Scheduling della CPU

Lo scheduling

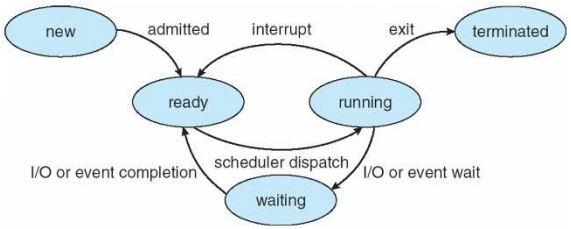
Stefano Quer
Dipartimento di Automatica e Informatica
Politecnico di Torino

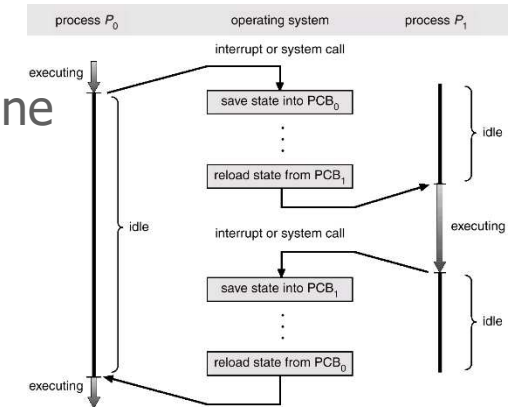
Concetti fondamentali

- Uno degli obbiettivi della multiprogrammazione è quello di massimizzare l'utilizzo delle risorse e in particolare della CPU
- Per raggiungere tale obbiettivo ogni CPU viene assegnata a più task (i.e., processo o thread)
  - Lo scheduler deve decidere quale **algoritmo** utilizzare per assegnare la CPU a un task
  - Le prestazioni dello scheduler sono valutate tramite **funzioni di costo**
  - Applicazioni diverse richiedono algoritmi e funzioni di costo diverse

Algoritmi

- Procedimento generale di scheduling
  - La CPU viene assegnata a un task
  - Qualora il processo entri in uno stato di attesa, termini, venga ricevuto un interrupt, etc., è necessario effettuare un context switching
  - Per ogni context switching
    - Il task in running viene spostato nella coda di ready
    - Un task nella coda di ready viene spostato allo stato di running





Algoritmi

Estensione

Algoritmi senza prelazione	Algoritmi con prelazione
FCFS (First Come First Served) Scheduling in ordine di arrivo	RR (Round Robin) Scheduling circolare
SJF (Shortest Job First) Scheduling per brevità	SRTF (Shortest Remaining Time First) Scheduling per tempo rimanente minimo
PS (Priority Scheduling) Scheduling per priorità	
MQS (Multilevel Queue Scheduling) Scheduling a code multi-livello	

Senza prelazione (non preemptive)
La CPU **non** può essere sottratta a un altro task, i.e., il task deve rilasciare la CPU volontariamente

Con prelazione (preemptive)
La CPU può essere sottratta a un altro task ovvero si definiscono dei CPU burst (tempi di esecuzione massimi) al termine dei quali la CPU deve essere rilasciata

Funzioni di costo

Funzione di costo	Descrizione	Ottimo
Utilizzo della CPU (CPU utilization)	Percentuale di utilizzo della CPU	[0-100%] Massimo
Produttività (Throughput)	Numero di processi completati nell'unità di tempo	Massimo
Tempo di completamento (Turnaround time)	Tempo che trascorre dalla sottomissione al completamento dell'esecuzione	Minimo
Tempo di attesa (Waiting time)	**Tempo totale passato nella coda ready (somma dei tempi trascorsi in coda)**	**Minimo**
Tempo di risposta (Response time)	Tempo intercorso tra la sottomissione e la prima risposta prodotta	Minimo

FCFS (First-Come First-Served)

- Algoritmo
  - La CPU è assegnata ai task seguendo l'ordine con cui la richiedono
    - I task vengono gestiti attraverso una coda FIFO
      - Un task in arrivo viene inserito in coda
      - Un task da servire viene estratto dalla testa
  - Lo scheduling può essere illustrato mediante un **diagramma (o carta) di Gantt** (1917)
    - Diagramma a barre che illustra la pianificazione (tempi di inizio e fine) delle attività

Ricordare: Nessun task viene interrotto, ovvero la CPU può **solo** essere rilasciata volontariamente

FCFS (First-Come First-Served)

