Scheduling
nei sistemi mono-processore

Corso di Laurea in Ingegneria Informatica

Università degli Studi di Napoli Federico II

Anno Accademico 2024/2025, Canale San Giovanni

Sommario della lezione

- Sommario della lezione:
  - Tipologie di scheduler
  - Algoritmi di scheduling
  - Preemption
  - Scheduling a Priorità, Starvation

Riferimenti bibliografici

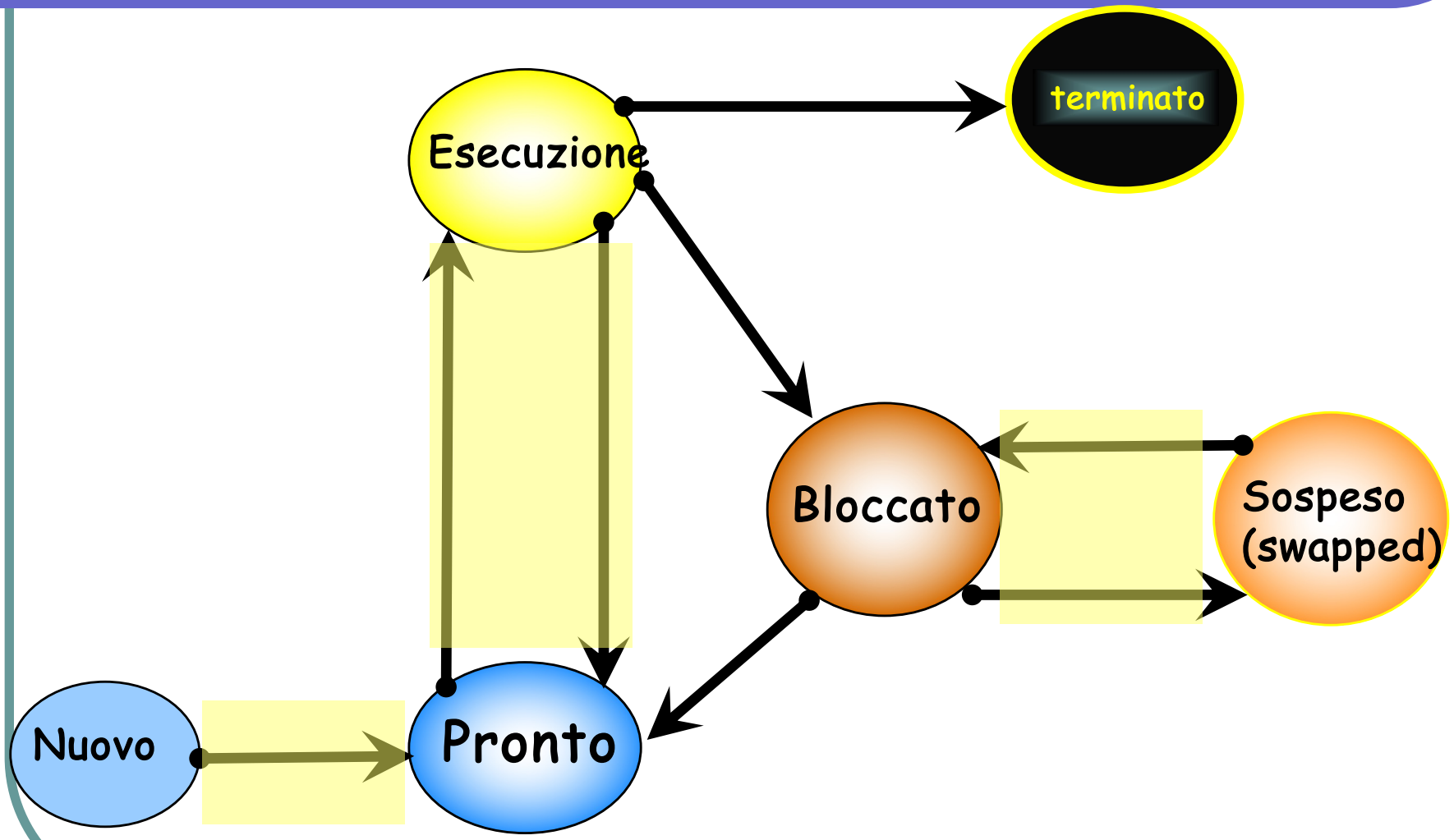
- Riferimenti
  - P. Ancilotti, M. Boari, A. Ciampolini, G. Lipari, "Sistemi Operativi", Mc-Graw-Hill (Cap.2)
  - www.ostep.org, Cap. 7, Cap. 8
- Testi di Approfondimento
  - W. Stallings, "Operating Systems: Internals and Design Principles (6th Edition) ", Pearson Education (Cap. 9)

Scheduler

Lo **scheduler** è la parte del S.O. preposta all'**assegnazione di risorse** a favore dei processi

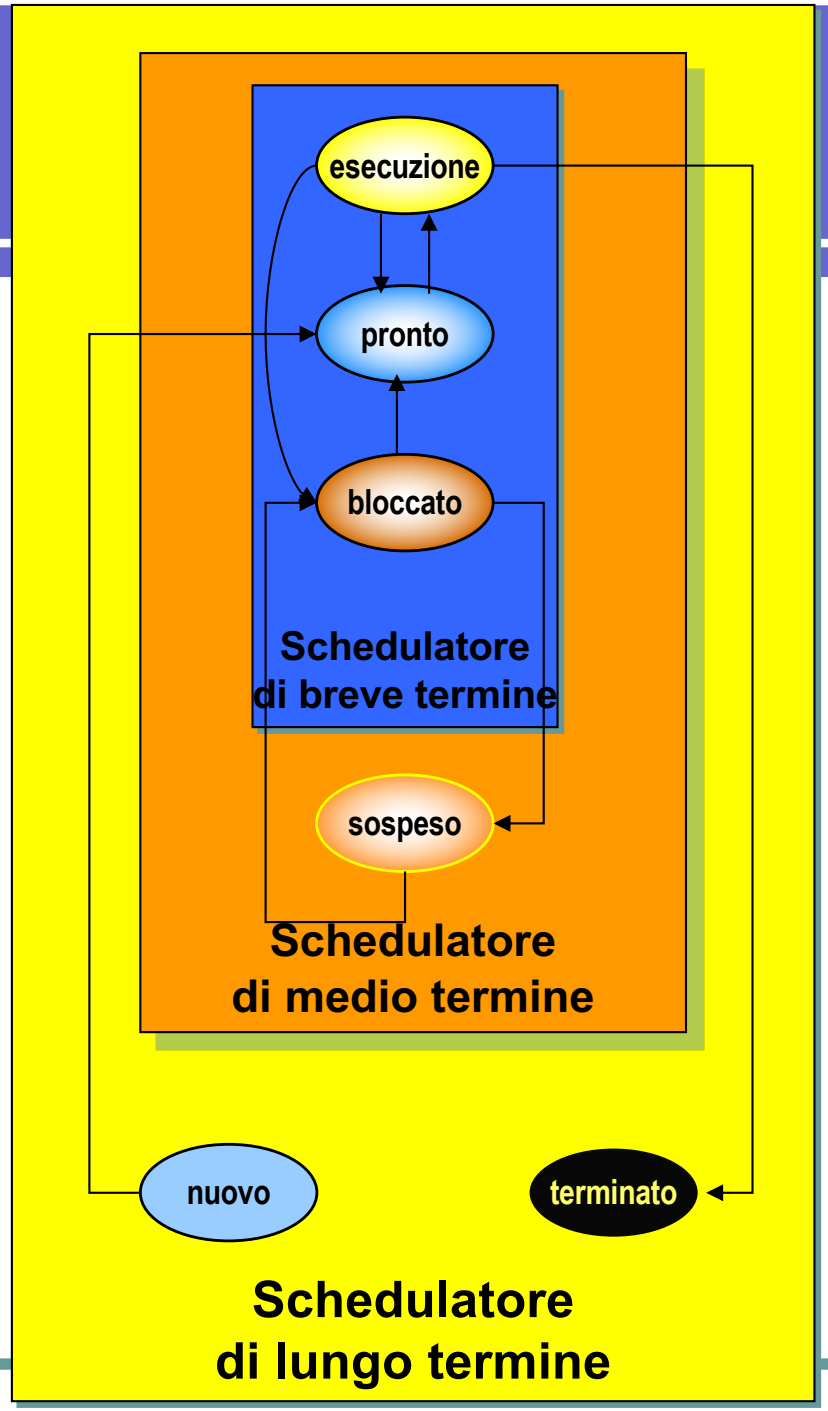
Un **algoritmo di scheduling** seleziona il processo assegnatario da una coda, in base a **vari criteri**

Schedulatore dei processi



Tipologie di scheduler

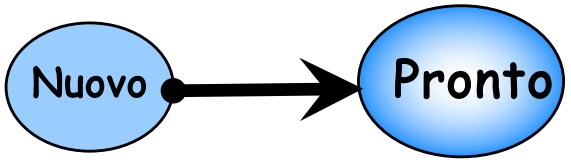
- schedulatore a breve termine, o scheduler della CPU
- schedulatore a medio termine, o schedulatore di swap (swapper)
- schedulatore a lungo termine, o schedulatore di job


esecuzione
pronto
bloccato
Schedulatore
di breve termine
sospeso
Schedulatore
di medio termine
nuovo
terminato
Schedulatore
di lungo termine

Lo scheduler di lungo termine

Lo scheduler di lungo termine determina quale programma caricare nella coda dei processi pronti del sistema

- Obiettivo: regolare la quantità di processi nel sistema (**grado di multiprogrammazione**)



Lo scheduler di lungo termine

- Possibili criteri per la scelta:
  - FIFO (first in first out)
  - Priorità
  - Tempo di esecuzione presunto
  - Requisiti di I/O
  - Tempo presunto di CPU

