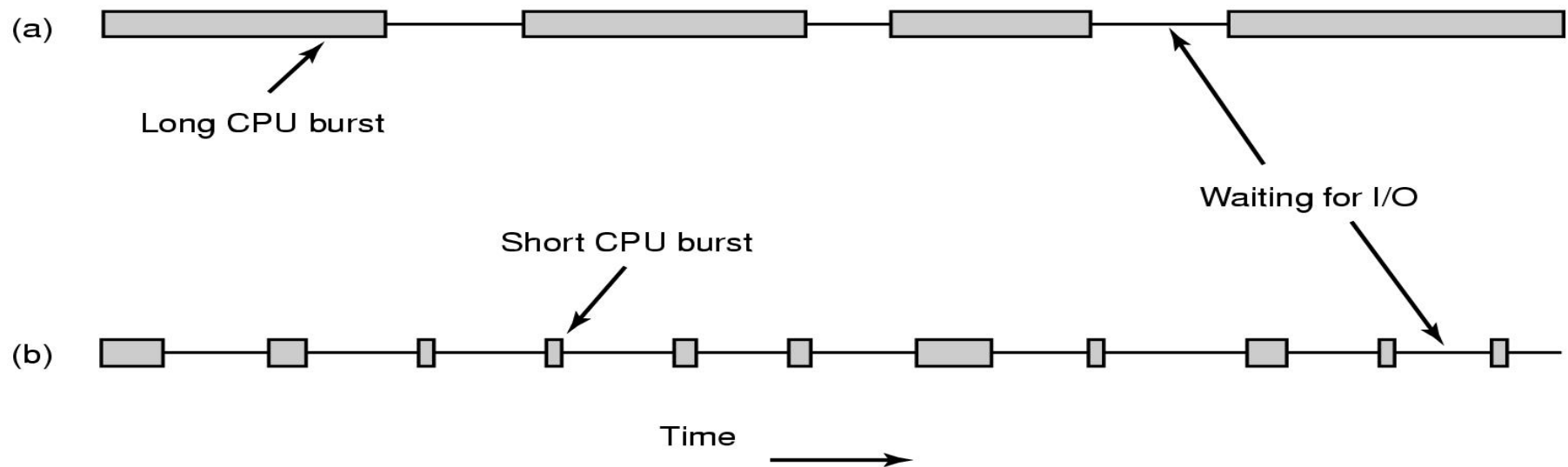


Sistemi Operativi

Lez. 7-8

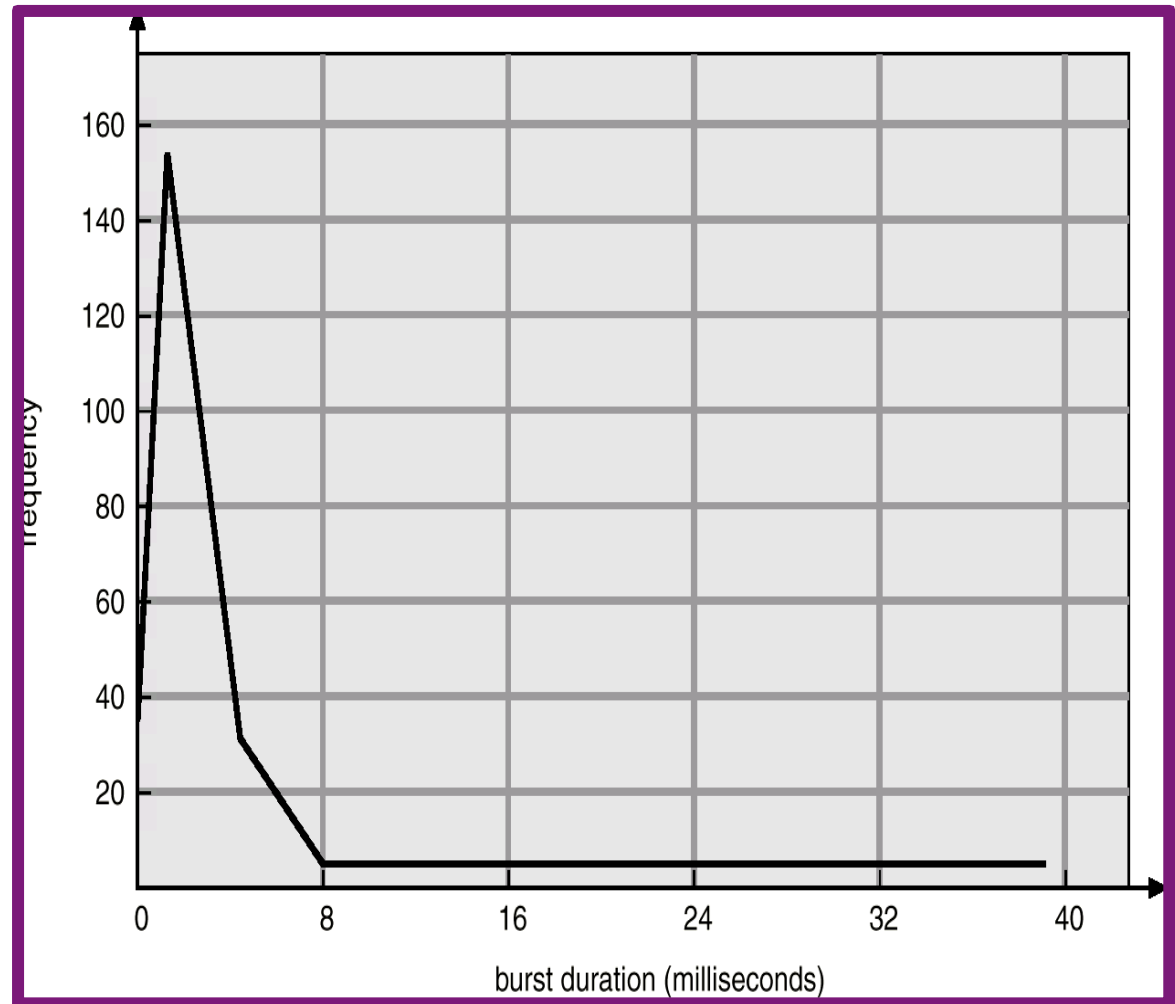
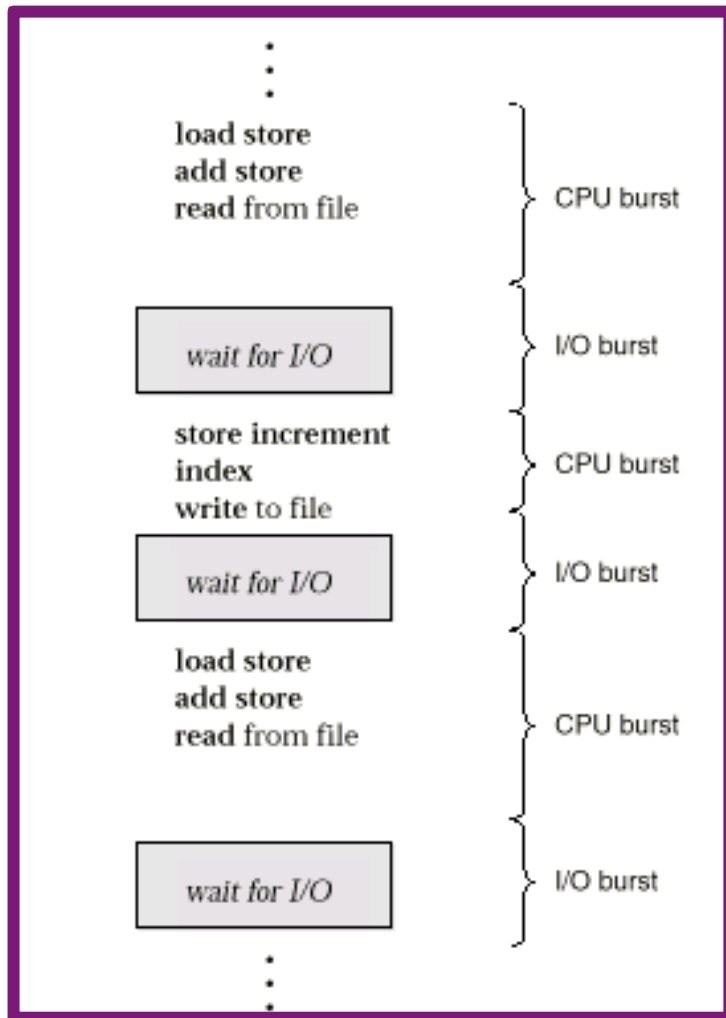
Lo scheduling dei processi