P	Tempo arrivo	Burst Time
P_1	0	24
P_2	0	3
P_3	0	3

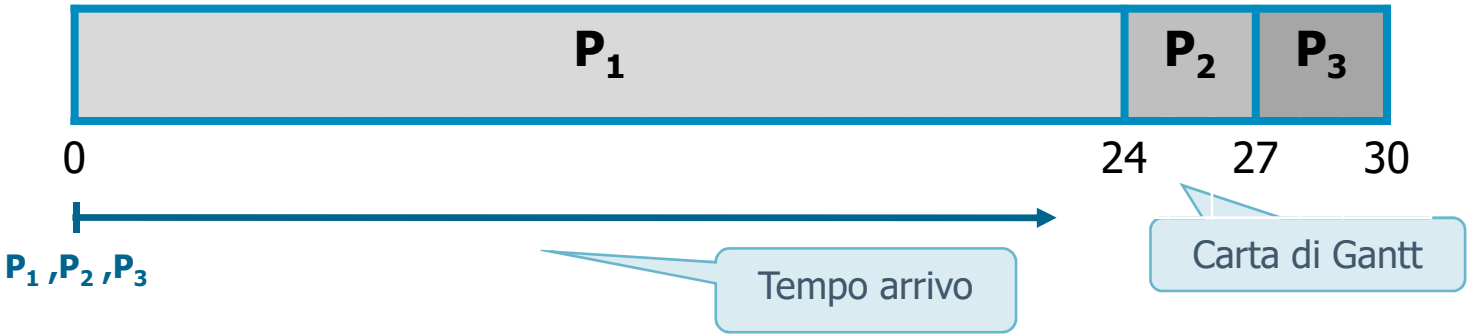
Esempio 1

P	Tempo di attesa
P_1	(0-0) = 0
P_2	(24-0) = 24
P_3	(27-0) = 27

Tempo di attesa medio: (0+24+27)/3=17

Ordine di arrivo dei task

Durata prevista (unità di tempo)



FCFS (First-Come First-Served)

P	Tempo arrivo	Burst Time
P_2	0	3
P_3	0	3
P_1	0	24

Esempio 2

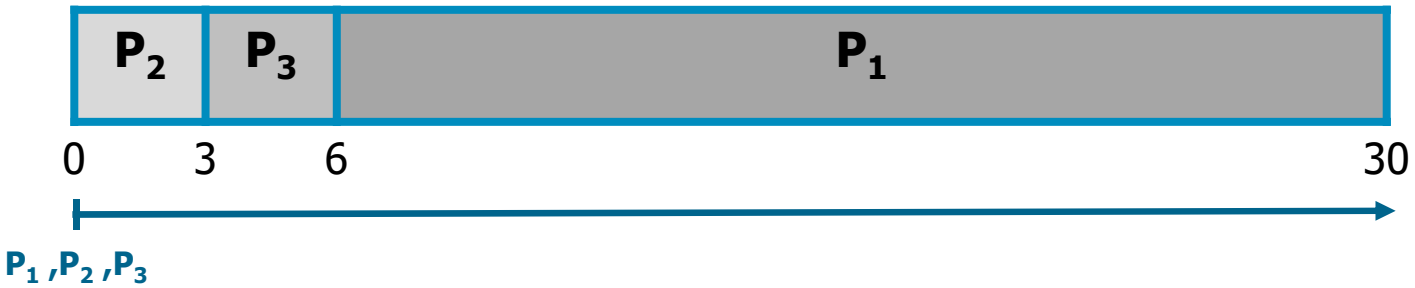
P	Tempo di attesa
P_1	(6-0)=6
P_2	(0-0)=0
P_3	(3-0)=3

Tempo di attesa medio: (6+0+3)/3=3

Ordine di arrivo dei task

Durata prevista (unià di tempo)

Molto **meglio** del precedente: processi lunghi ritardano quelli brevi



FCFS (First-Come First-Served)

- Pregi
  - Facile da comprendere
  - Facile da implementare
- Difetti
  - Tempi di attesa
    - Relativamente lunghi
    - Variabili e non ottimi
  - Inadatto per sistemi real-time (no prelazione)
  - Effetto convoglio
    - Task brevi in coda a task lunghi attendono molto tempo inutilmente

SJF (Shortest-Job First)

- Algoritmo
  - A ogni task viene associata la durata della sua prossima richiesta (**next CPU burst**)
  - I task vengono schedulati in ordine di durata della loro prossima richiesta
    - Scheduling per brevità
    - In caso di ex-aequo si applica lo scheduling FCFS

SJF (Shortest-Job First)

