CS & IT ENGINEERING

Operating Systems

CPU Scheduling

Lecture No. 3

By- Dr. Khaleel Khan Sir

TOPICS TO BE COVERED

- Round Robin
- Performance of Round Robin
- Multi-Level Queue Scheduling

Performance of Round-Robin :

Time Quantum

Very Small
cpu efficiency ~ 0

Small
→ More context switching overhead
→ cpu efficiency drops
→ System becomes more Interactive

large
→ Less context switch overhead
→ System becomes less (Poor) Interactive

very large:
→ It degenerates to work like FCFS

The value of TQ should neither be too large, & nor too small;

P.No	A.T	B.T
1	0	4
2	0	3
3	0	2
4	1	1

(i) TQ = 10

cpu

P1	P2	P3	P4
0–4	4–7	7–9	9–10

<FCFS>

(ii) TQ = 0.1 ; δ = 2

CPU

δ	P1	δ	P2	δ	P3	δ	P4
0–2	2–2.1	2.1–4.1	4.1–4.2	4.2–6.2	6.2–6.3	6.3–8.3	8.3–8.4

$$\text{cpu efficiency} = \frac{0.4}{8.4} = \frac{1}{21} \sim 5\%$$

cpu ovh'd = 95%

Multi-Level Queue Scheduling :

1000

R.Q

- - - - - - -

Processes

CPU

Single R.Q System

(i) Searching Time

Filteration "

(ii) Apply one Scheduling Technique

Feedback Multi-Level Q

(2-levels of Scheduling)