Cicli d'elaborazione



- In ogni processo i burst di CPU si alternano con i tempi di I/O

Usò tipico di un calcolatore



CPU-bound e I/O-bound

- Processi CPU-bound
 - Alternano lunghe computazioni ad attività di I/O relativamente poco frequenti
- Processi I/O-bound
 - Alternano **brevi** computazioni a frequenti attività di I/O
 - Migliorando le prestazioni delle CPU, i CPU burst diventano sempre più brevi
 - I processi I/O-bound devono essere eseguiti ogni volta che lo richiedono per poter tenere occupati i dischi

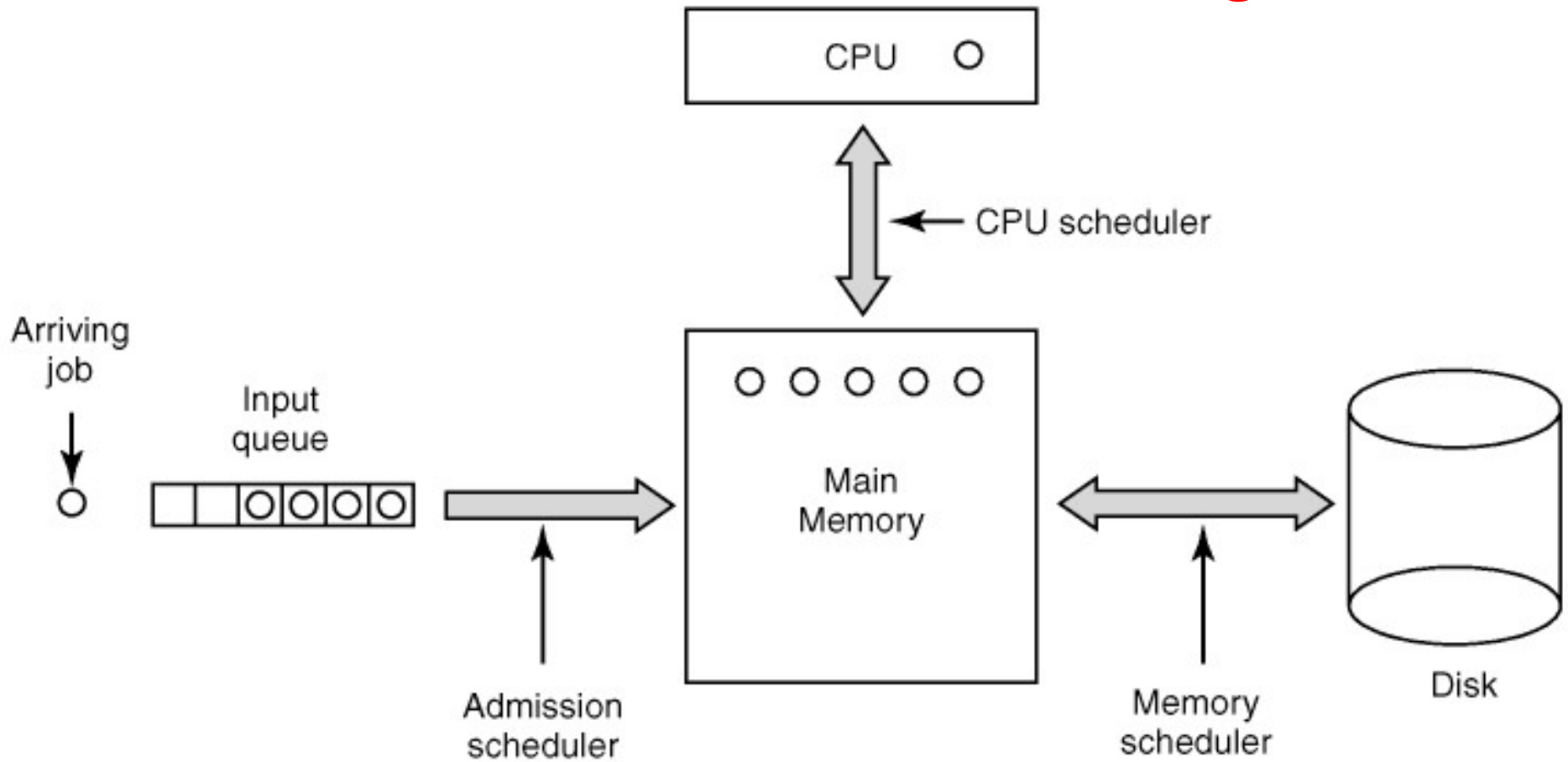
Lo scheduler

- Per evitare di sprecare tempo di CPU durante l'I/O, i processi che eseguono I/O vengono messi in stato di wait (o blocked)
- Viene selezionato tra i processi ready un nuovo processo da eseguire
 - processi pronti per l'esecuzione
- Queste attività vengono svolte da un particolare processo di sistema: lo scheduler