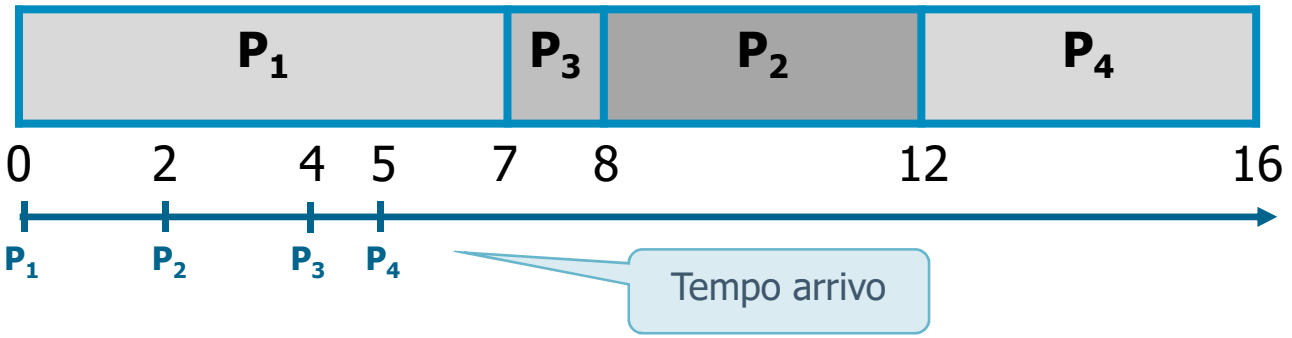
P	Tempo arrivo	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

Ordine di arrivo dei task

Durata prevista

P	Tempo di attesa
P_1	(0-0) = 0
P_2	(8-2) = 6
P_3	(7-4) = 3
P_4	(12-5) = 7

Tempo di attesa medio: (0+6+3+7)/4=4



SJF (Shortest-Job First)

- Pregi
  - Si può dimostrare che SJF è un algoritmo ottimo, utilizzando il tempo di attesa come criterio
    - Spostando i processi brevi prima di quelli lunghi il tempo di attesa dei primi diminuisce più di quanto aumenta il tempo di attesa dei secondi
- Difetti
  - Possibile l'attesa indefinita, ovvero la starvation
  - Difficoltà di applicazione derivata dall'impossibilità di conoscere a priori il comportamento futuro
    - Il tempo del burst successivo è **ignoto**
    - È possibile effettuare delle **stime** utilizzando diversi criteri, tra i quali la media esponenziale

SJF (Shortest-Job First)

- Media esponenziale

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

Stima n-esimo burst

Valore previsto per il successivo burst

Durata (reale) n-esimo burst

$\alpha = [0, 1]$
controlla il peso relativo storia recente vs passata
$\alpha = 0 \rightarrow \tau_{n+1} = \tau_n$
$\alpha = 1 \rightarrow \tau_{n+1} = t_n$

- Procedendo per sostituzione

$$\tau_{n+1} = \alpha \cdot t_n + (1-\alpha) \cdot \alpha \cdot t_{n-1} + .. + (1-\alpha)^j \cdot \alpha \cdot t_{n-j} + ... + (1-\alpha)^{n+1} \cdot \tau_0$$

  - Dato che sia α che $1-\alpha$ sono minori di 1, ogni termine successivo ha un peso minore

PS (Priority Scheduling)

- Algoritmo
  - A ogni processo viene associata la sua priorità
    - La priorità è generalmente rappresentata mediante valori interi
    - A priorità maggiore corrisponde intero minore
    - Le priorità possono essere determinate in base a criteri
      - Interni, memoria utilizzata, numero file utilizzati, etc.
      - Esterni, proprietario del task, etc.
  - La CPU viene allocata al processo con la priorità maggiore
    - PS = SJF con la durata del CPU burst sostituita dalla priorità

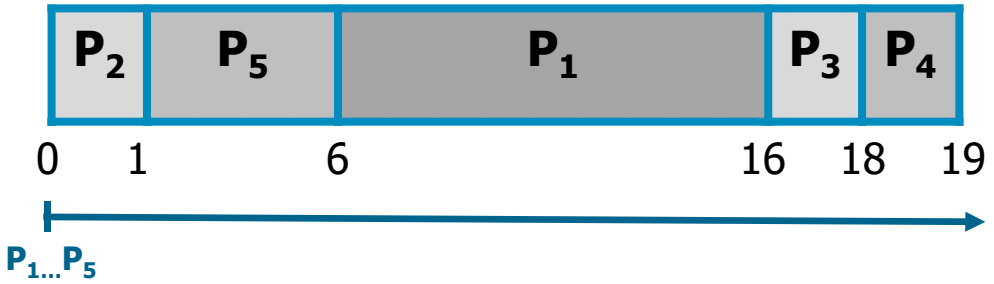
Min o Max Heap ...