RQ1 — OS processes

RQ2 — Interactive

RQ3 — Foreground

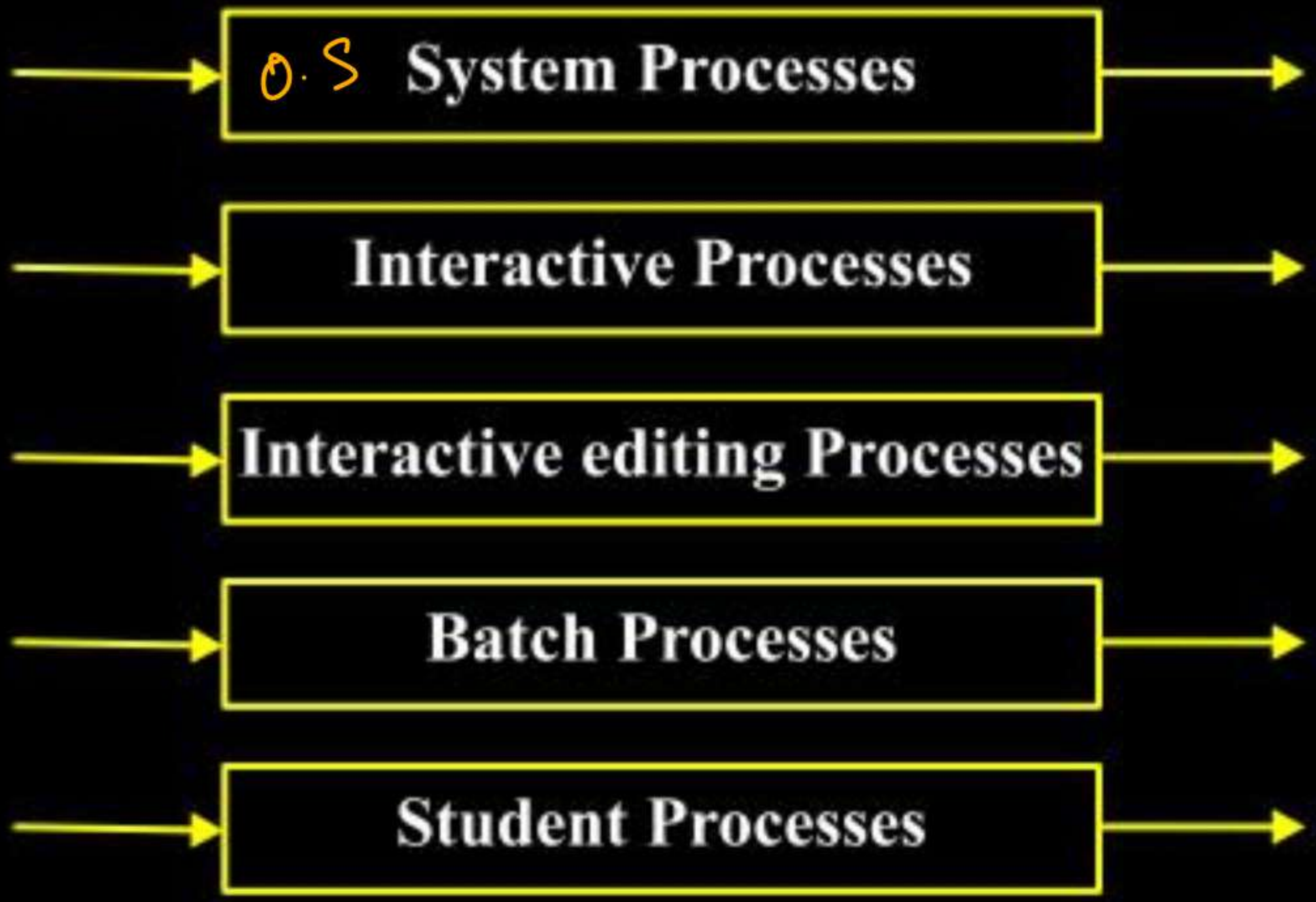
RQ4 — Background

CPU

(The Processes in the Lower level Q's will Starve)

Highest Priority

Simple Multi-Level 'Q'