Vari tipi di scheduling

Long-term scheduling	The decision to add to the pool of processes to be executed
Medium-term scheduling	The decision to add to the number of processes that are partially or fully in main memory
Short-term scheduling	The decision as to which available process will be executed by the processor
I/O scheduling	The decision as to which process's pending I/O request shall be handled by an available I/O device

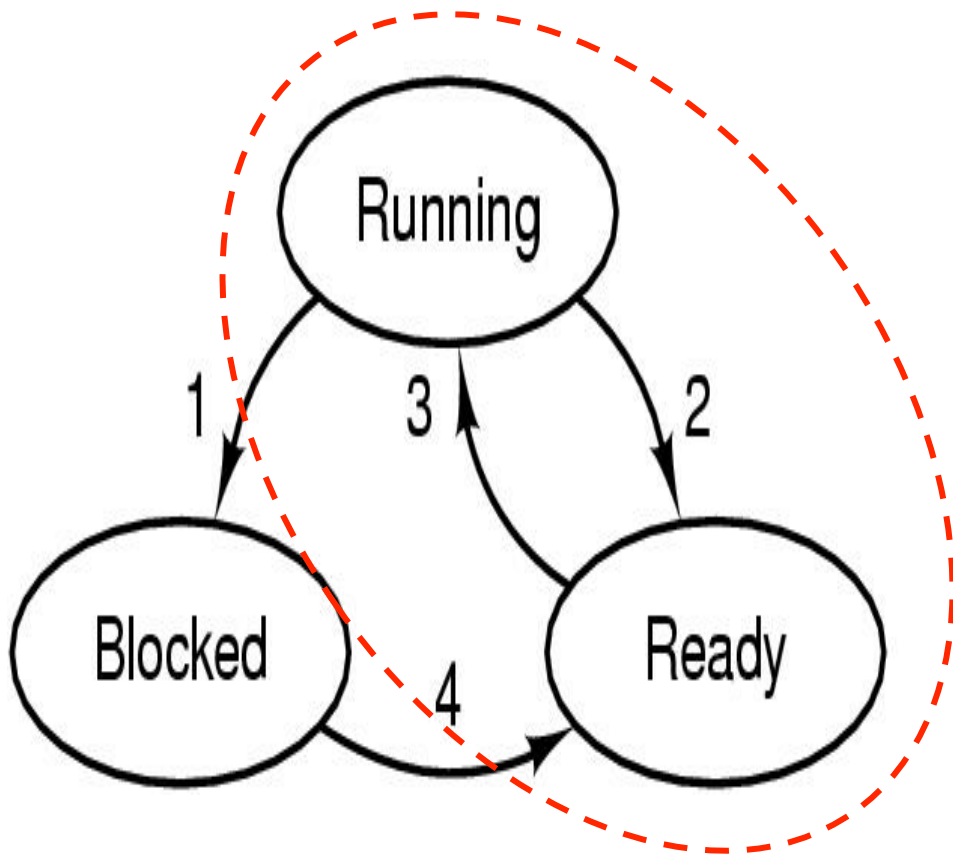
Three Level Scheduling



Short term scheduler

- Noto anche come dispatcher o CPU scheduler
- È lo scheduler eseguito più frequentemente
- Invocato quando si verifica uno dei seguenti eventi
 - Clock interrupts
 - I/O interrupts
 - Operating system calls
 - Signals

CPU scheduler



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

Cosa fa lo scheduler

- Lo scheduler della CPU decide
 - quando eseguire un processo
 - creazione di un nuovo processo
 - terminazione di un processo esistente
 - sospensione di un processo attivo
 - interruzione di un processo attivo
 - esaurimento del tempo assegnato
 - quale processo eseguire
 - per quanto tempo lasciarlo eseguire
 - prelazionabile o no

Criteri adottati per scegliere la vittima:

- Da quanto tempo il processo è stato caricato \scaricato
- Quanto tempo di CPU ha utilizzato recentemente il processo?
- Quanto è grande il processo?
- Quanto importante è il processo?

Obiettivi di progetto

- User-oriented
 - Response Time
 - Tempo che intercorre tra la sottomissione di un processo ed il primo output
- System-oriented
 - Uso efficiente delle risorse del sistema

Obiettivi

All systems

Fairness - giving each process a fair share of the CPU

Policy enforcement - seeing that stated policy is carried out

Balance - keeping all parts of the system busy

Batch systems

Throughput - maximize jobs per hour

Turnaround time - minimize time between submission and termination

CPU utilization - keep the CPU busy all the time

Interactive systems

Response time - respond to requests quickly

Proportionality - meet users' expectations

Real-time systems

Meeting deadlines - avoid losing data

Predictability - avoid quality degradation in multimedia systems

Modalità

- Gli algoritmi di scheduling sono divisi in due grosse categorie:
 - Preemptive (con diritto di prelazione)
 - algoritmi che prevedono la possibilità di sospendere momentaneamente i processi in esecuzione per eseguire altri processi
 - Non preemptive o run to completion
 - un processo in esecuzione mantiene l'uso della CPU sino alla naturale terminazione del burst
 - terminazione del processo o richiesta di I/O

Preemption

- Lo scheduling può avvenire quando un processo passa:
 - Da running a waiting (es. I/O request)
 - Da running a ready (es. IRQ) [preemptive]
 - Da waiting a ready (es. I/O completed) [preemptive]
 - Terminated

First come first served

- FCFS
- Algoritmo non-preemptive
- Implementazione molto facile
- Average wait time molto variabile e generalmente lontano dall'ottimo
 - Dipende dai tempi di esecuzione di chi è in coda davanti a noi
- Da non usare in sistemi interattivi

Esempio

<u>Process</u>	<u>Tempo di arrivo</u>	<u>Burst Time</u>
<i>P1</i>	0	24
<i>P2</i>	10	3
<i>P3</i>	12	3

Tempo di attesa $P1 = 0$; $P2 = 14$; $P3 = 15$

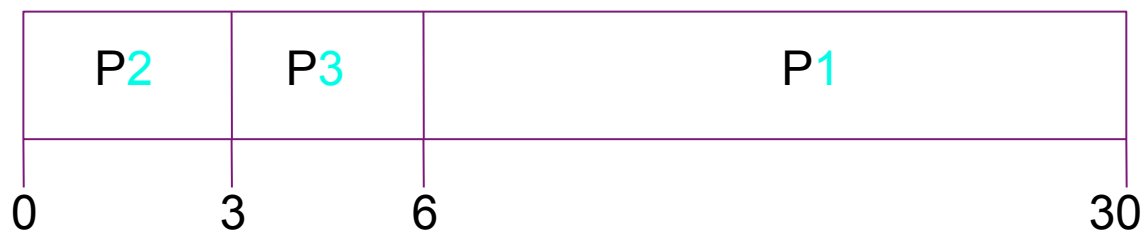
Average waiting time: $(0 + 14 + 15)/3 = 9,66$