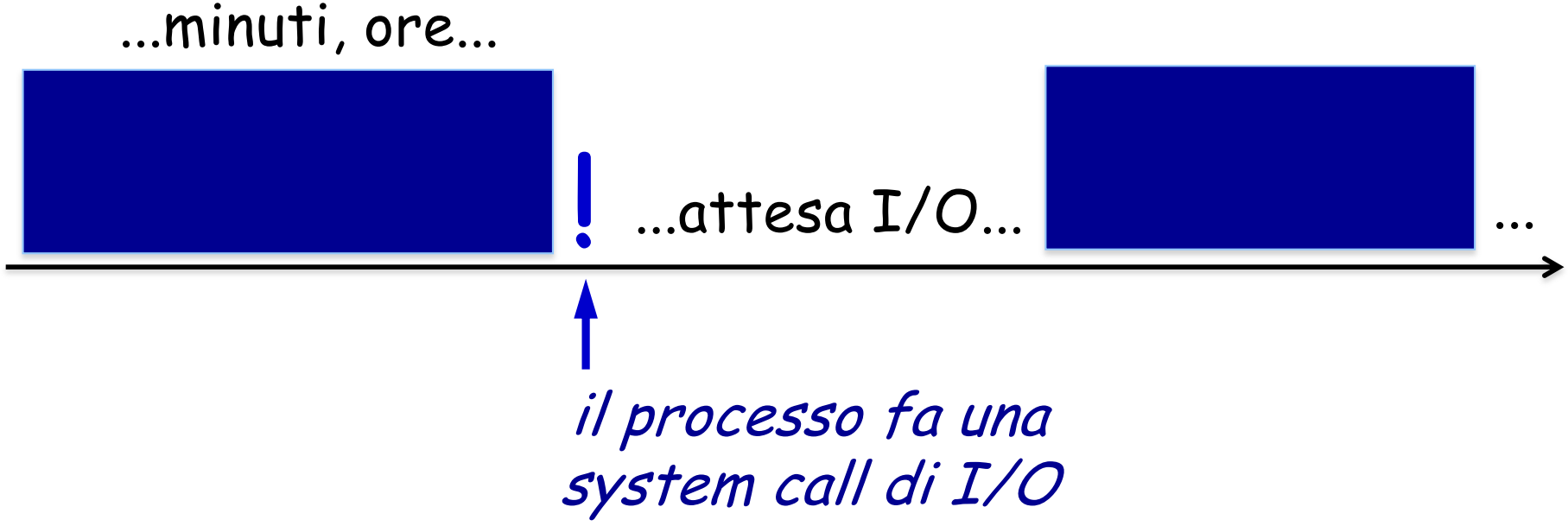
Esempio:

per aumentare l'efficienza d'utilizzo delle risorse, lo scheduler di lungo termine ammette nel sistema **un numero comparabile di processi CPU-bound e I/O-bound**

Processi CPU-bound

- Processi *CPU-bound*:
  - poche chiamate di sistema
  - tendono ad occupare la CPU per lunghi periodi (se il SO non li interrompe)
  - tipico di applicazioni batch, calcolo numerico

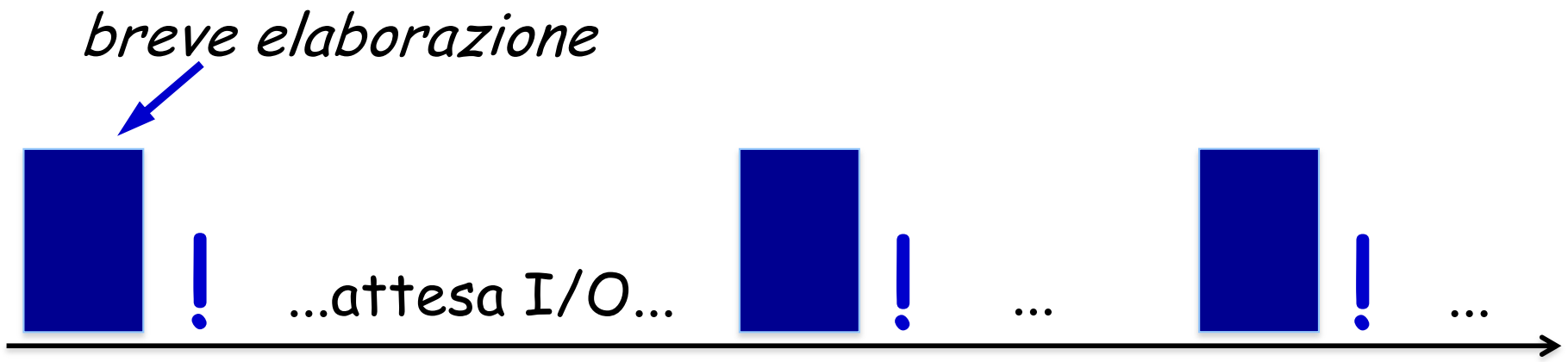
Processi CPU-bound



Processi I/O-bound

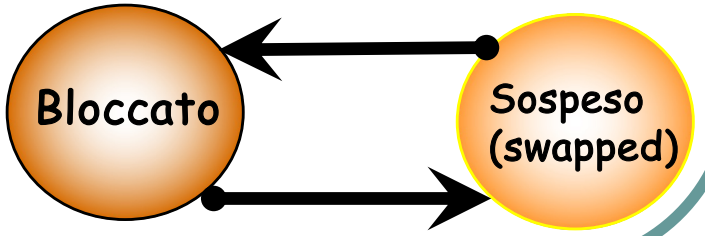
- Processi **I/O-bound**:
  - fanno frequenti chiamate di sistema
  - usano brevemente la CPU, per poi mettersi subito in attesa di I/O
  - tipico dei programmi interattivi (es. browser, editor di testo, ...)

Processi I/O-bound



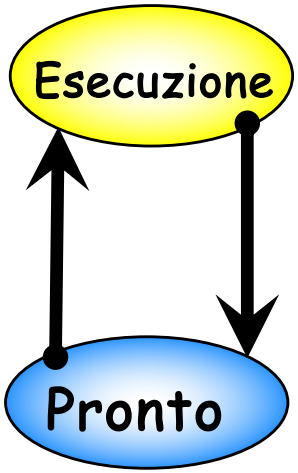
Lo scheduler di medio termine

- Ha il compito di trasferire temporaneamente processi (o parte di essi) **dalla memoria centrale alla memoria di massa** (*swap-out*) e viceversa (*swap-in*)
  - per **liberare spazio nella memoria centrale**
  - per rendere possibile il **caricamento di altri processi**
- Obiettivo: Gestire efficientemente la memoria

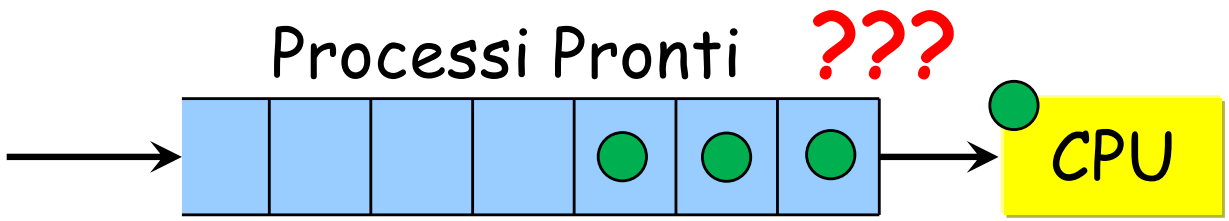


Lo scheduler di breve termine

- Ha il compito di **scegliere un processo pronto** a cui assegnare la **CPU**
- È lo scheduler attivato più frequentemente, quando il processo in esecuzione si interrompe:
  - Interruzioni del Clock (timer)
  - System call
  - ...



Lo scheduler di breve termine

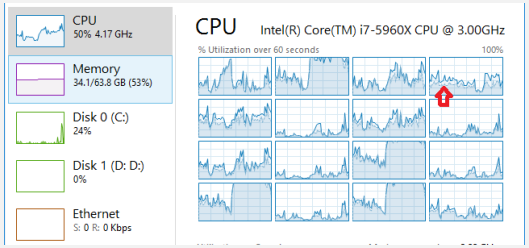


Criteri per la scelta di un algoritmo di scheduling

Lo scheduler a breve termine mira a
ottimizzare dei criteri di prestazione

Criteri user-oriented:
la percezione dell'utente e
dei singoli processi

Criteri system-oriented:
il punto di vista delle risorse e
dello amministratore del sistema



Parametri User-Oriented

Tempo di risposta:
il tempo tra l'invio di una richiesta dell'utente fino all'inizio della sua elaborazione

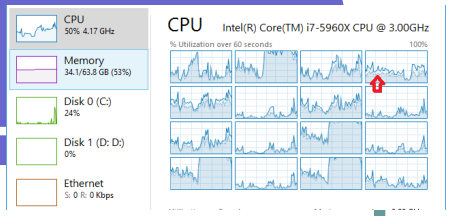
- es., avvio di un processo, inserimento di un comando

Tempo di turnaround:
il tempo tra l'invio di un processo al sistema e la sua terminazione.
Include:

- il tempo di risposta
- il tempo speso in esecuzione
- il tempo speso in attesa

Deadlines:
se l'utente indica una scadenza di completamento (deadline), si deve massimizzare la percentuale di scadenze rispettate

Parametri System-Oriented



Throughput:
produttività in termini di numero di processi completati per unità di tempo (es., 10 esecuzioni/ora)

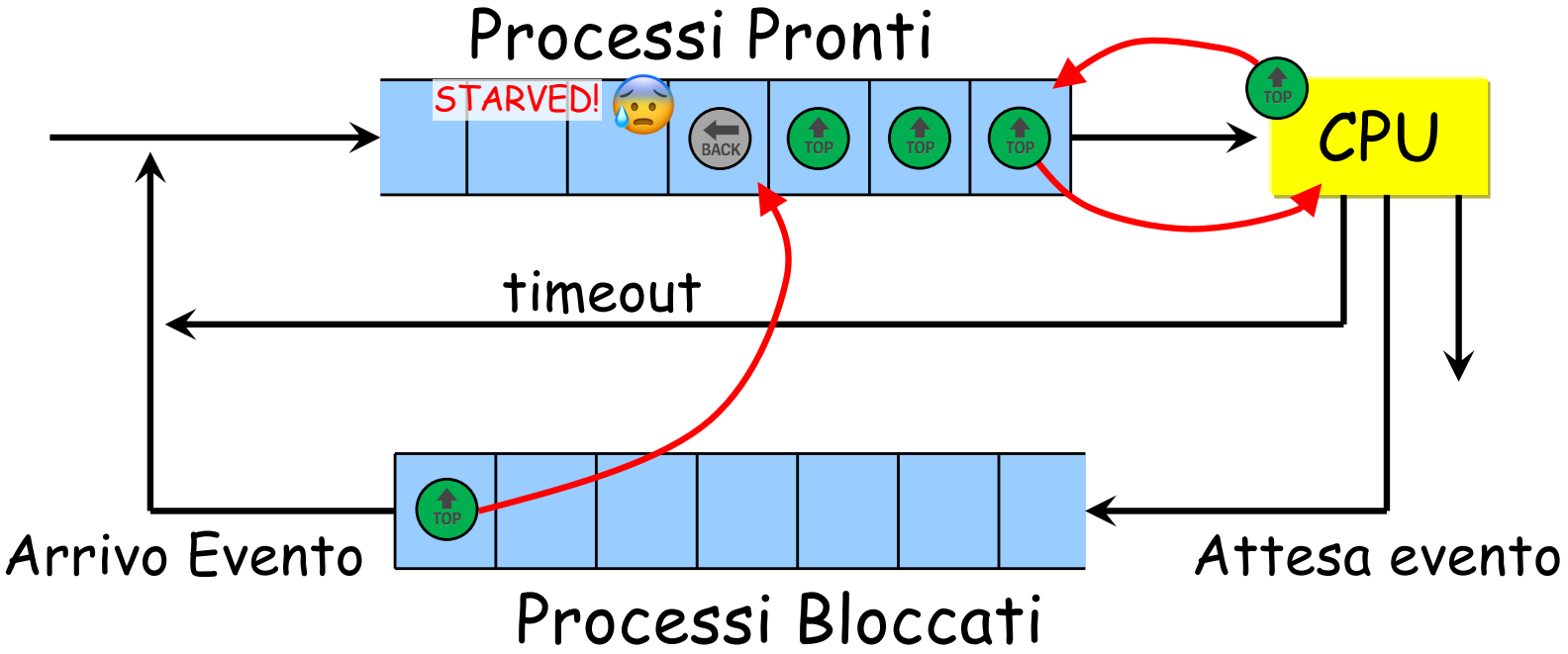
Utilizzo della CPU:
percentuale di tempo in cui la CPU risulta occupata