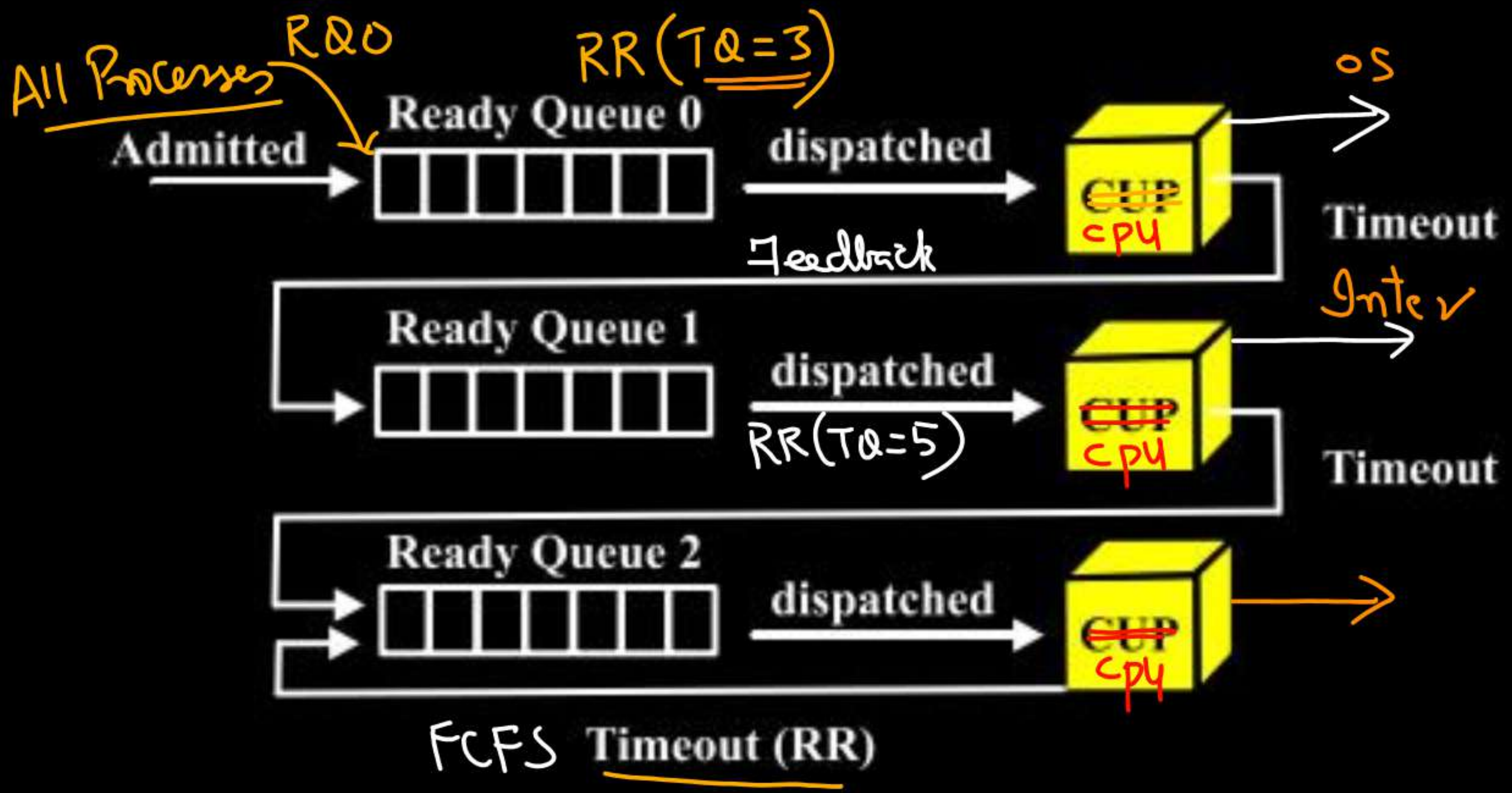


Highest Priority

Multilevel feedback Queue Scheduling

- Another way to put a preference on short-lived processes
  - Penalize processes that have been running longer
- Preemptive

Last Q in the System generally uses FCFS to ensure completion



Processes in Rq1 are scheduled only when no processes exist in Rq0

Processes in RQ2 are scheduled only when no processes exist in Rq1

Longer processes gradually drift downward

Case Study (Interviews)

Implementation of CPU Scheduling in

a) LINUX : O(n) ; O(1) : F.S

b) UNIX : R.R based

c) WINDOWS : Priority +

d) MAC :

Q. (*)

Consider a System with 'n' Processes arriving at time 0^+ with substantially large Burst Times. The CPU scheduling overhead is 's' seconds, Time Quantum is 'q' seconds. Using Round Robin scheduling, what must be the value of Time Quantum 'q' such that each Process is guaranteed to get its turn at the CPU exactly after 't' seconds in its subsequent run-on CPU.

$n = 5$;

$S = 's'$

$TQ = q$

RQ: ~~P_1~~ P_2 P_3 P_4 P_5 P_1

't'

CPU	s	P_1	s	P_2	s	P_3	s	P_4	s	P_5	s	P_1
	0	s	q	s	q	s	q	s	q	s	q	s

→ Time in C.S = ns

→ Time in Comput by $(n-1)$ processes $= (t - ns)$

$$(t - ns) = (n-1) \cdot q$$

$$\boxed{q = \frac{t - ns}{n-1}}$$

$\rightarrow \quad q = \left(\frac{t - ns}{n-1}\right)$: Each Process gets its turn exactly after 't'

$\rightarrow \quad q \le \left(\frac{t - ns}{n-1}\right)$: Each Process may get a chance atleast once within 't';

$\rightarrow \quad q \ge \left(\frac{t - ns}{n-1}\right)$: Each Process gets its chance atleast every 't' secs;

Q. Which of the following statements (s) is/are correct in the context of CPU Scheduling?

MSQ

(A) The goal is to only maximize CPU utilization and minimize throughput x

(B) Turnaround time includes waiting time

(C) Implementing preemptive scheduling needs hardware support (Timer)

(D) Round-robin policy can be used even when the CPU time required by each of the processes is not known Apriori~~ty~~. (Before hand)

Q. Consider the following set of Processes, assumed to have arrived at time 0 . Consider the CPU scheduling algorithms Shortest Job First (SJF) and Round Robin (RR). For RR assume that the processes are scheduled in the order P_1, P_2, P_3, P_4.