Average Turnaround: $(24 + 17 + 18)/3 = 19,66$



Effetto convoglio (convoy effect)

- Se il processo in esecuzione è CPU-bound, molti altri processi (magari I/O-bound) non possono essere eseguiti
- Waiting time for $P1 = 6$; $P2 = 0$; $P3 = 3$
- Average waiting time: $(6 + 0 + 3)/3 = 3$



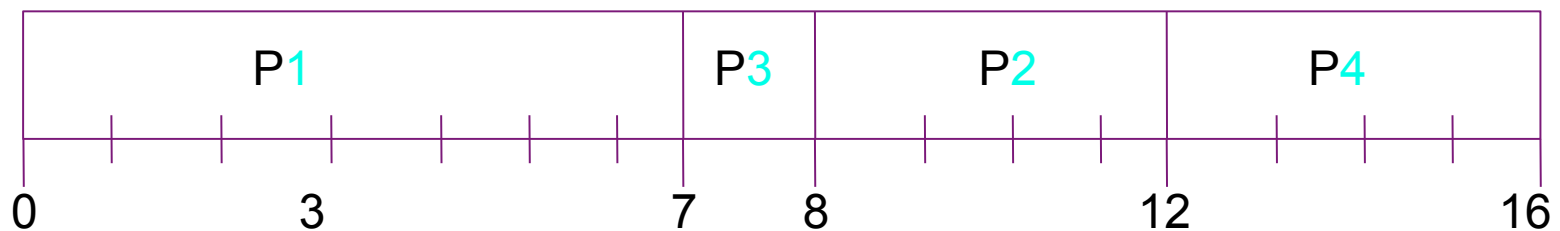
Shortest Job First

- Presuppone la conoscenza del tempo di esecuzione di ciascun processo
- Tutti i processi hanno uguale importanza
- La CPU viene assegnata al processo col tempo di esecuzione minore
- Versione preemptive: shortest remaining time next
 - All'arrivo di un nuovo job si rivaluta l'ordine e la scelta corrente
- Garantisce il minimo tempo di turnaround

Esempio

<u>Processo</u>	<u>Arrival Time</u>	<u>Burst Time</u>
<i>P1</i>	0.0	7
<i>P2</i>	2.0	4
<i>P3</i>	4.0	1
<i>P4</i>	5.0	4

- Average waiting time = $(0 + 6 + 3 + 7)/4 = 4$
- Average Turnaround = $(7 + 10 + 4 + 11)/4 = 8$

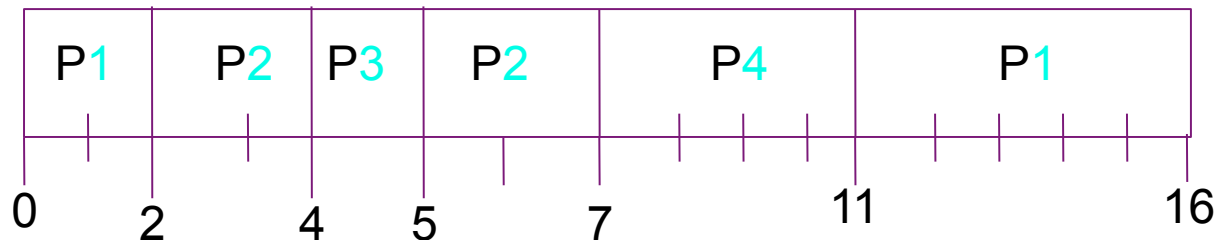


Esempio SJF/preemptive

Process	Arrival Time	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

Average waiting time = $(0 + 6 + 3 + 7)/4 = 4$

Average Turnaround = $(9 + 5 + 1 + 7)/4 = 5,5$



FCFS vs SJF

Dati n processi con lo stesso tempo di arrivo e $t_{\text{exe}} = a_j$, il processo j -esimo dovrà aspettare $a_1 + a_2 + \dots + a_{j-1}$

Quindi:

$$T_{\text{medio di attesa}} = a_1 + (a_1 + a_2) + (a_1 + a_2 + a_3) + \dots + a_n = [(n-1)a_1 + (n-2)a_2 + \dots + a_{n-1}]/n$$

che risulta minimizzato quando $a_1 < a_2 < \dots < a_n$

Round robin

- Scheduling circolare (round robin sono le petizioni in cui le firme si mettono intorno, in modo che non si capisca chi è il primo)
- Preemptive
- Ad ogni processo in attesa di esecuzione viene assegnato un intervallo di tempo detto quanto
- Terminato il quanto, il processo, se non ha terminato la sua esecuzione, viene rimesso in coda

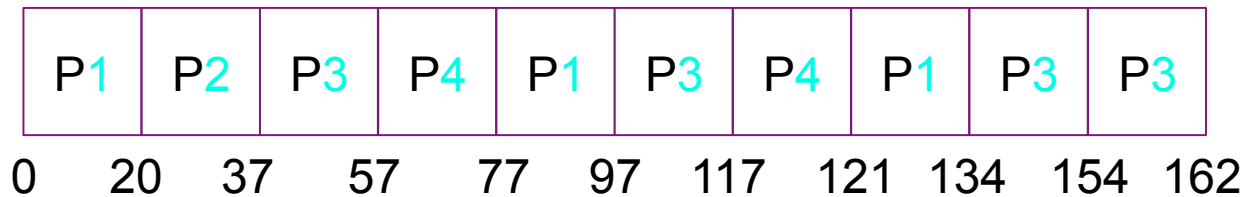
RR

- Problema: la dimensione del quanto di tempo
- Il quanto di tempo è “tempo sprecato” per il context switch
 - Quanto di tempo troppo breve: scarsa efficienza della CPU
 - lavoro utile vs amministrazione
 - Quanto di tempo troppo lungo: RR degenera in FCFS
 - degenerano le prestazioni
 - Quanto di tempo poco più lungo del CPU burst
 - la prelazione si riduce, migliorando le prestazioni