PS (Priority Scheduling)

P	Tempo arrivo	Priorità	Burst Time
P_1	0	3	10
P_2	0	1	1
P_3	0	4	2
P_4	0	5	1
P_5	0	2	5

P	Tempo di attesa
P_1	(6-0) = 6
P_2	(0-0) = 0
P_3	(16-0) = 16
P_4	(18-0) = 18
P_5	(1-0) = 1

Tempo di attesa medio:
(6+0+16+18+1)/5=8.2



PS (Priority Scheduling)

- Difetti
  - Possibile l'attesa indefinita, ovvero la starvation
    - Nei sistemi molto carichi, task con bassa priorità possono attendere per sempre
      - MIT: IBM fermato nel 1973 aveva in coda processo dal 1967
    - Una possibile soluzione alla starvation è l'invecchiamento (aging) dei task
      - La priorità viene incrementata gradualmente con il passare del tempo

RR (Round Robin)

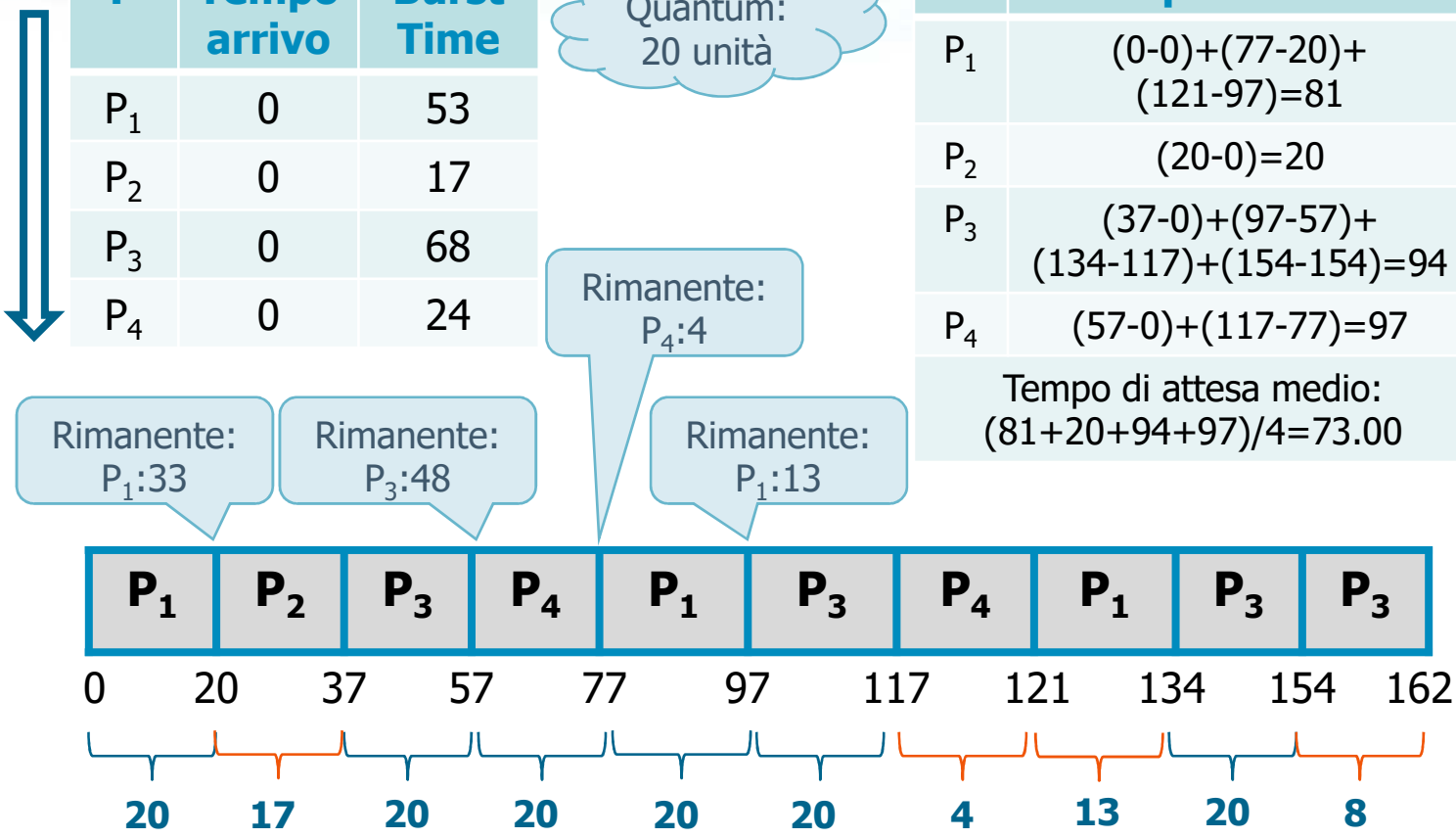
- Round Robin o scheduling circolare
- Versione di FCFS che permette la prelazione
- Algoritmo
  - L'utilizzo della CPU viene suddiviso in "time quantum" (porzioni temporali)
  - Ogni task riceve la CPU per un tempo massimo pari al quantum e poi viene inserito nuovamente nella ready queue
  - La ready queue è gestita con modalità FIFO
    - Eventuali nuovi processi sono aggiunti alla coda
- Progettato appositamente per sistemi time sharing e real time