Processes	P_1	P_2	P_3	P_4
Burst time (in ms)	8	7	2	4

If the time quantum for RR is 4 ms, then the absolute value of the difference between the average turnaround times (in ms) of SJF and RR (round off to 2 decimal places) is

Q.

Consider a System using Round Robin Scheduling with 10 Processes all arriving at the time 0. Each Process is associated with 20 identical Request. Each Process request consumes 20 ms of CPU time after which it spends 10 ms of time on I/O, thereafter, initiates subsequent Request. Assuming Scheduling Overhead of 2 ms and Time Quantum of 20 ms, Calculate

i. Response time of the 1st request of the 1st Process : 22 ms

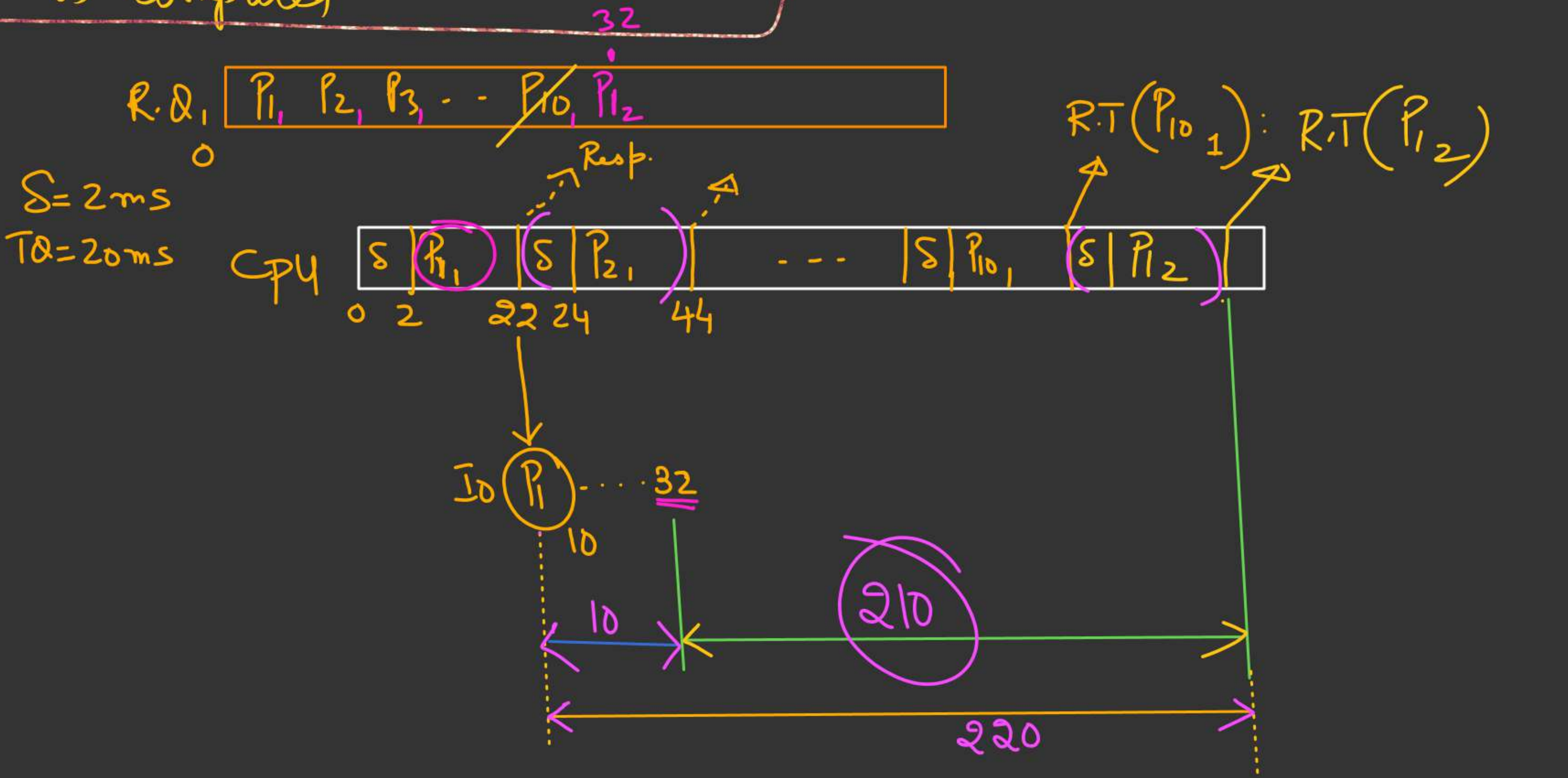
ii. Response time of the 1st request of the last Process : 10 (2+20) = 220 ms