Fairness:
a meno di indicazioni da parte dell'utente, i processi dovrebbero essere trattati tutti allo stesso modo, e nessuno deve subire una attesa indefinita (starvation)

Starvation

Un processo è in **starvation** se può attendere per un tempo **arbitrariamente lungo** ("attesa indefinita")

Starvation

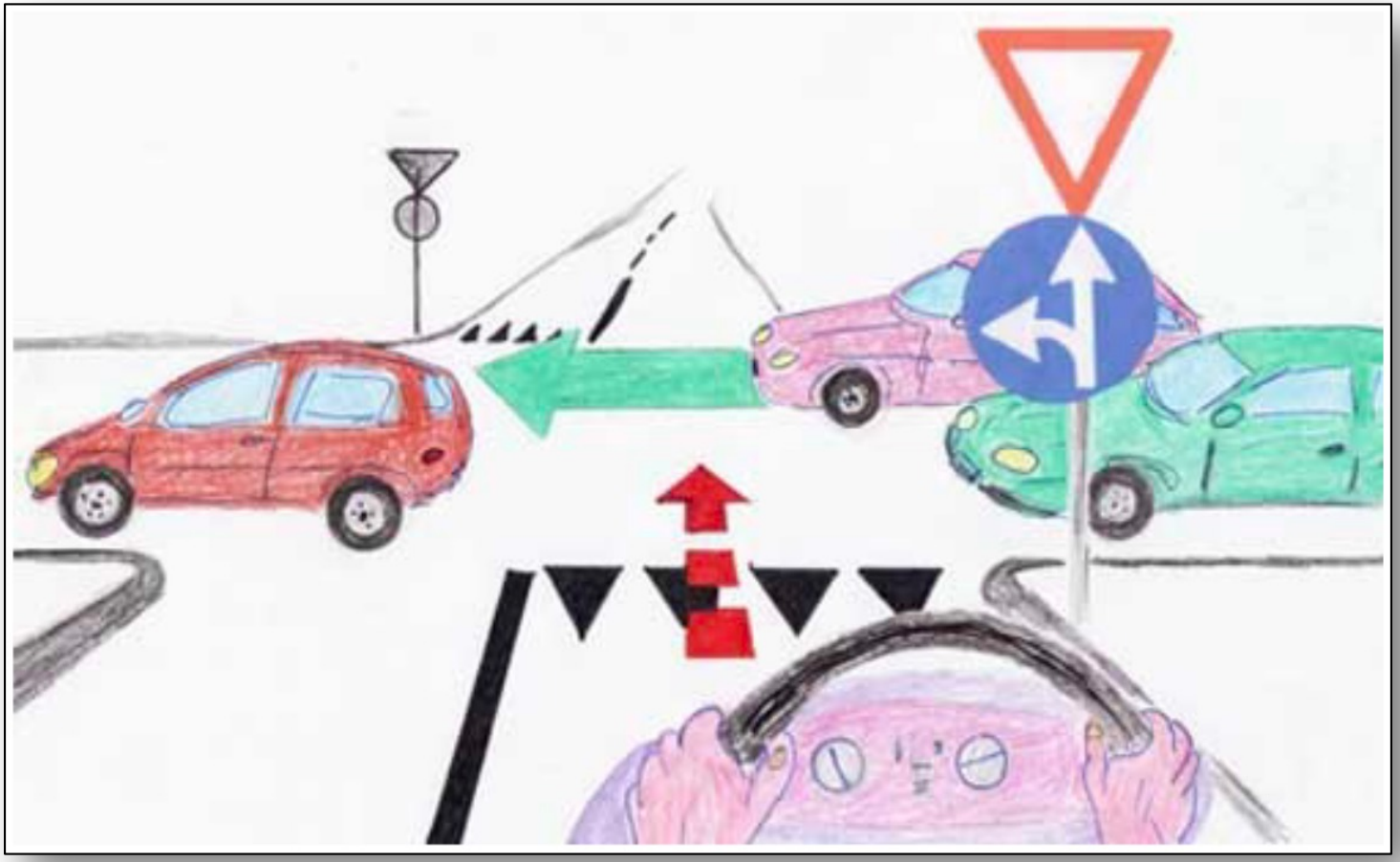


attesa indefinita = no garanzie di esecuzione nel prossimo futuro (da non confondere con "infinita")

Priorità e Starvation

- La starvation si verifica negli algoritmi di scheduling che danno maggiore priorità a specifici processi
- Lo scheduler tende a scegliere sempre i processi a priorità "alta", lasciando gli altri in attesa

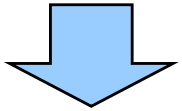
Starvation



da: www.italy-ontheroad.it

Esiste un algoritmo PERFETTO?

- **Naturalmente NO!**
- I vari criteri sono interdipendenti, o persino in contrasto tra loro
  - il tempo di risposta si può ridurre con il prerilascio dei processi o con uso di priorità
  - la percentuale di uso della CPU si riduce con il prerilascio
  - la starvation è un effetto collaterale delle priorità
  - ...



La scelta dovrà essere fatta tenendo conto "*per cosa dovrà essere utilizzato il sistema*"

Algoritmi di scheduling a breve termine

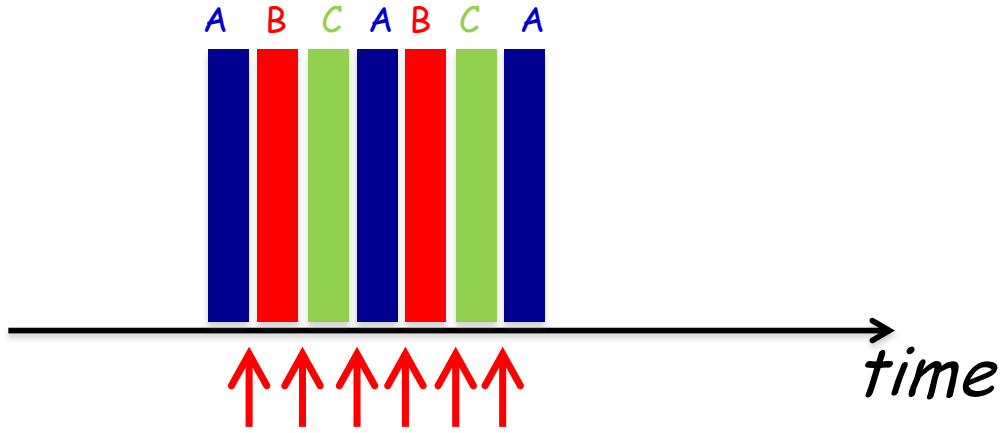
- First-Come First-Served (non-preemptive)
- Round-Robin (preemptive)
- Shortest Process Next (non-preemptive)
- Shortest Remaining Time (preemptive)
- Multilevel Feedback (preemptive)

Preemption

- **Algoritmi non-preemptive**
  - Un processo in esecuzione permane in tale stato finchè non si sospenderà volontariamente

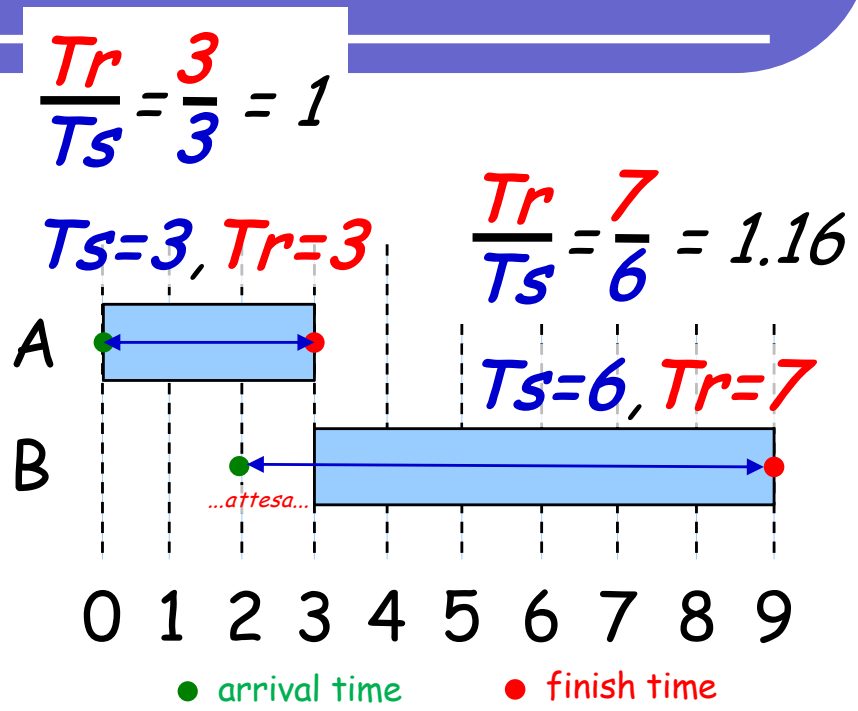
Preemption

- **Algoritmi preemptive**
  - Un processo in esecuzione può essere interrotto anche se non effettua system call
  - Viene forzato nello stato pronto



Un esempio di riferimento

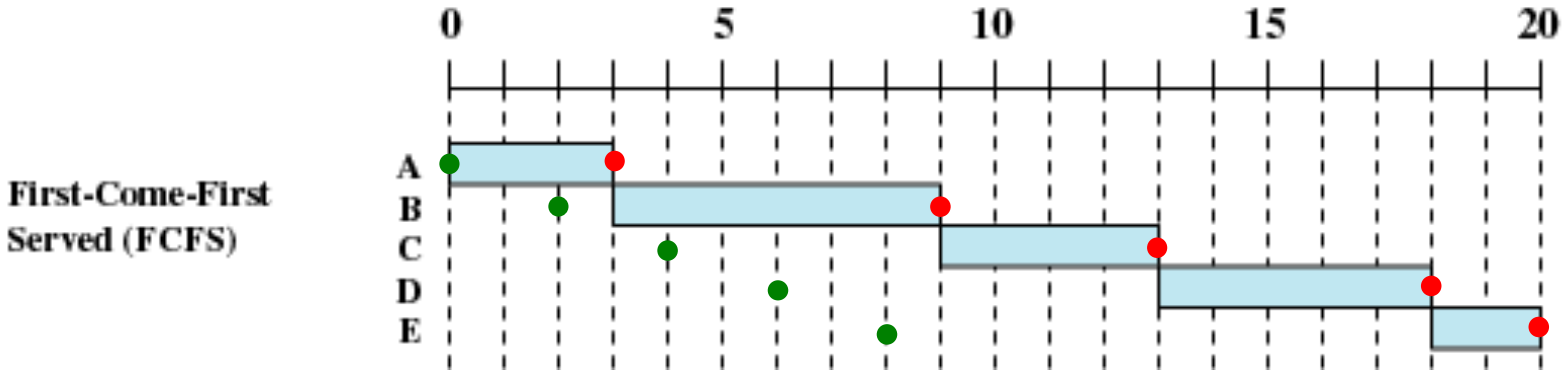
Processo	Arrival time	Service time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2