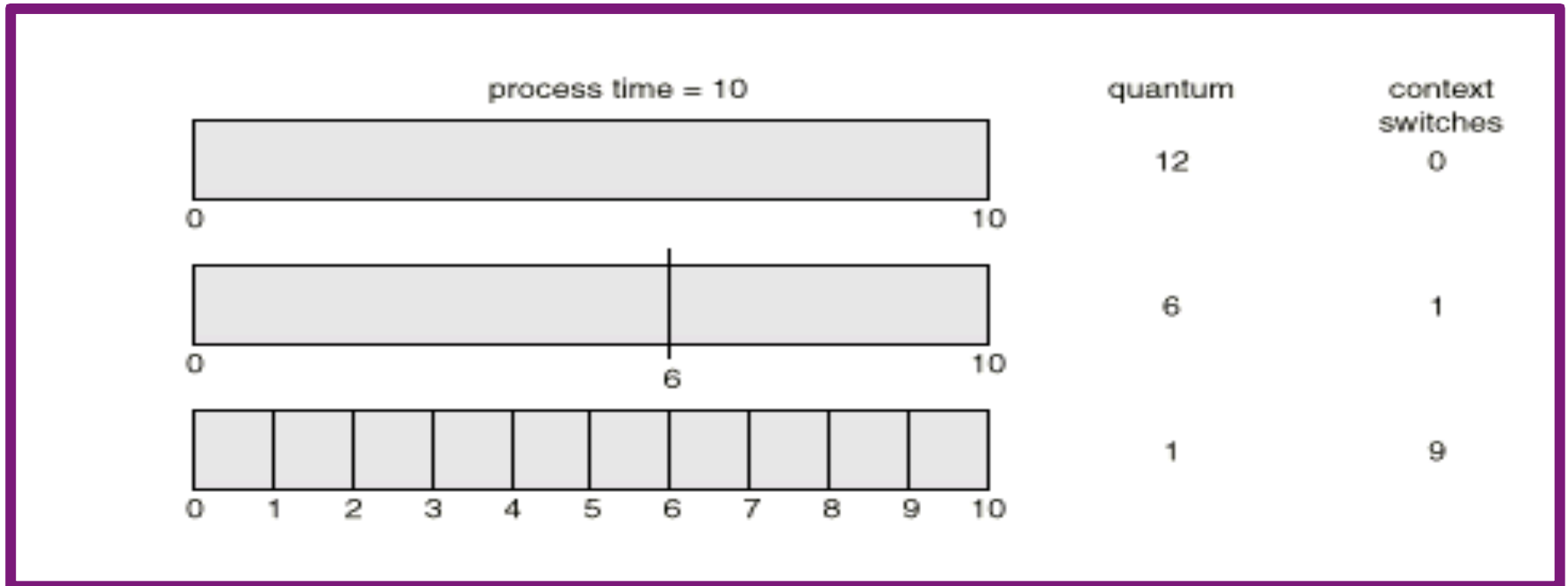
Esempio (quanto = 20)

<u>Process</u>	<u>Burst Time</u>
P_1	53
P_2	17
P_3	68
P_4	24

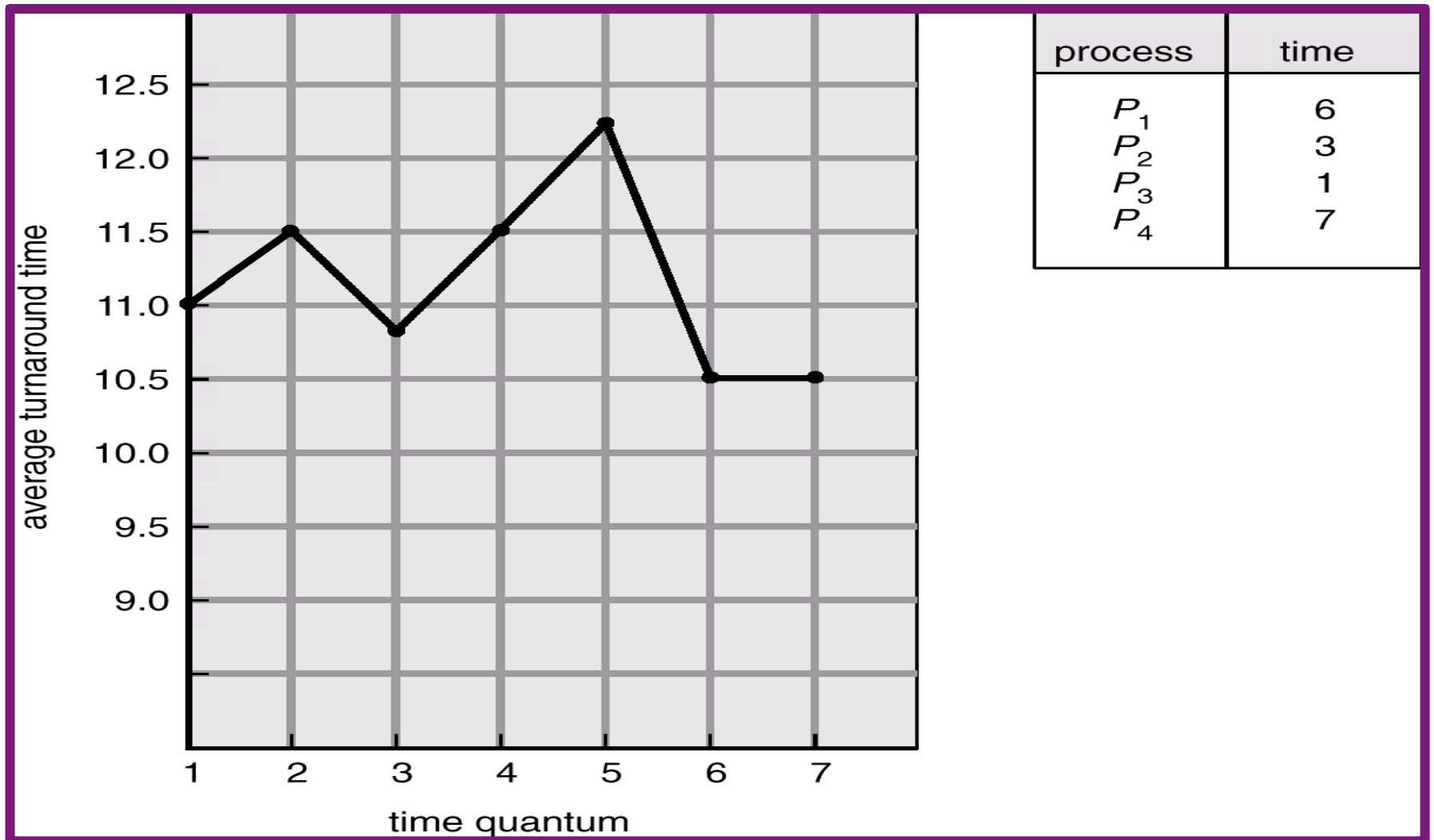
Average Waiting time = $(117 + 20 + 117 + 97) / 4$



RR



RR



Esercizio 1

Si supponga che i seguenti processi arrivino nel sistema al tempo indicato. Ogni processo deve essere eseguito (CPU time) per la quantità di tempo indicata.