iii. Response time of the subsequent request of any Process.

TQ = 20 ms

TQ = 10 ms (i) (ii)

→ The Response Time of the Request of the Process is the time @ which the request is submitted to the time at which its Computation (CPU) is Completed.



R.Q

P_1	P_2	- -	P_{10}	P_1

CPU

2	10	2	10					
S	P_{1_1}	S	P_{2_1}	- - - -	S	P_{10_1}	S	P_{1_1}
0	2	12	14	24				

$$\left[(2+10)\cdot 10 + (2+10)\right] = 132\,ms \checkmark$$

$$\left[(2+10)10 + (2+10)\cdot 10\right] = 240\,ms \checkmark$$

Q. Consider four Processes P, Q, R, and S scheduled on a CPU as per Round Robin Algorithm with a Time Quantum of 4 units. The Processes arrive in the order P, Q, R, S, all at time t = 0. There is exactly one context switch from S to Q, exactly one context switch from R to Q, and exactly two context switches from Q to R. There is no context switch from S to P. Switching to a ready process after the termination of another process is also considered a context switch. Which one of the following is NOT possible as CPU B.T of Processes?

PYQ

RQ: $\boxed{\cancel{P}\ Q\ R\ S\ Q\ R}$ $TQ = 4$

(A) P = 4, Q = 10, R = 6, S = 2

(B) P = 2, Q = 9, R = 5, S = 1

(C) P = 4, Q = 12, R = 5, S = 4

(D) $\boxed{P = 3, Q = 7, R = 7, S = 3}$ ✓

CPU	P	Q	R	S	Q	R	Q
	0	4					

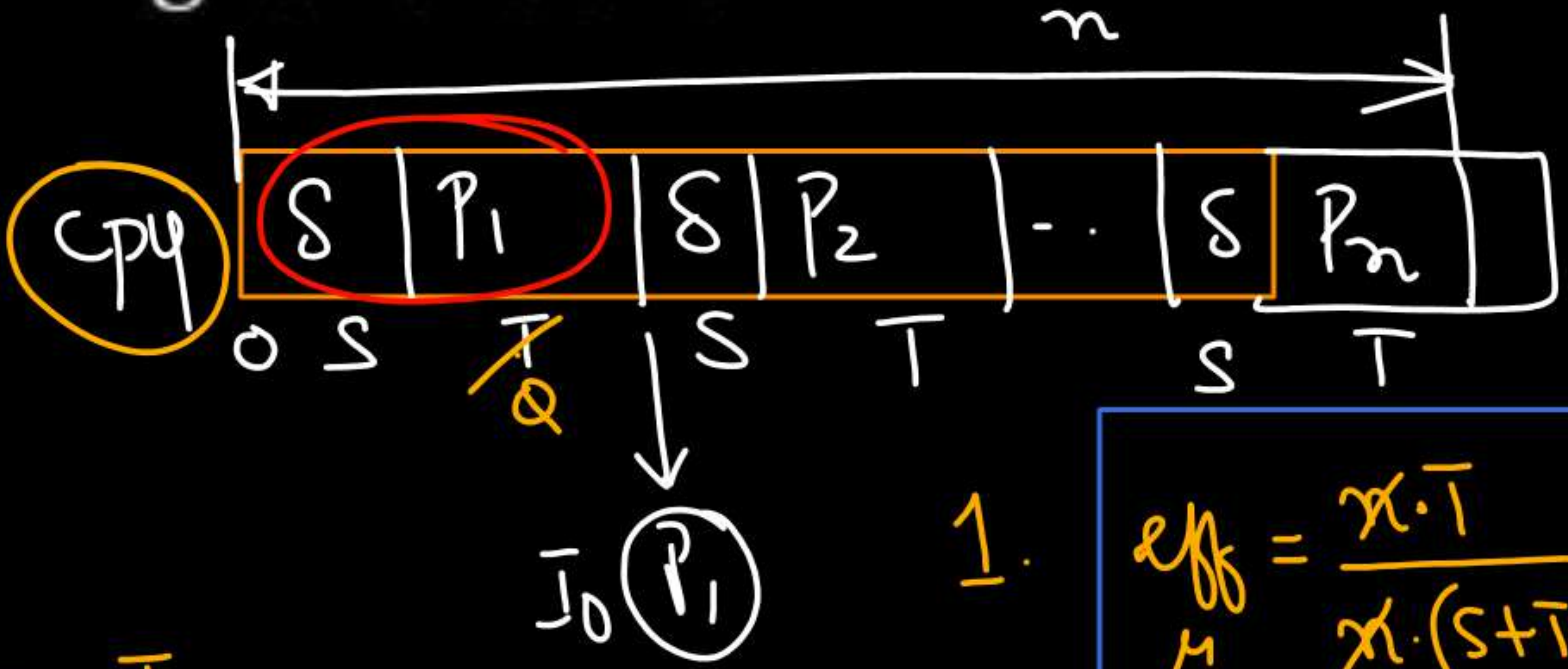
$S \to Q : 1$ ✓
$R \to Q : 1$
$Q \to R : \cancel{2}\ 1$ ✓
$S \to P : 0$ ✓

(ii) $BT(Q) > 8$
(i) $BT(P) \leq 4$

Q. Consider a System using RR Scheduling with TQ of 'Q' seconds & CPU Scheduling overhead is 'S' seconds. Each Process on an average run for 'T' seconds before blocking on I/O. Give a formula for CPU efficiency for each of the following conditions.

Text Book

1. $Q = \infty$
2. $Q > T$
3. $S < Q < T$
4. $Q = S$
5. $Q \approx 0$



1. $\boxed{\text{eff} = \mu = \dfrac{\cancel{n}\cdot T}{\cancel{n}\cdot(S+T)} = \dfrac{T}{S+T}}$