- Ts = Tempo di Servizio
- Tr = Tempo di Turnaround
- *Tr/Ts* = indicatore di ritardo "relativo"
  (se Ts è alto, l'utente può "tollerare" un ritardo più alto)

First-Come-First-Served (FCFS)

FCFS: Quando un processo termina, viene eseguito il primo processo in **ordine di arrivo**



Turnaround $TrA=3,\ TrB=7,\ TrC=9,\ TrD=12,\ TrE=12$

Tr/Ts $TrA/TsA=1,\ TrB/TsB=1.17,\ TrC/TsC=2.25$

$TrD/TsD=2.40,\ TrE/TsE=6.00$

Il processo *E* spende nel sistema un tempo che è **6 volte** quello di esecuzione!

First-Come-First-Served (FCFS)

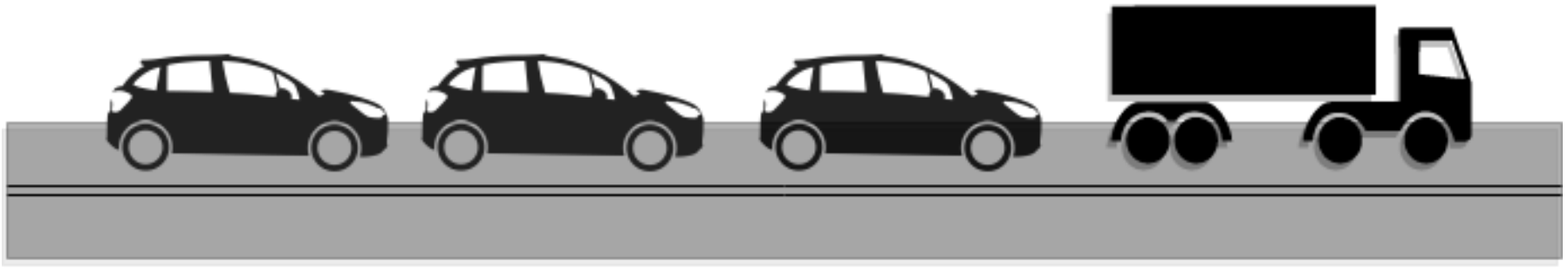
- Non risente di starvation
- Può risentire dell'effetto convoglio

Effetto convoglio:
un processo CPU-bound occupa la CPU per un tempo eccessivo, a scapito dei processi I/O-bound

Effetto convoglio

I/O-bound
(uso breve della CPU)

CPU-bound
(uso prolungato della CPU)

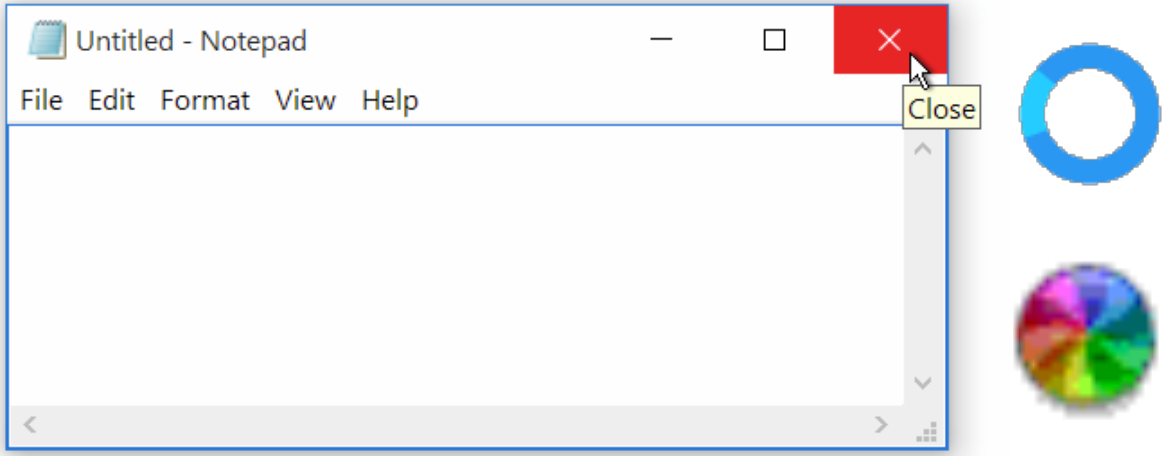


Effetto convoglio

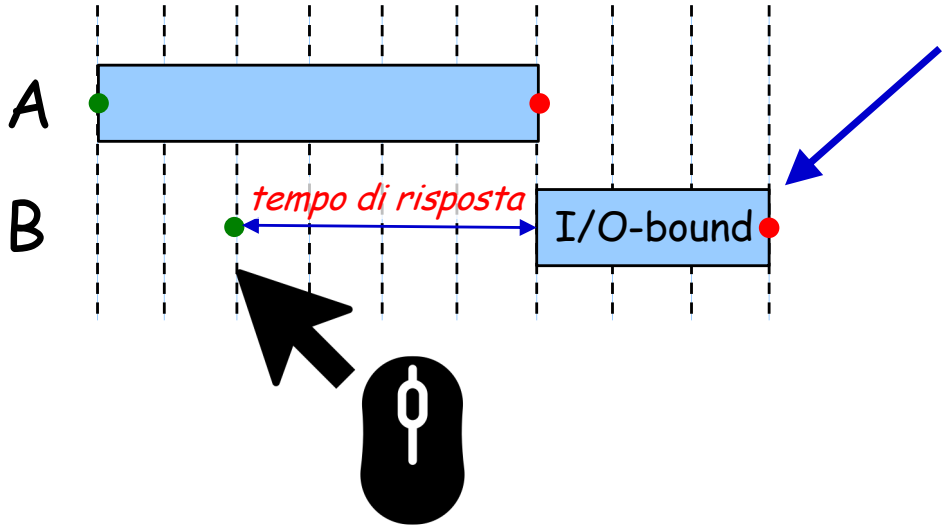
- L'effetto convoglio causa una riduzione dell'utilizzo delle risorse (sia CPU, sia I/O)
- I processi I/O-bound rimangono fermi a lungo, lasciando i dispositivi inutilizzati
- Quando i processi I/O-bound eseguono, la CPU è sotto-utilizzata

Processi I/O-bound

- I/O include le interazioni con l'utente (mouse, tastiera, video, ...)
- Se un processo I/O-bound ha un tempo di risposta lungo, l'utente percepisce un rallentamento!



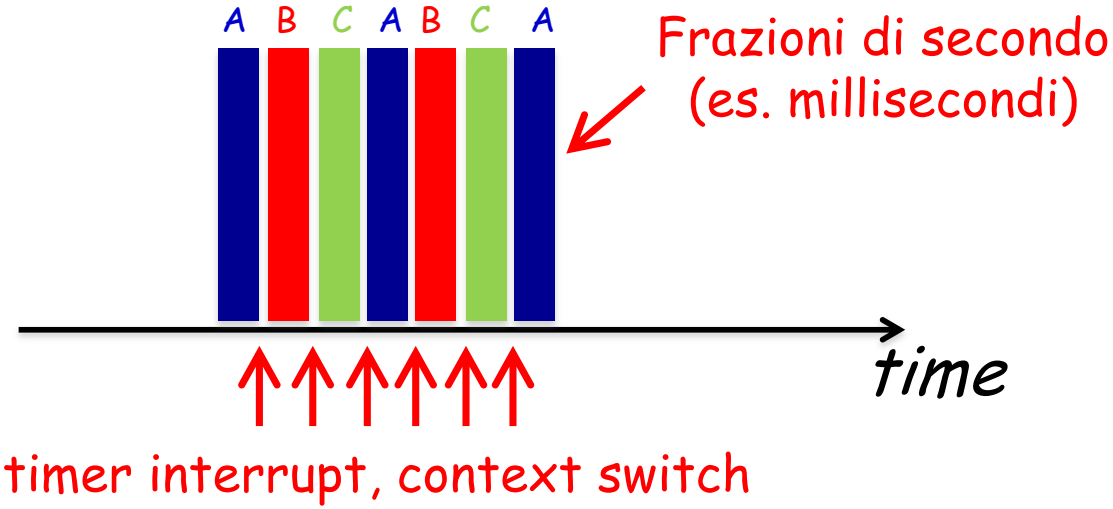
Processi I/O-bound



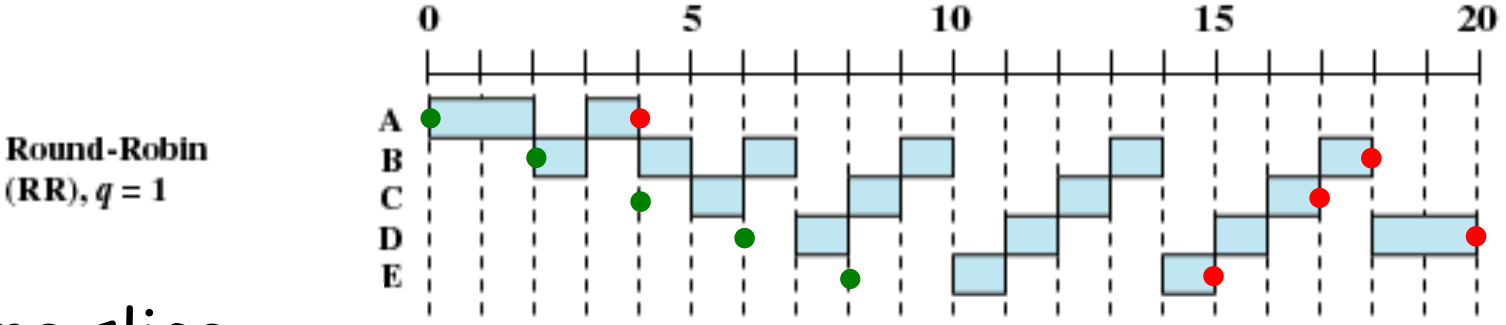
L'utente percepisce un lungo tempo di risposta

Round-Robin

- **Round-Robin**: versione **preemptive** del FCFS, mediante l'impiego di un timer
- Viene assegnato un tempo massimo di esecuzione, detto "time slice" o "quanto di tempo"



Round-Robin



Time slice =
1 unità di tempo

TrA=4, TrB= 16, TrC=13, TrD=14, TrE=7

*TrA/TsA=**1.33**, TrB/TsB=**2.67**, TrC/TsC=**3.25***
*TrD/TsD=**2.80**, TrE/TsE=**3.50***

Con il round-robin il processo *E* **dimezza** il suo ritardo relativo

Round-Robin: scelta del time slice

- Le prestazioni del sistema dipendono dalla scelta della durata del time slice
- Aumentando il time slice, tende a trasformarsi in FCFS
- Diminuendo il time slice, aumenta la **frequenza dei cambi di contesto fra processi**, e quindi l'**overhead** del S.O.