RR (Round Robin)

P	Tempo arrivo	Burst Time
P_1	0	53
P_2	0	17
P_3	0	68
P_4	0	24

Quantum: 20 unità

P	Tempo di attesa
P_1	(0-0)+(77-20)+ (121-97)=81
P_2	(20-0)=20
P_3	(37-0)+(97-57)+ (134-117)+(154-154)=94
P_4	(57-0)+(117-77)=97
Tempo di attesa medio: (81+20+94+97)/4=73.00	



RR (Round Robin)

- Difetti
  - Il tempo di attesa medio è relativamente lungo
  - Notevole dipendenza delle prestazioni dalla durata del quanto di tempo
    - Quantum lungo: RR degenera in FCFS
    - Quantum corto: vengono effettuati troppi context switching e i tempi di commutazione/gestione risultano molto elevati

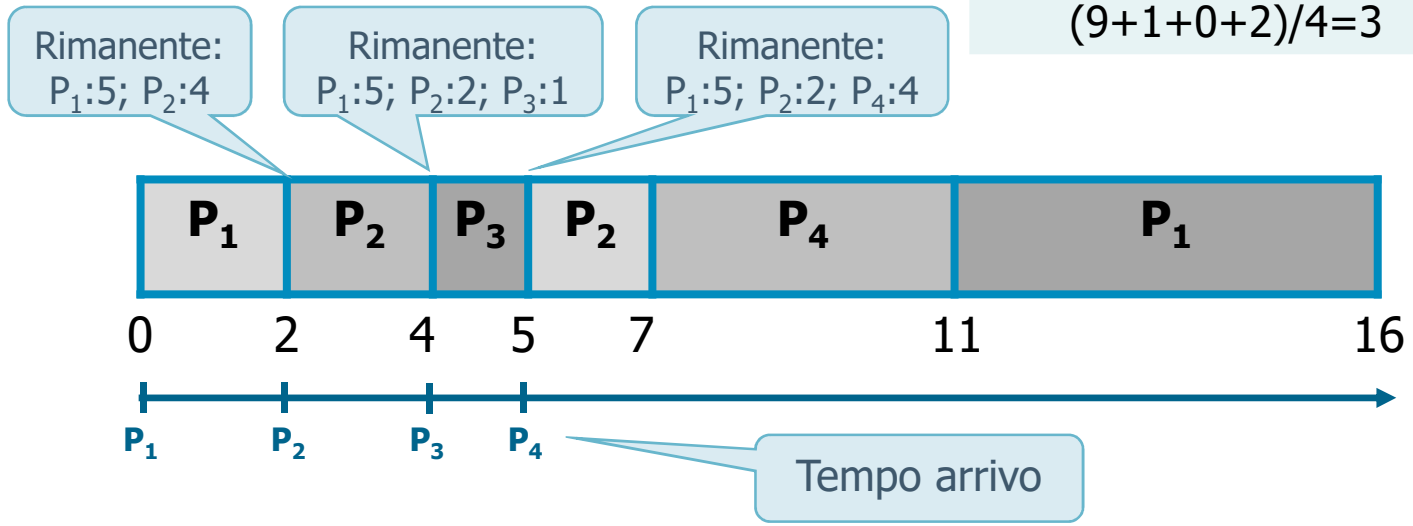
SRTF (Shortest-Remaining-Time First)

- Versione di SJF che permette la prelazione
- Algoritmo
  - Sui procede a uno scheduling SJF ma
  - Se viene sottomesso un processo con burst più breve di quello in esecuzione la CPU viene prelazionata
- Caratteristiche simili allo scheduling SJF

SRTF (Shortest-Remaining-Time First)

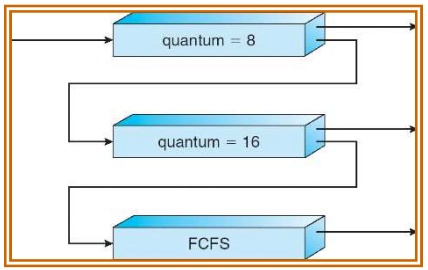
P	Tempo arrivo	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

