiệu:
- Gọi hàm xử lý tín hiệu.
- Xử lý theo cách riêng của tiến trình.
- Bỏ qua tín hiệu.

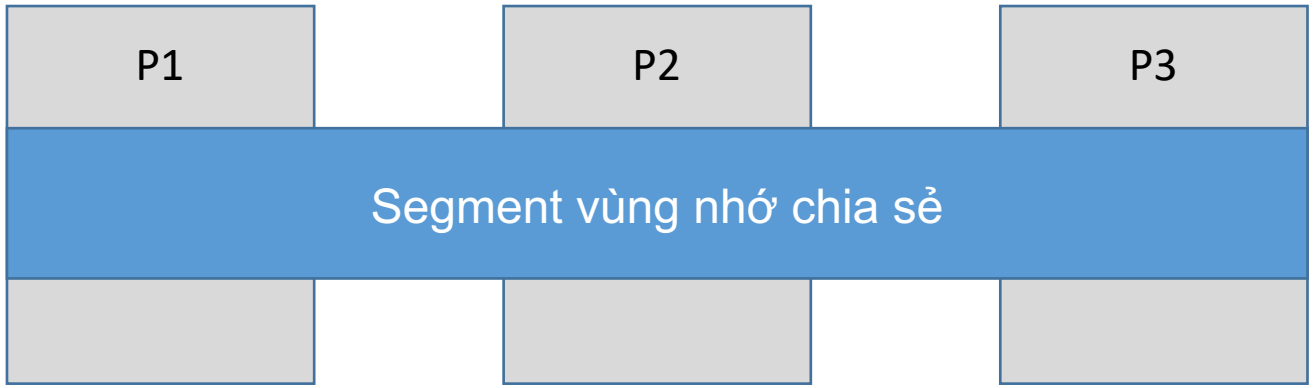
Liên lạc bằng đường ống (Pipe)



- Dữ liệu truyền: dòng các byte (FIFO)
- Tiến trình đọc pipe sẽ bị khóa nếu pipe trống, và đợi đến khi pipe có dữ liệu mới được truy xuất.
- Tiến trình ghi pipe sẽ bị khóa nếu pipe đầy, và đợi đến khi pipe có chỗ trống để chứa dữ liệu.

Liên lạc qua vùng nhớ chia sẻ (shared memory)

- Vùng nhớ chia sẻ độc lập với các tiến trình
- Tiến trình phải gắn kết vùng nhớ chung vào không gian địa chỉ riêng của tiến trình



Vùng nhớ chia sẻ là:

- phương pháp nhanh nhất để trao đổi dữ liệu giữa các tiến trình.
- cần được bảo vệ bằng những cơ chế đồng bộ hóa.
- không thể áp dụng hiệu quả trong các hệ phân tán

Liên lạc bằng thông điệp (Message)

- thiết lập một mối liên kết giữa hai tiến trình
- sử dụng các hàm send, receive do hệ điều hành cung cấp để trao đổi thông điệp
- **Cách liên lạc bằng thông điệp:**
  - **Liên lạc gián tiếp (indirect communication)**
    - Send(A, message): gởi một thông điệp tới port A
    - Receive(A, message): nhận một thông điệp từ port A
  - **Liên lạc trực tiếp (direct communication)**
    - Send(P, message): gởi một thông điệp đến tiến trình P
    - Receive(Q,message) : nhận một thông điệp từ tiến trình Q

Ví dụ: Bài toán nhà sản xuất - người tiêu thụ (producer-consumer)

```
void nsx()
{       while(1)
        {       tạo_sp();
                send(ntt,sp); //gởi sp cho ntt
        }
}
void ntt()
{       while(1)
        {       receive(nsx,sp); //ntt chờ nhận sp
                tiêu_thụ(sp);
        }
}
```

Liên lạc qua socket

- Mỗi tiến trình cần tạo một socket riêng
- Mỗi socket được kết buộc với một cổng khác nhau.
- Các thao tác đọc/ghi lên socket chính là sự trao đổi dữ liệu giữa hai tiến trình.
- Cách liên lạc qua socket
  - Liên lạc kiểu thư tín (socket đóng vai trò bưu cục)
    - "tiến trình gởi" ghi dữ liệu vào socket của mình, dữ liệu sẽ được chuyển cho socket của "tiến trình nhận"
    - "tiến trình nhận" sẽ nhận dữ liệu bằng cách đọc dữ liệu từ socket của "tiến trình nhận"
  - Liên lạc kiểu điện thoại (socket đóng vai trò tổng đài)
    - Hai tiến trình cần kết nối trước khi truyền/nhận dữ liệu và kết nối được duy trì suốt quá trình truyền nhận dữ liệu