Round-Robin: scelta del time slice

Esempio (dalla prima lezione)

Ipotizzando che

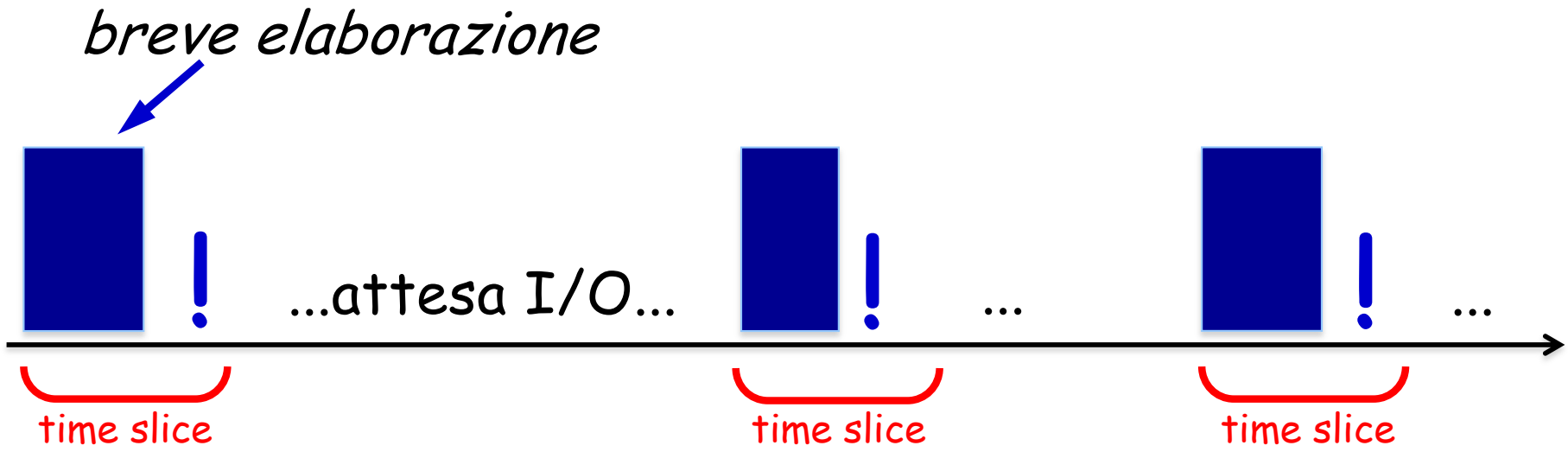
- ogni **quanto di tempo dura 10 ms**
- ogni **context switch dura 1 ms (un caso estremo!)**

viene **"sprecato" circa il 10%** del tempo di CPU per la gestione del SO (invece che fare lavoro utile)

Nelle CPU moderne, il context switch impiega circa **5 microsecondi**...
... con ulteriori effetti collaterali sulle cache

https://learn.microsoft.com/en-us/gaming/gdk/_content/gc/system/overviews/finding-threading-issues/high-context-switches

Round-Robin: scelta del time slice



Idealmente, il time slice dovrebbe essere appena superiore alla durata dei processi I/O-bound

- per dargli tempo di attivare i dispositivi e liberare la CPU
- valori tipici: da 10 a 100 millisec.

Shortest Process

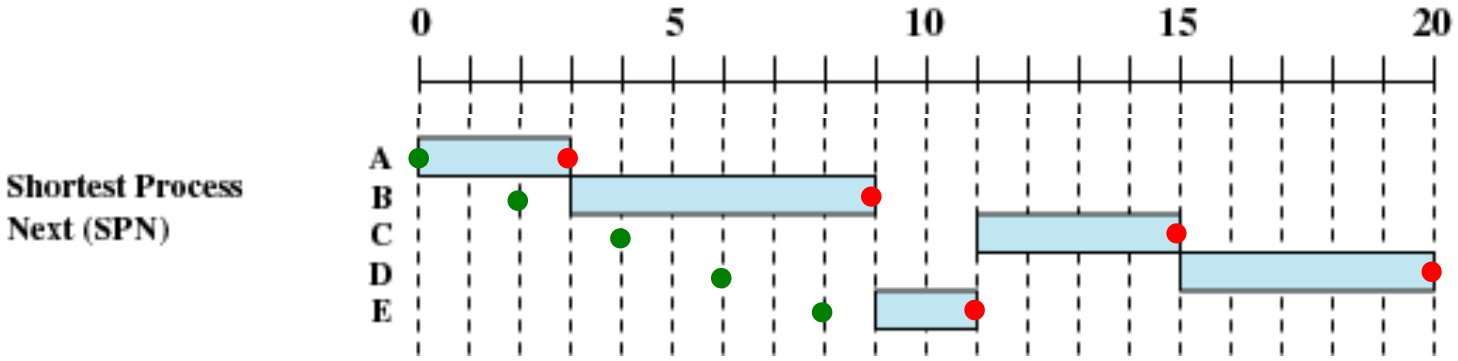
- "Shortest Process": esegue prima il processo con il **minor tempo di esecuzione (stimato)**
- Riduce i tempi di risposta per i processi brevi...
- ...a scapito dei processi lunghi!
- Possibilità di starvation

Shortest Process

- Versione **non-preemptive**:
  Shortest Process Next

- Versione **preemptive**:
  Shortest Remaining Time

Shortest Process Next



TrA=3, TrB= 7, TrC=11, TrD=14, TrE=3

*TrA/TsA=**1**, TrB/TsB=**1.17**, TrC/TsC=**2.75***

*TrD/TsD=**2.80**, TrE/TsE=**1.50***

Possibilità di **starvation** per i processi più lunghi (es., il processo *D*)

SPN: stima del tempo di servizio

- SPN richiede di **conoscere, o almeno stimare**, il tempo di servizio di ciascun processo
- Su indicazione del programmatore
- Stima dalla media dei periodi di esecuzione passati

SPN: stima del tempo di servizio

Esempio: **media aritmetica**

$$S_{n+1} = \frac{1}{n}\sum_{i=1}^{n} T_i = \frac{1}{n} T_n + \frac{n-1}{n} S_n$$

Valore stimato per esecuzione (n+1)-esima

Esecuzione i-esima

Valore stimato per esecuzione n-esima

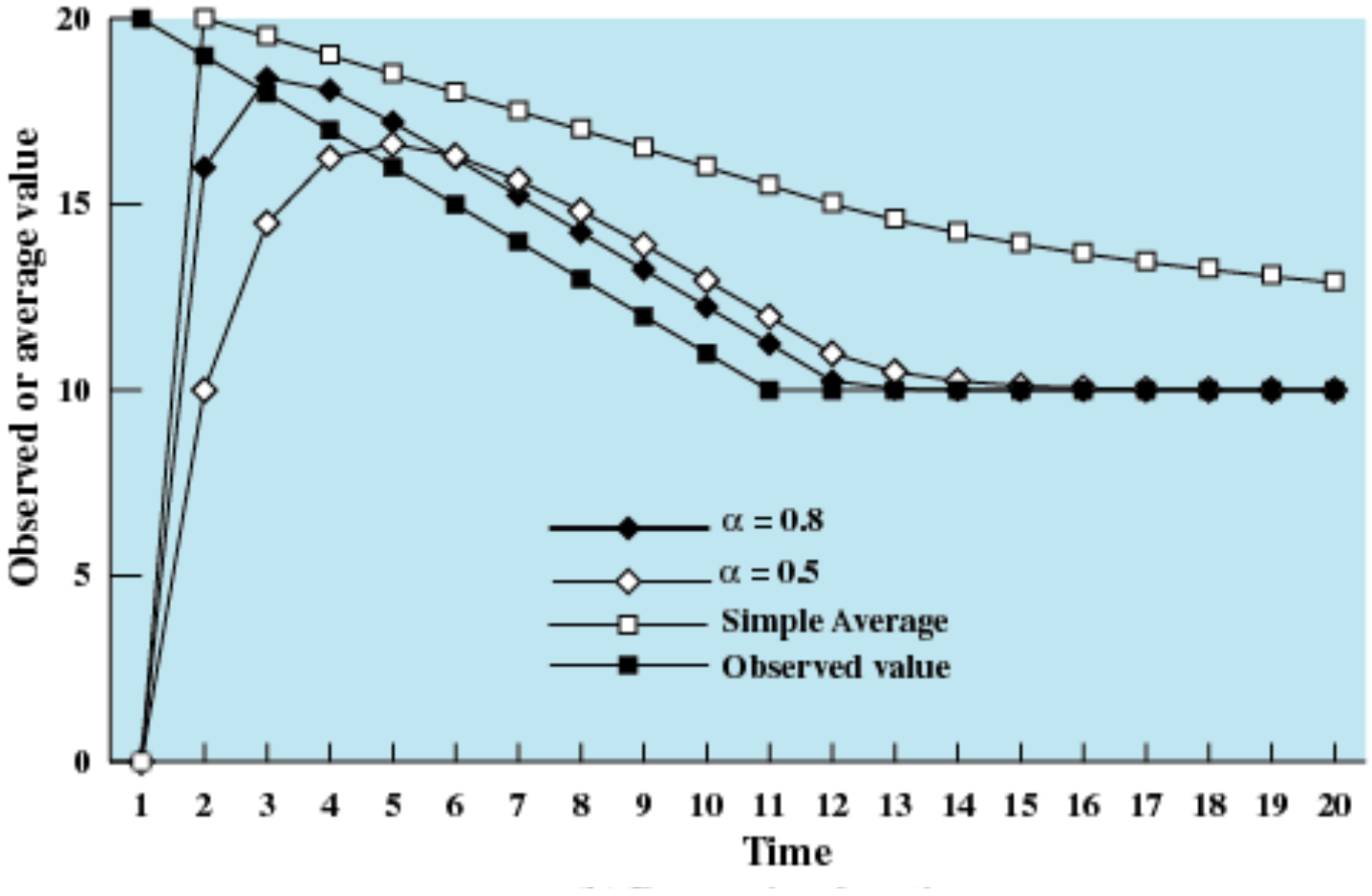
SPN: stima del tempo di servizio

- La media aritmetica da ugual peso ad ogni T_i
- Un approccio comune è dare un peso maggiore ai T_i più recenti, in quanto più rappresentativi del comportamento futuro del processo