P	Tempo di attesa
P_1	(0-0)+(11-2) = 9
P_2	(2-2)+(5-4) = 1
P_3	(4-4) = 0
P_4	(7-5) = 2
Tempo di attesa medio: (9+1+0+2)/4=3	



MQS (Multilevel Queue Scheduling)

- Applicato a situazioni in cui i task possono essere classificati in gruppi diversi
  - Foreground, background, di sistema, etc.
- Algoritmo
  - La ready queue viene suddivisa in code diverse
  - Ogni coda può essere gestita con il proprio algoritmo di scheduling
  - Può essere modificato per permettere il trasferimento dei task tra le varie code
    - MQS con retroazione



Considerazioni aggiuntive

- Lo scheduler è un task che deve essere schedulato in maniera simile agli altri task
  - Nello scheduling senza prelazione
    - Lo scheduler è invocato ogni volta che un programma termina o lascia il controllo
  - Nello scheduling con prelazione
    - Lo scheduler è invocato periodicamente da un interrupt periodico della CPU
    - Gli altri task non possono prevenire questo procedimento

Considerazioni aggiuntive

- Lo scheduling può essere effettuato a livello di processi oppure di thread
  - Se il SO prevede thread normalmente lo scheduling viene effettuato a livello di thread, trascurando i processi
- Scheduling dei thread
  - Il SO gestisce i T a livello kernel e ignora i T a livello utente (gestiti da una libreria)
  - Quindi lo scheduling può essere effettuato solo per i T a livello kernel (se esistono)

Considerazioni aggiuntive

- Scheduling per sistemi multiprocessori
  - Tutte le esemplificazioni precedenti sono state fatte supponendo l'esistenza di una sola CPU
  - Nel caso siano disponibili più unità elaborative il carico può essere distribuito
    - Il **bilanciamento del carico** è automatico per SO con code di attesa comuni a tutti i processori
  - Esistono diversi schemi
    - Multi-elaborazione **asimmetrica**: un processore master distribuisce il carico tra i processori server
    - Mullti-elaborazione **simmetrica**: ciascun processore provvede al proprio scheduling

Considerazioni aggiuntive

- Scheduling per sistemi real-time
  - Tentano di rispondere in tempo reale al verificarsi di eventi (con deadline)
    - Gli eventi guidano il comportamento dello scheduling
    - Si definisce **latenza** il tempo che intercorre tra il verificarsi dell'evento e la sua gestione
  - Esistono due tipi di sistemi real-time
    - Soft real-time
      - Danno priorità ai processi critici ma non garantiscono le tempistiche di risposta
    - Hard real-time
      - Si garantisce l'esecuzione dei task entro un tempo massimo limite

Esame del 17.02.2017

Esercizio

- Si consideri il seguente insieme di processi

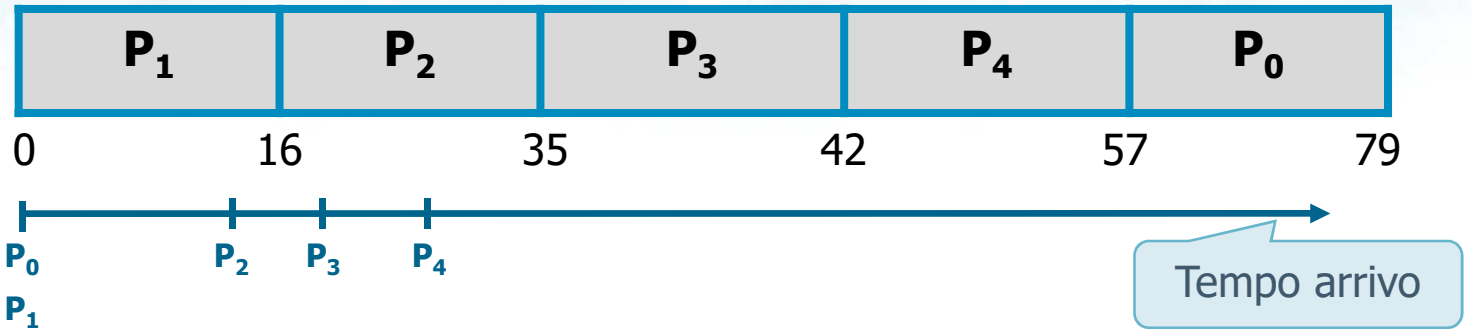
P	Tempo arrivo	Burst Time	Priorità
P_0	0	22	5
P_1	0	16	2
P_2	15	19	4
P_3	17	7	1
P_4	25	15	1