Processo	Tempo di arrivo	CPU burst
P1	0.0	5
P2	0.7	3
P3	4.0	2

Qual è il tempo medio di attesa di questi processi quando vengono usati rispettivamente gli algoritmi di scheduling: FCFS, SJF, nella versione non-preemptive RR (con quanto di tempo 2 e context switch = 0.1).

- a) 2.76, 2.43, 3.7
- b) 4.33, 4, 3.66
- c) 4.33, 4.33, 2
- d) 6.1, 5.76, 6.76

Esercizio 2

Si supponga che i seguenti processi arrivino nel sistema al tempo e nell'ordine indicato. Ogni processo deve essere eseguito (CPU time) per la quantità di tempo indicata

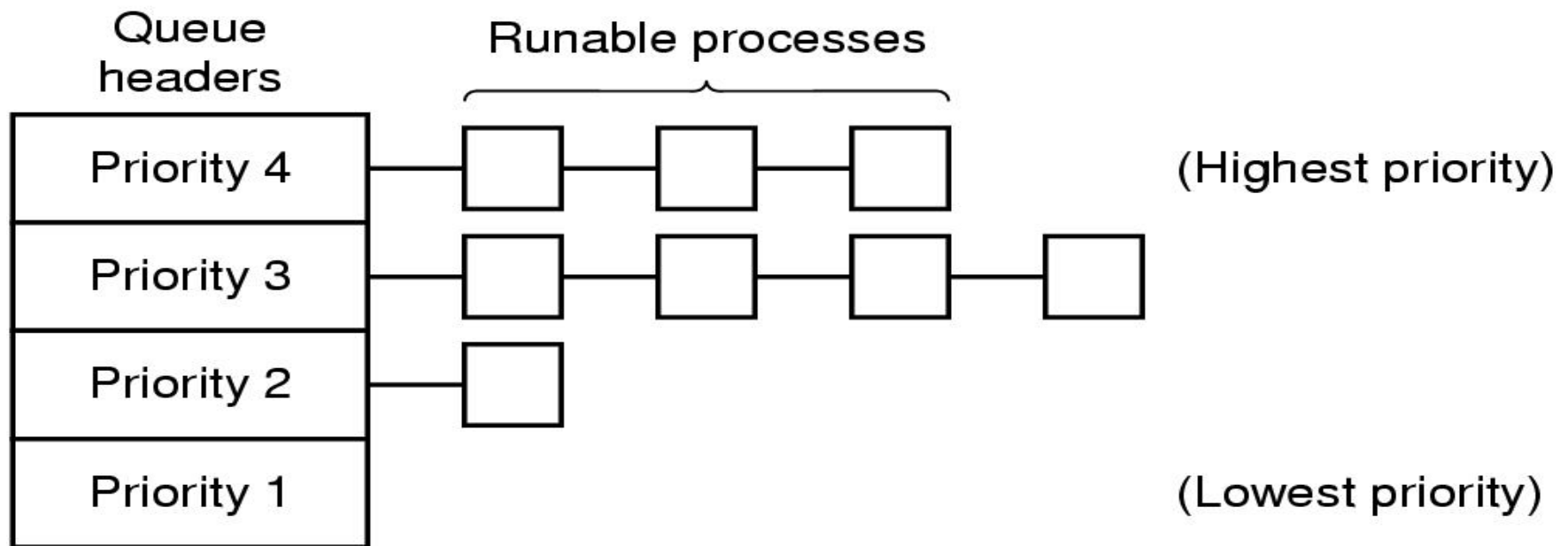
Processo	Tempo di arrivo	CPU burst
P1	0.0	5
P2	1.0	3
P3	3.0	6
P4	2.0	4

Qual è il tempo medio di attesa di questi processi quando vengono usati rispettivamente gli algoritmi di scheduling: FCFS, SJF, RR (con quanto di tempo 2 e context switch = 0)

- a) 6,3; 6,3; 8,5
- b) 6,3; 5,3; 8,5
- c) 8,2; 8,2; 8,5
- d) nessuna delle risposte fornite è corretta

Scheduling con priorità

- SJF è un caso speciale di scheduling con priorità
 - priorità = l'inverso della lunghezza stimata