Esempio: **media esponenziale**

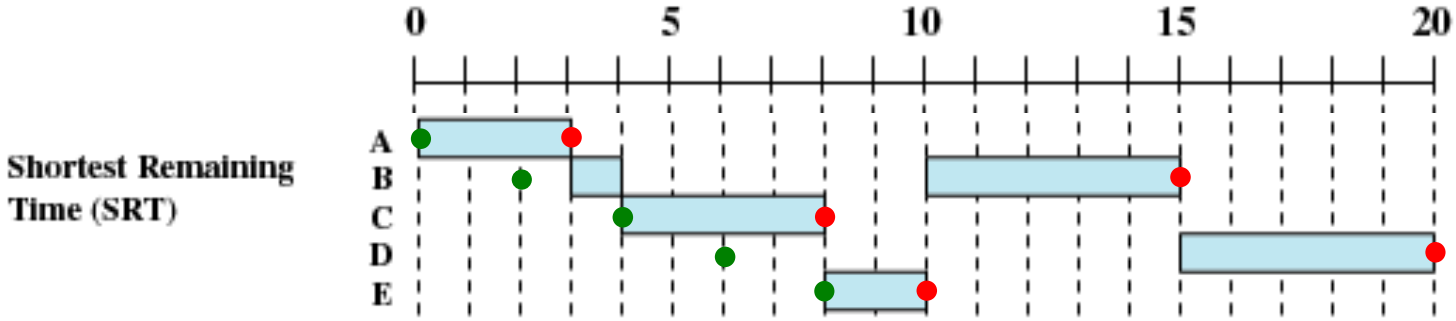
$$S_{n+1} = \alpha T_n + (1-\alpha) S_n \qquad 0 < \alpha < 1$$

SPN: confronto tra media semplice e media esponenziale



Shortest Remaining Time

- SRT: versione **preemptive** dello Shortest Process Next (SPN)
- La prelazione può **avvenire subito** quando entra un nuovo processo
- In pratica, i processi brevi non aspettano mai



*TrA/TsA=**1**, TrB/TsB=**2.17**, TrC/TsC=**1***

*TrD/TsD=**2.80**, TrE/TsE=**1***

Rispetto al caso SPN, il processo **C** prelaziona il processo **B**

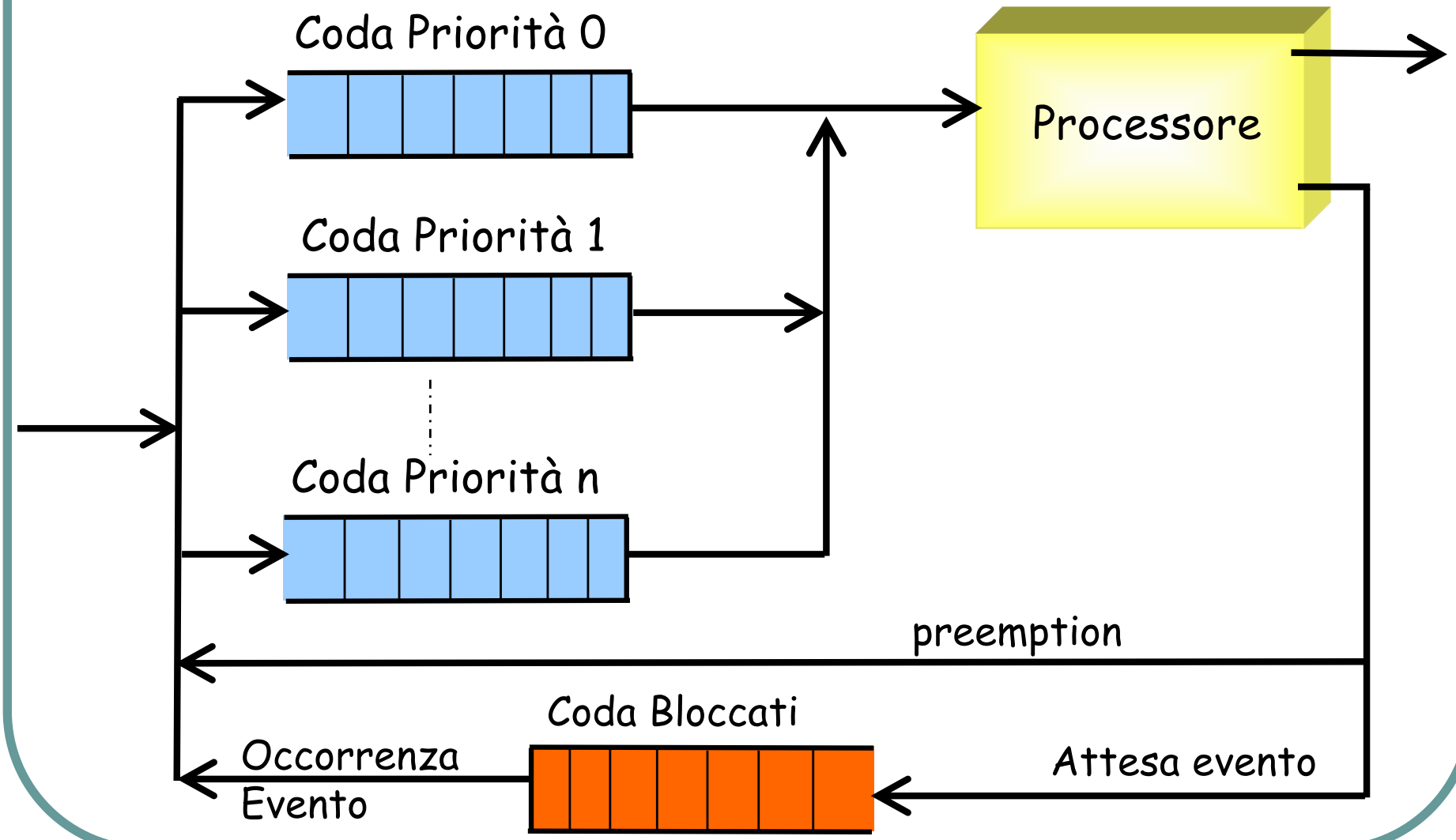
Scheduler a Priorità

- Gli algoritmi SPN e SRT sono di **difficile utilizzo**, perché è difficile fare stime precise
- Lo **scheduler a priorità** (**multilevel**) semplifica, raggruppando i processi in livelli di priorità
- I processi a stesso livello sono gestiti in modo round-robin

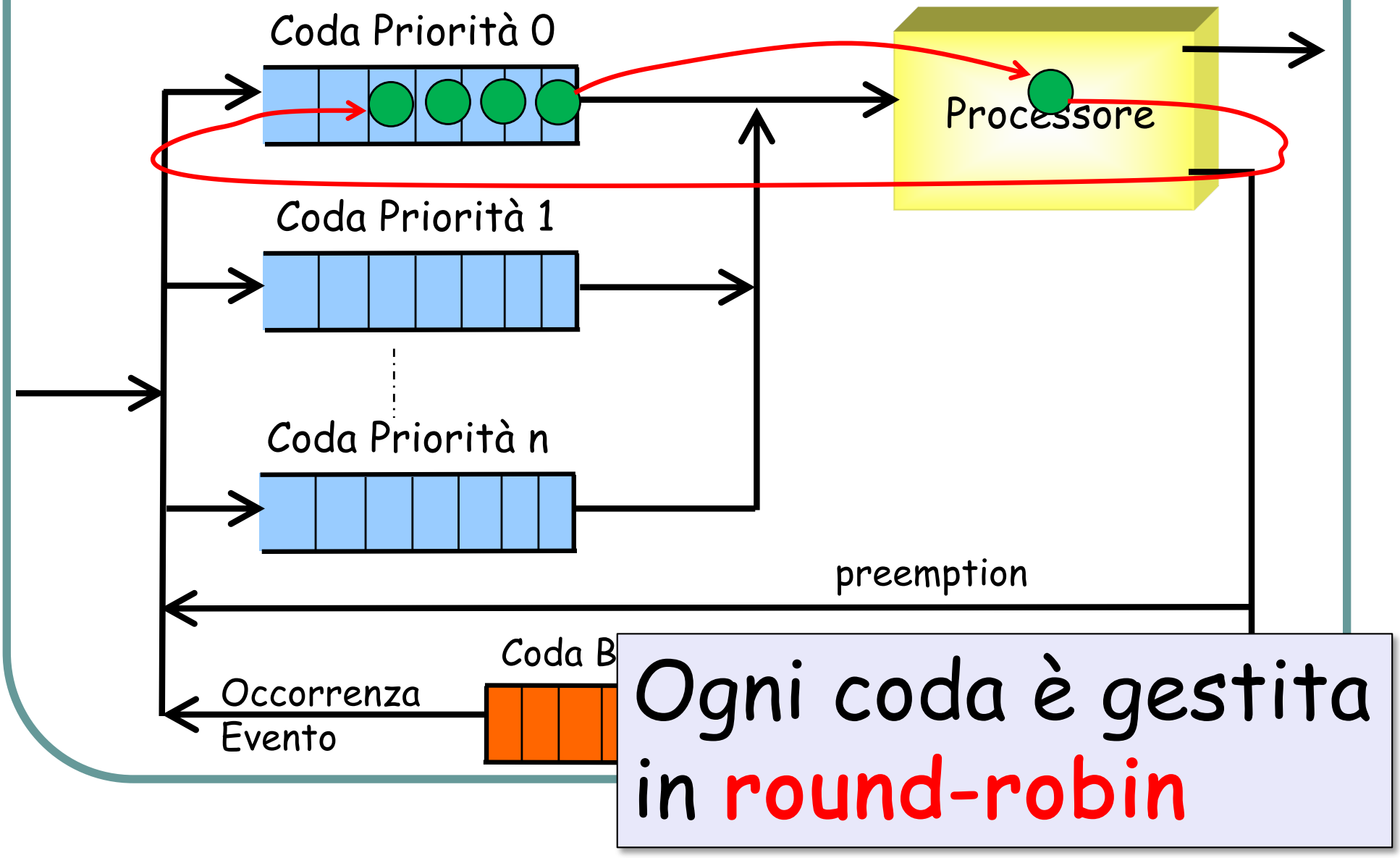
Scheduler a Priorità

- Ogni gruppo di processi è gestito con una **coda separata**
  - La 1ª coda ha precedenza sulla 2ª
  - La 2ª coda ha precedenza sulla 3ª
  - ...