  - Rappresentare il diagramma di Gantt per gli algoritmi PS (Priority Scheduling), RR (Round Robin) e SRTF (Shortest Remaining Time First)
  - Calcolare il tempo di attesa medio

Ordine di arrivo dei task

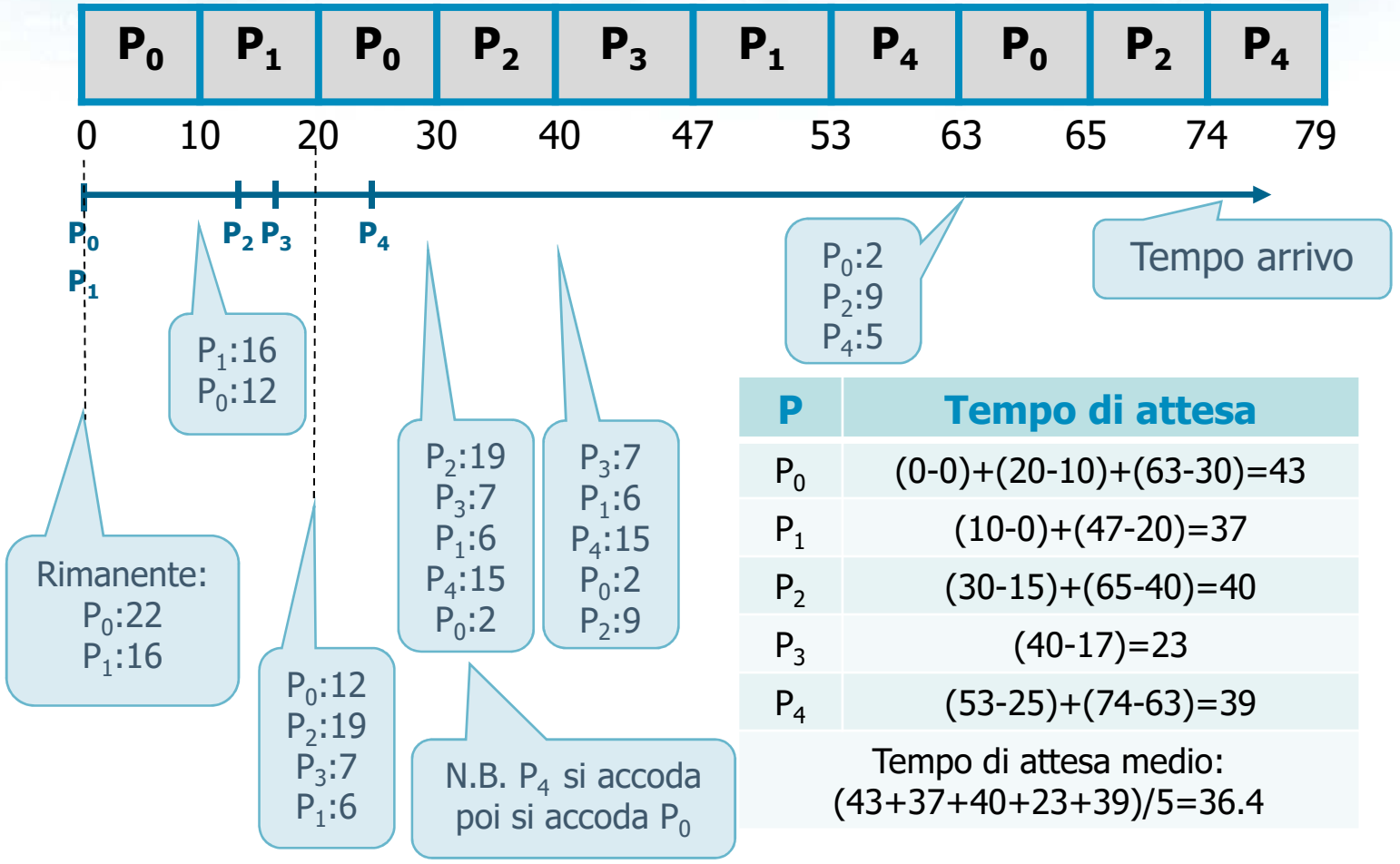
Priorità massima = valore inferiore
Quantum temporale = 10