2. $\mu = \dfrac{T}{S+T}$

3. $\mu = \dfrac{Q}{S+Q}$

4. $\mu = \dfrac{1}{2}$

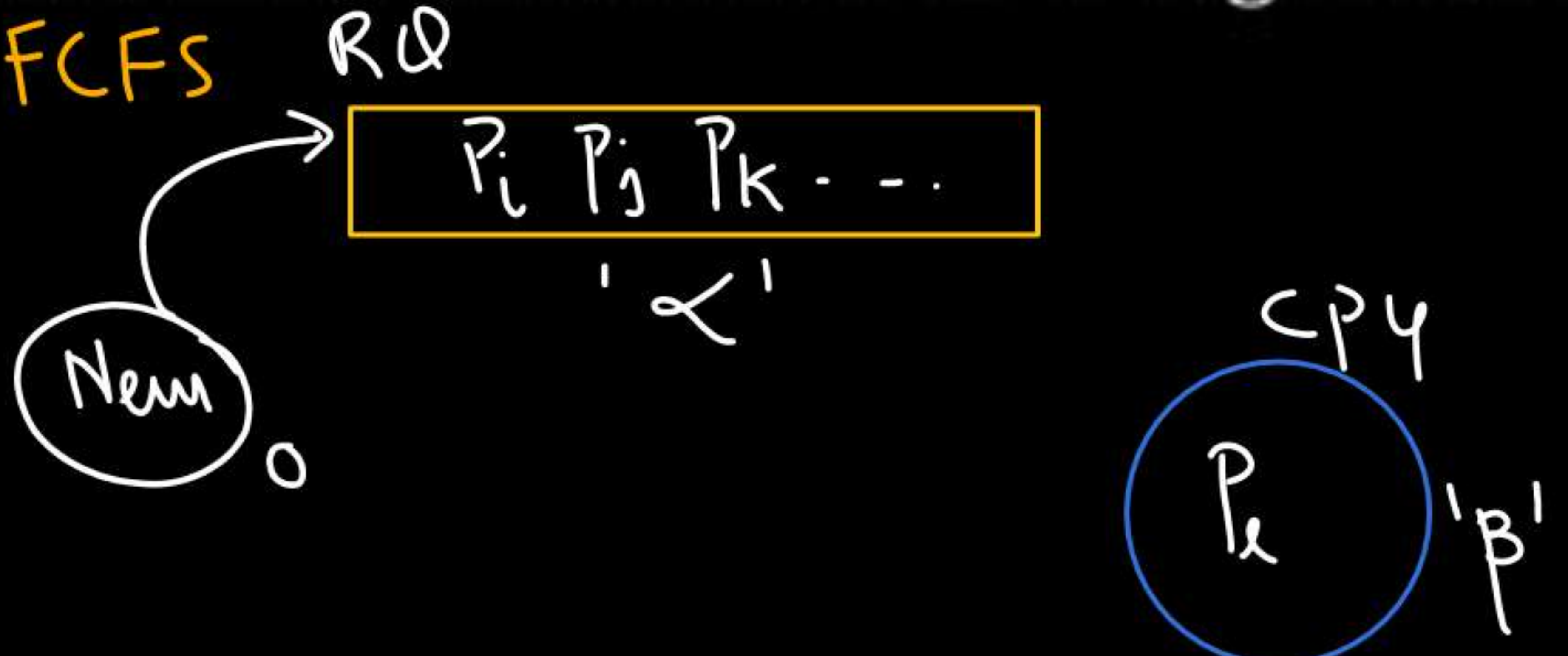
5. $\mu = 0$

Q. Consider a System using Preemptive Priority based scheduling with dynamically changing priorities. On its arrival a Process is assigned a priority of zero and Running Process Priority increases at the rate of 'β' and Priority of the Processes in the ready Q increases at the rate of 'α'. By dynamically changing the values of α and β one can achieve different Scheduling disciplines among the Processes. What discipline will be followed for the following conditions.

Text Book

1. $\beta > \alpha > 0$: FCFS
2. $\alpha < \beta < 0$:

LCFS



Q.

Consider Processes $P_1 \& P_2$ arriving in the ready queue at time 0 with following properties.

i) P1 needs a total of 12 units of CPU time and 20 units of I/O time. After every 3 units of CPU time $P1$ spends 5 units on I/O.

$P_1 \langle 3; 5; 3; 5; 3; 5; 3; 5 \rangle$

ii) $P2$ needs a total of 15 units of CPU time and no I/O. P2 arrives just after P1. $P_2 \langle 15 \rangle$

Compute the Completion times of $P_1 \& P_2$ using the following scheduling techniques:

1. SRTF
2. Round Robin with Time Quanta = 4 units

Q.

Three processes A, B and C each execute a loop of 100 iterations. In each iteration of the loop, a process performs a single computation that requires tc CPU milliseconds and then initiates a single I/O operation that lasts for t_{io} milliseconds. It is assumed that the computer where the processes execute has sufficient number of I/O devices and the OS of the computer assigns different I/O devices to each process. Also the scheduling overhead of the OS is negligible. The processes have the following characteristics:

Process Id	t_c	t_{io}
A	100 ms	500 ms
B	350 ms	500 ms
C	200 ms	500 ms

The processes A, B, and C are started at times 0,5 and 10 milliseconds respectively in a pure time-sharing system (round robin scheduling) that uses a time slice of 50 milliseconds. The time in milliseconds at which process C would complete its first I/O operation is

III. I.P.C & [Process Synchronization / Process Coordination]

Inter-Process Communication

MINIX


THANK YOU
GW
SOLDIERS !