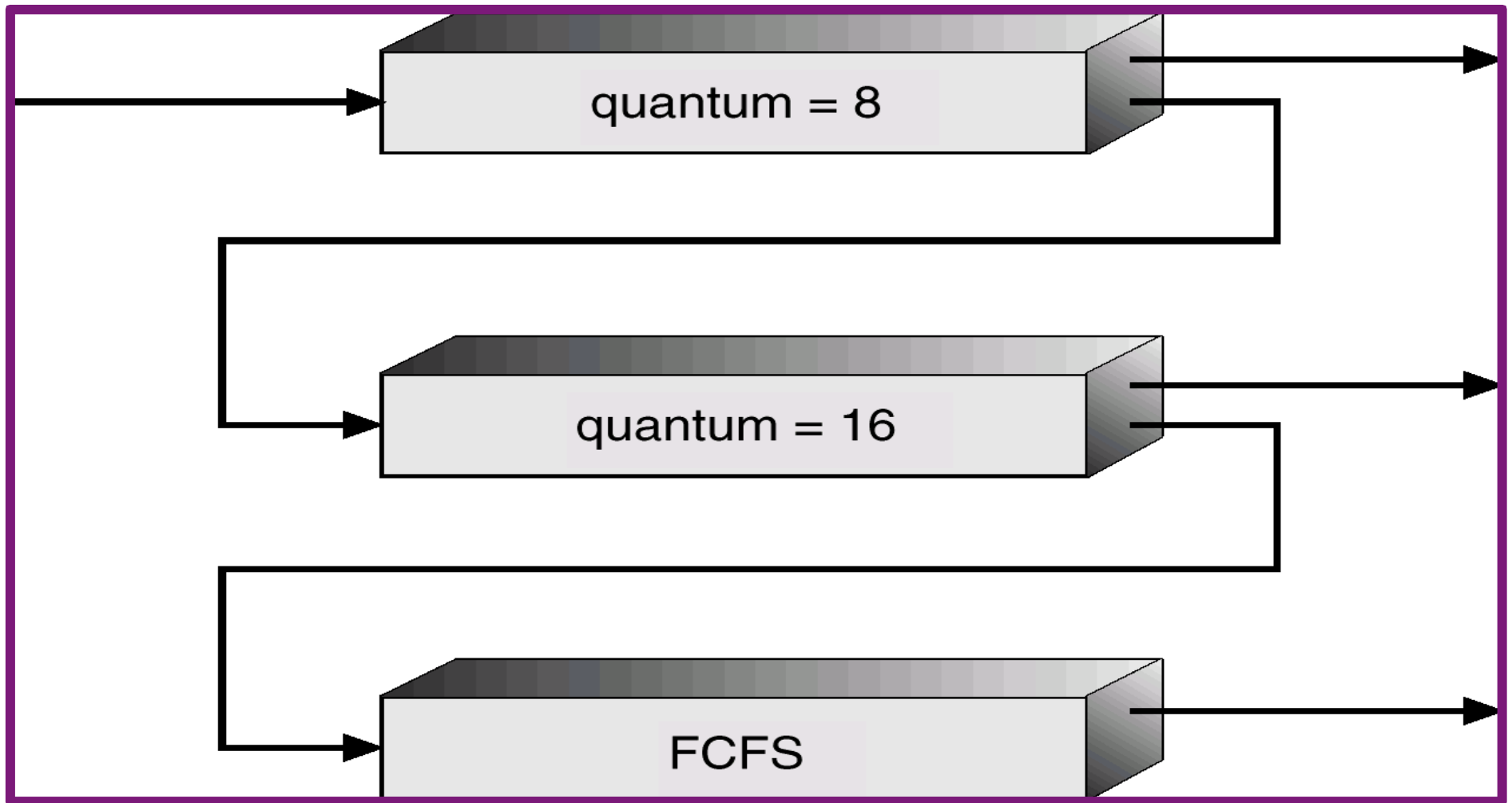
Scheduling con priorità

- Problema: la starvation dei processi a bassa priorità
- Una soluzione a questo problema è data dalla procedura di aging
 - La priorità assegnata ad un processo varia nel tempo in funzione della sua permanenza nel sistema e del tempo di CPU consumato

Scheduling a code multiple

- Due code separate
 - foreground
 - background
- Due algoritmi separati
 - foreground: RR
 - background: FCFS
- Scheduling fra le code
 - fg-bg; time slice 80%fg, 20% bg

Code multiple con feedback



Sistemi real time

- La correttezza del sistema non dipende solo dai risultati dei singoli processi ma anche dagli istanti di tempo in cui questi risultati sono prodotti
- I processi reagiscono ad eventi che si verificano nel mondo esterno
- Questi eventi si verificano in “tempo reale” e i processi devono essere in grado di gestirli in tempo debito

Sistemi Real-Time

- Controllo di esperimenti di laboratorio
- Sistemi per il controllo di processo
- Robot
- Controllo traffico aereo
- Telecomunicazioni
- Sistemi di controllo militari

Proprietà di un Real-Time OS

- **Determinismo**
 - Tutte le operazioni svolte nel sistema devono essere svolte in istanti di tempo fissati e predeterminati oppure entro intervalli di tempo predeterminati
 - L'intervallo di tempo richiesto dal SO per rispondere ad un Interrupt deve essere superiormente limitato

Caratteristiche

- Context switch molto veloce
- Dimensioni del sistema contenute
- Capacità nel rispondere molto velocemente ad interrupt esterni
- Multitasking con primitive di interprocess communication come semafori, signals e eventi
- Files in grado di memorizzare grosse quantità di dati ad alta velocità

Deadline Scheduling

- Informazioni usate
 - Ready time
 - Starting deadline
 - Completion deadline
 - Processing time
 - Resource requirements
 - Priority
 - Subtask scheduler

Esempio

Table 10.2 Execution Profile of Two Periodic Tasks

Process	Arrival Time	Execution Time	Ending Deadline
A(1)	0	10	20
A(2)	20	10	40
A(3)	40	10	60
A(4)	60	10	80
A(5)	80	10	100
•	•	•	•
•	•	•	•
•	•	•	•
B(1)	0	25	50
B(2)	50	25	100
•	•	•	•
•	•	•	•
•	•	•	•