Scheduler a Priorità

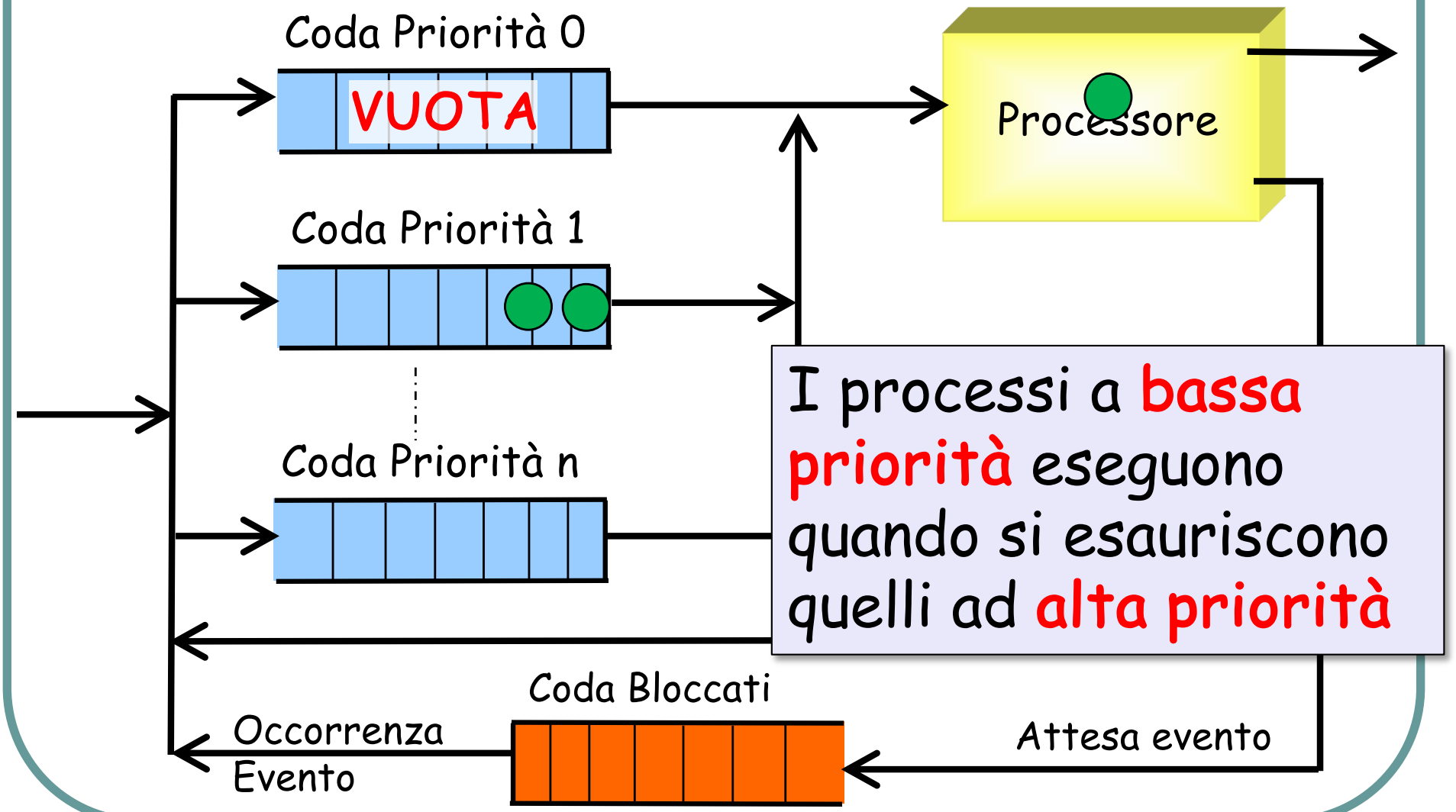


Scheduler a Priorità



Ogni coda è gestita in **round-robin**

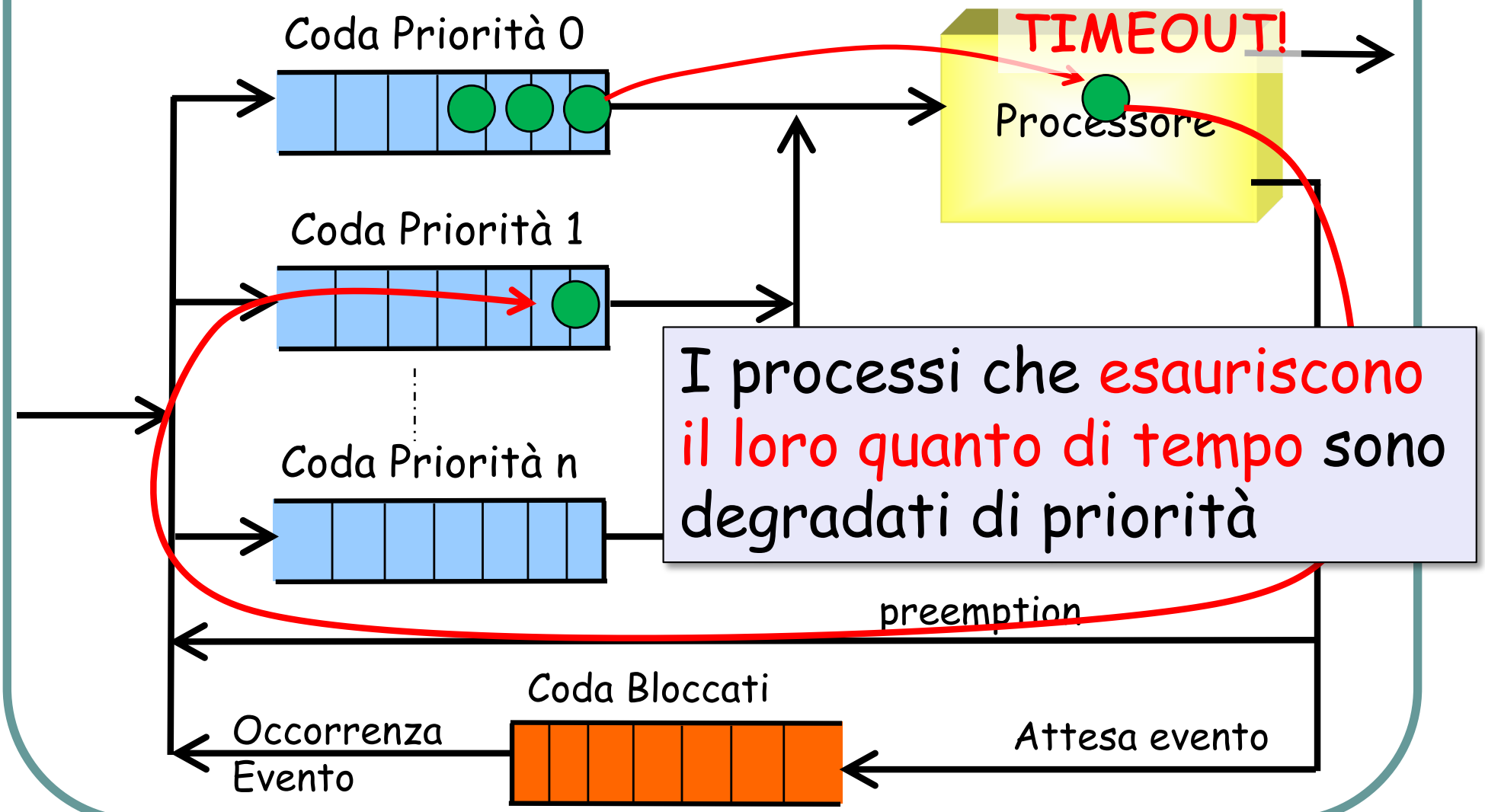
Scheduler a Priorità



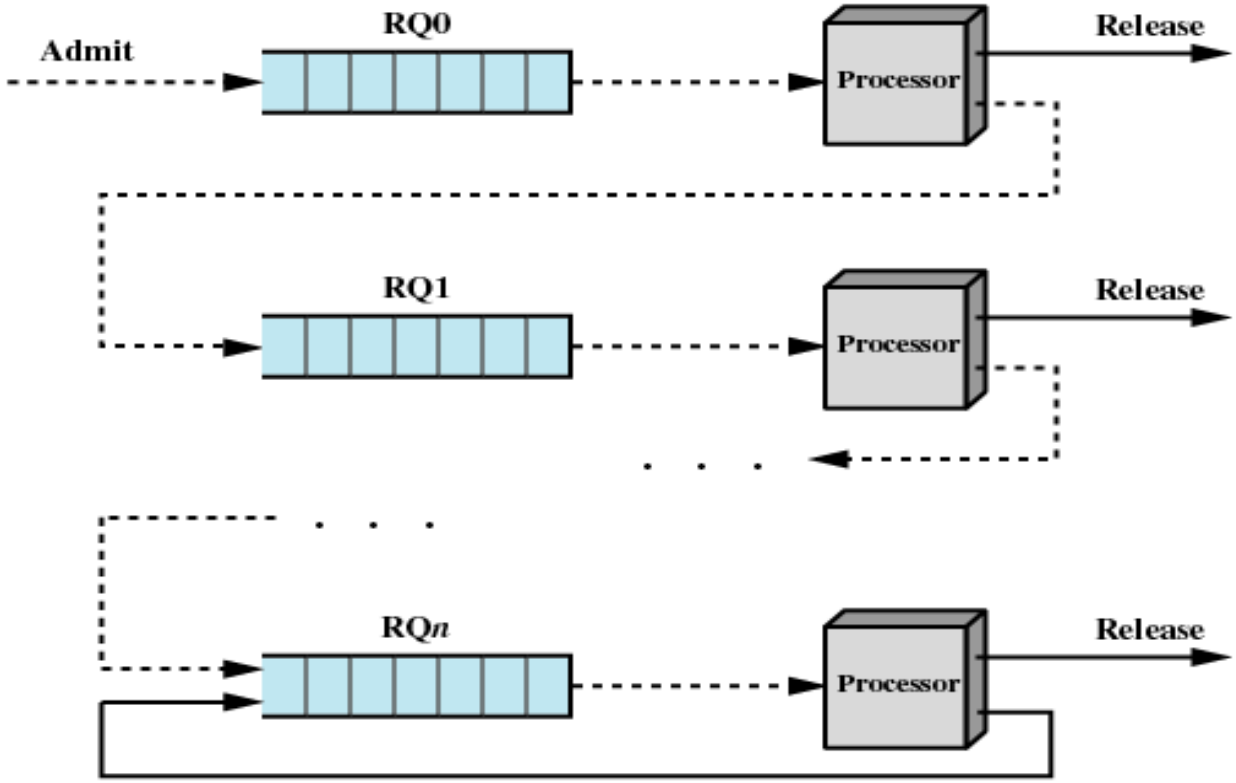
Multilevel Feedback

- Per non penalizzare i processi a bassa priorità, si utilizza un meccanismo di "**feedback**"
- Si assegna ai processi una **priorità dinamica**, penalizzando i processi CPU-bound a favore di quelli I/O-bound

Scheduler a Priorità, con feedback



Multilevel Feedback



Tipicamente, ogni coda e servita con un algoritmo Round-Robin. L'ultima coda è caratterizzata da uno scheduler FCFS (es., processi di background)