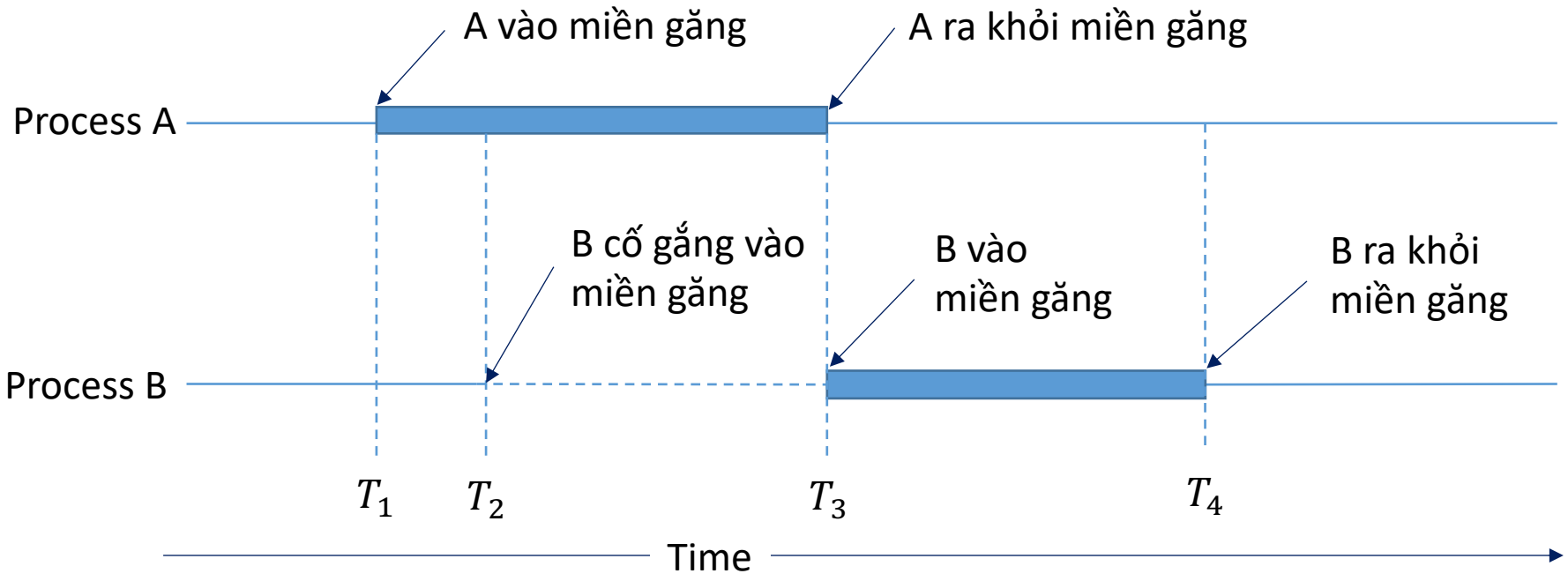
ĐỒNG BỘ CÁC TIẾN TRÌNH

- bảo đảm các tiến trình xử lý song song không tác động sai lệch đến nhau.
- **Yêu cầu độc quyền truy xuất (Mutual exclusion):**
  - tại một thời điểm, chỉ có một tiến trình được quyền truy xuất một tài nguyên không thể chia sẻ.
- **Yêu cầu phối hợp (Synchronization):**
  - các tiến trình cần hợp tác với nhau để hoàn thành công việc.
- **Hai "bài toán đồng bộ" cần giải quyết:**
  - bài toán "độc quyền truy xuất" ("bài toán miền găng")
  - bài toán "phối hợp thực hiện".

Miền găng (critical section)

- Đoạn mã của một tiến trình có khả năng xảy ra lỗi khi truy xuất tài nguyên dùng chung (biến, tập tin,…).
- Ví dụ:

```
if (taikhoan >= tienrut)    taikhoan = taikhoan - tienrut;
        else Thong bao “khong the rut tien ! “;
```



Các điều kiện cần khi giải quyết bài toán miền găng

- Không có giả thiết về tốc độ của các tiến trình, cũng như về số lượng bộ xử lý.
- Không có hai tiến trình cùng ở trong miền găng cùng lúc.
- Một tiến trình bên ngoài miền găng không được ngăn cản các tiến trình khác vào miền găng.
- Không có tiến trình nào phải chờ vô hạn để được vào miền găng

Các nhóm giải pháp đồng bộ

- Busy Waiting
- Sleep And Wakeup
  - Semaphore
  - Monitor
- Message.

Busy Waiting (bận thì đợi)

- **Giải pháp phần mềm**
  - Thuật toán sử dụng biến cờ hiệu
  - Thuật toán sử dụng biến luân phiên
  - **Thuật toán Peterson**
- **Giải pháp phần cứng**
  - Cấm ngắt
  - Sử dụng lệnh TSL (Test and Set Lock)

Thuật toán sử dụng biến cờ hiệu (dùng cho nhiều tiến trình)

- lock=0 là không có tiến trình trong miền găng.
- lock=1 là có một tiến trình trong miền găng.

```
lock=0;
while (1)
{
        while (lock == 1);

        lock = 1;
        critical-section ();.
        lock = 0;

        noncritical-section();
}
```

Vi phạm: "Hai tiến trình có thể cùng ở trong miền găng tại một thời điểm".

Thuật toán sử dụng biến luân phiên (dùng cho 2 tiến trình)

- Hai tiến trình A, B sử dụng chung biến turn:
  - turn = 0, tiến trình A được vào miền găng
  - turn=1 thì B được vào miền găng