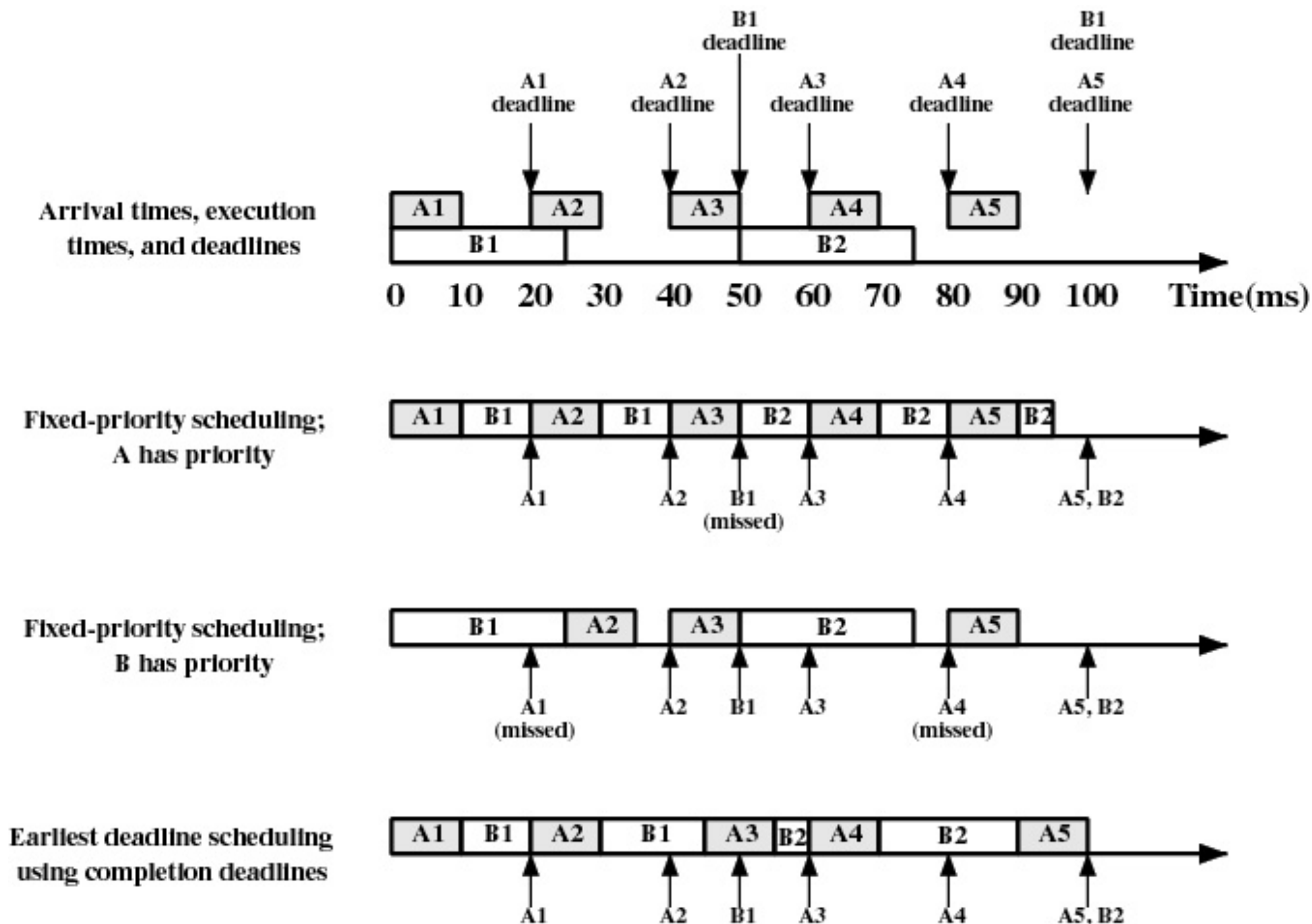


Figure 10.5 Scheduling of Periodic Real-time Tasks with Completion Deadlines

Scheduling in sistemi real-time

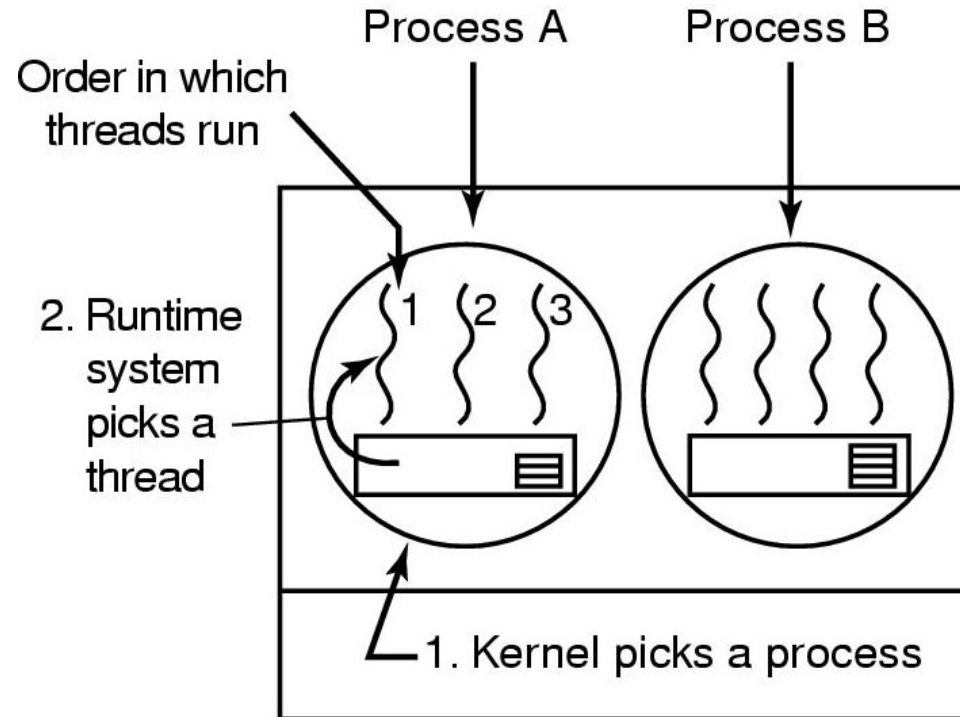
- Sistemi real-time in cui la schedulazione è possibile
 - Dati
 - m eventi periodici
 - L'evento i si verifica con periodo P_i e richiede C_i secondi di CPU
 - Il carico può essere gestito solo se

$$\sum_{i=1}^m \frac{C_i}{P_i} \leq 1$$

Scheduling in sistemi real-time

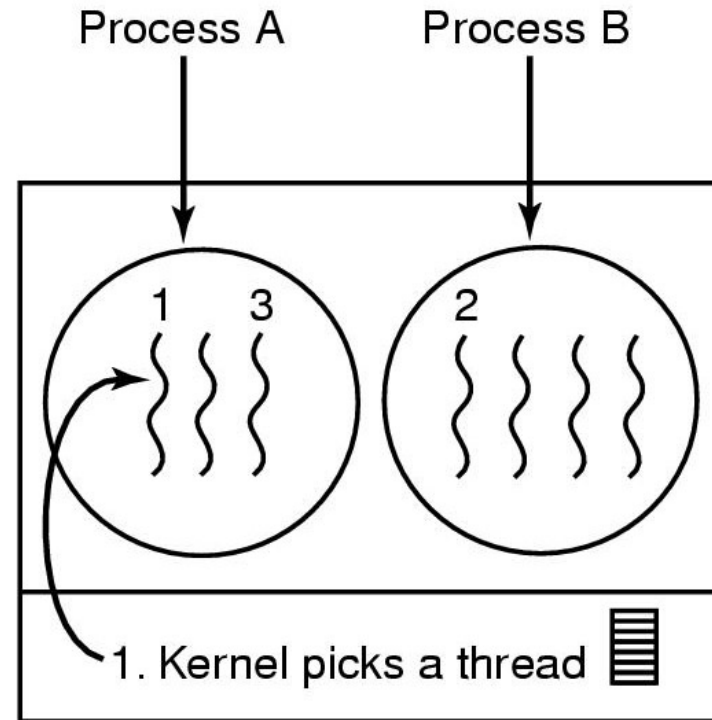
- Esempio
 - Si consideri un sistema soft real-time caratterizzato da tre eventi periodici rispettivamente con periodo 100, 200, 500 ms
 - Ciascuno richiede l'intervento della CPU per 50, 30, 100 ms, rispettivamente
 - È possibile ottenere uno scheduling di tali processi?
 - Se aggiungiamo un processo con periodo 1s, quali caratteristiche deve possedere per poter essere a sua volta schedulato?

Scheduling dei thread (1)



- Scheduling possibile per user-level thread
 - Quanto per processo 50 msec
 - Un thread esegue fino a che non cede volontariamente la CPU perché il run-time non può interromperlo

Scheduling dei thread (2)



Possible: A1, A2, A3, A1, A2, A3

Also possible: A1, B1, A2, B2, A3, B3

- Scheduling possibile per kernel-level threads
 - Quanto di tempo per processo 50 msec
 - I thread vengono eseguiti in burst di CPU di 5 msec