Multilevel Feedback

- Favorisce i processi più **corti**
- Occorre minimizzare la starvation per i processi più lunghi

- I processi a minor priorità sono compensati con un quanto di tempo più lungo

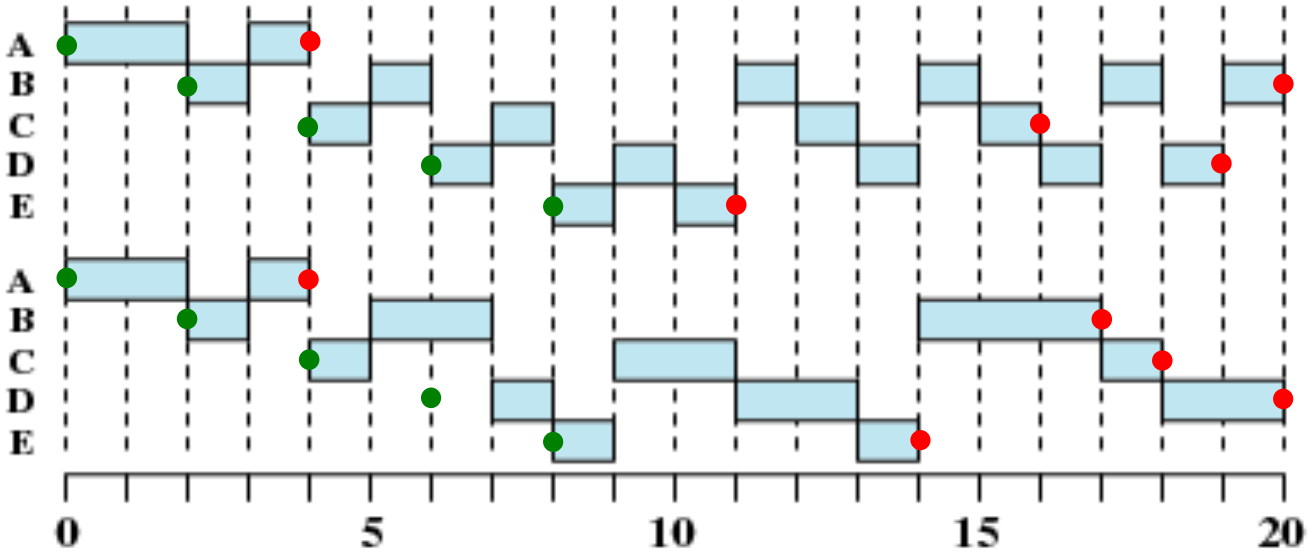
coda i-esima -> 2^i unità di tempo

- Dopo un tempo massimo, si fa ritornare un processo nella coda inferiore nella coda ad alta priorità

Multilevel Feedback

Feedback
$q = 1$

Feedback
$q = 2^i$



Processo	Arrival time	Service time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

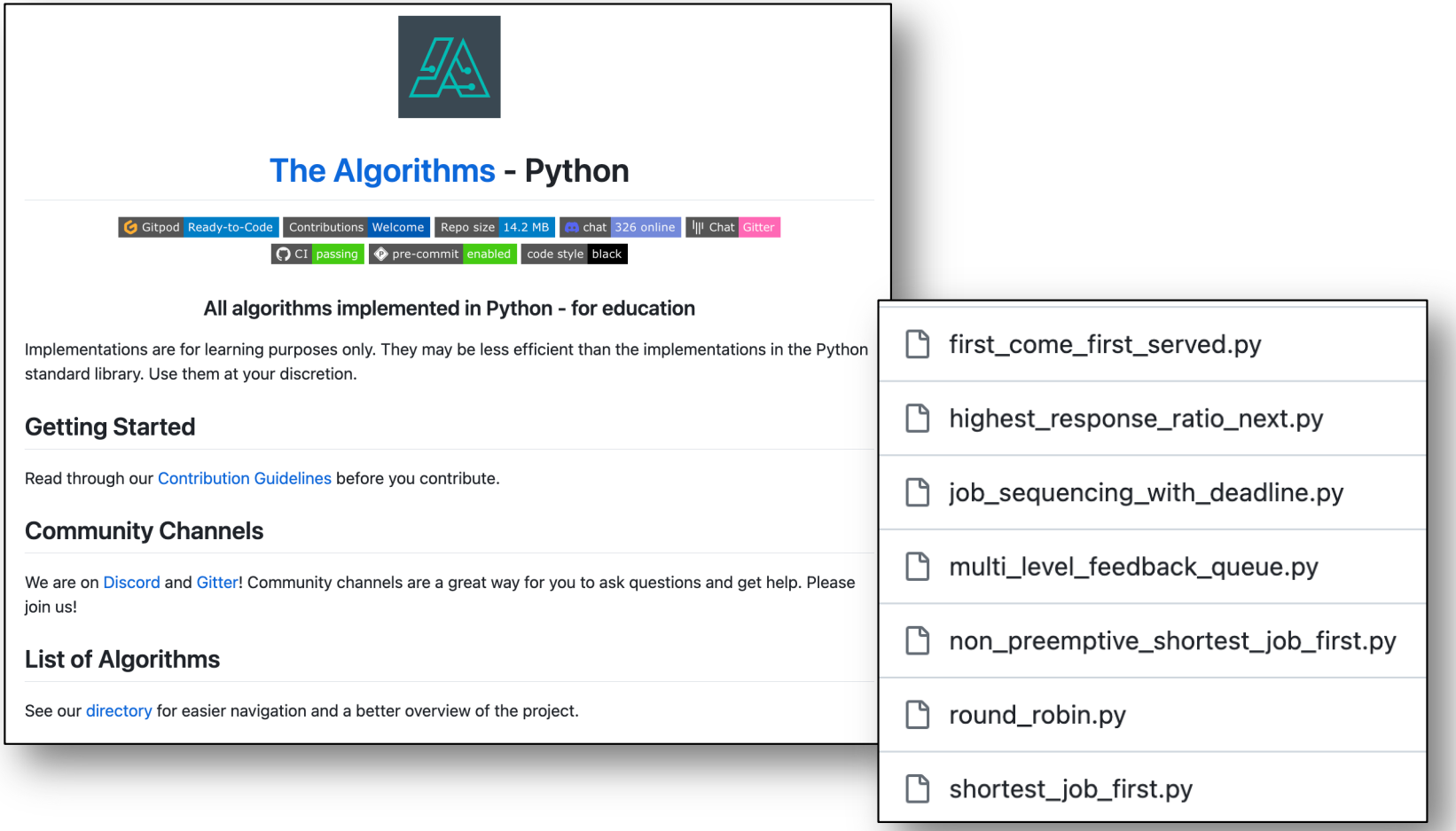
	Funzione di selezione	Modalità di decisione	Throughput	Tempo di risposta	Overhead	Impatto sui processi	Attesa indefinita
FCFS	Max[w]	Non interrrompente	Non enfatizzato	Potrebbe essere alto, nel caso in cui il tempo di esecuzione abbia una varianza elevata	Minimo	Penalizza i processi brevi ed I/O bound	No
Round Robin	Costante	Interrrompente (quanto di tempo)	Può essere basso se il quanto di tempo è eccessivamente piccolo	Tempi di risposta buoni per processi brevi	Minimo	Politica fair	No
SPN	Min [s]	Non interrrompente	Elevato	Tempi di risposta buoni per processi brevi	Può essere elevato	Penalizza i processi lunghi	Possibile
SRT	Min [s-e]	Interrrompente (istante di arrivo)	Elevato	Buono	Può essere elevato	Penalizza i processi lunghi	Possibile
Feedback	Con retroazione	Interrrompente (quanto di tempo)	Non enfatizzato	Non enfatizzato	Può essere elevato	Potrebbe favorire i processi I/O bound	Possibile

Caratteristiche delle differenti politiche di schedulazione

w= tempo di attesa; e= tempo di esecuzione; s= tempo di servizio, incluso e

	Processo **Tempo di Arrivo** **Tempo di Servizio (Ts)**	A 0 3	B 2 6	C 4 4	D 6 5	E 8 2	Media
FCFS	**T. di Completamento** **T.di Turnaround (Tr)** **Tr/Ts**	3 3 1	9 7 1.17	13 9 2.25	18 12 2.40	20 12 6	8.60 2.56
RR q=1	**T. di Completamento** **T.di Turnaround (Tr)** **Tr/Ts**	4 4 1.33	18 16 2.67	17 13 3.25	20 14 2.80	15 7 3.50	10.80 2.71
RR q=4	**T. di Completamento** **T.di Turnaround (Tr)** **Tr/Ts**	3 3 1	17 15 2.5	11 7 1.75	20 14 2.80	19 11 5.50	10 2.71
SPN	**T. di Completamento** **T. di Turnaround (Tr)** **Tr/Ts**	3 3 1	9 7 1.17	15 11 2.75	20 14 2.80	11 3 1.50	7.60 1.84
SRT	**T. di Completamento** **T. di Turnaround (Tr)** **Tr/Ts**	3 3 1	15 13 2.17	8 4 1	20 14 2.80	10 2 1	7.20 1.59
FB q=1	**T. di Completamento** **T. di Turnaround (Tr)** **Tr/Ts**	4 4 1.33	20 18 3	16 12 3	19 13 2.60	11 3 1.5	10 2.29
FB $q=2^i$	**T. di Completamento** **T. di Turnaround (Tr)** **Tr/Ts**	4 4 1.33	17 15 2.50	18 14 3.50	20 14 2.80	14 6 3	10.60 2.63

Per approfondire



The Algorithms - Python

Gitpod Ready-to-Code | Contributions Welcome | Repo size 14.2 MB | chat 326 online | Chat Gitter

CI passing | pre-commit enabled | code style black

All algorithms implemented in Python - for education

Implementations are for learning purposes only. They may be less efficient than the implementations in the Python standard library. Use them at your discretion.

Getting Started

Read through our Contribution Guidelines before you contribute.

Community Channels

We are on Discord and Gitter! Community channels are a great way for you to ask questions and get help. Please join us!

List of Algorithms

See our directory for easier navigation and a better overview of the project.

- first_come_first_served.py
- highest_response_ratio_next.py
- job_sequencing_with_deadline.py
- multi_level_feedback_queue.py
- non_preemptive_shortest_job_first.py
- round_robin.py
- shortest_job_first.py

https://github.com/TheAlgorithms/Python/tree/master/scheduling

Quiz

1. Quali di questi algoritmi possono causare starvation? (selezionare più di una)

- [ ] FCFS
- [ ] Round Robin
- [ ] SPN
- [ ] Multilevel feedback

2. Quanto dovrebbe essere lungo il time slice?

- ( ) 10 nanosecondi
- ( ) 10 microsecondi
- ( ) 10 millisecondi
- ( ) 1 secondo

https://forms.office.com/r/UT5DTWqxLG