Esercizio: PS



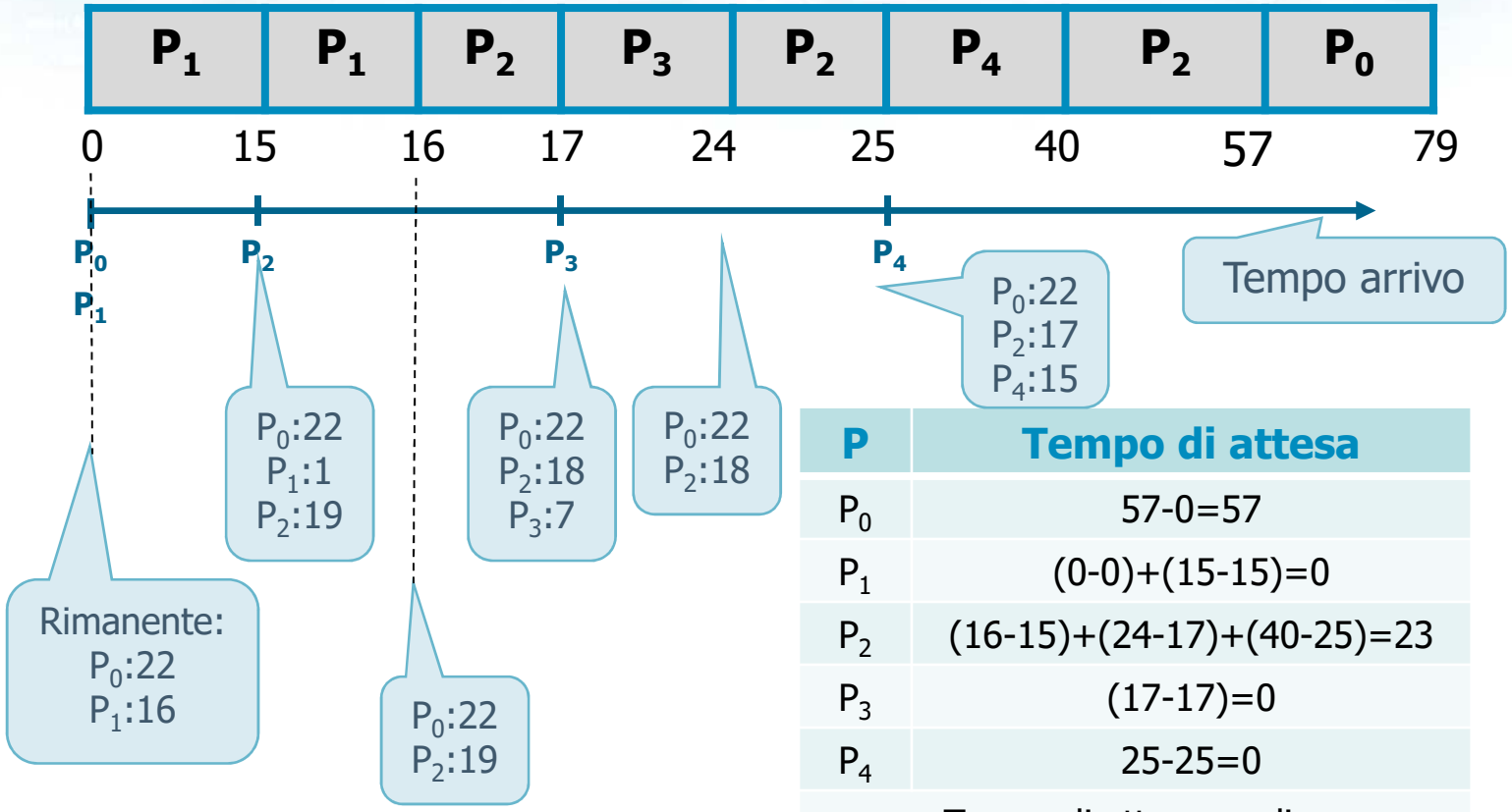
P	Tempo di attesa
P_0	57-0=57
P_1	0-0=0
P_2	16-15=1
P_3	35-17=18
P_4	42-25=17
Tempo di attesa medio: (57+0+1+18+17)/5=18.6	

Esercizio: RR



P	Tempo di attesa
P_0	(0-0)+(20-10)+(63-30)=43
P_1	(10-0)+(47-20)=37
P_2	(30-15)+(65-40)=40
P_3	(40-17)=23
P_4	(53-25)+(74-63)=39
Tempo di attesa medio: (43+37+40+23+39)/5=36.4	

Esercizio: SRTF



P	Tempo di attesa
P_0	57-0=57
P_1	(0-0)+(15-15)=0
P_2	(16-15)+(24-17)+(40-25)=23
P_3	(17-17)=0
P_4	25-25=0
Tempo di attesa medio: (57+0+23+0+0)/5=16.0	