OPERATING SYSTEM (CS-206)

Lecture-4

JAISHREE MAYANK

ASSISTANT PROFESSOR

IIITDM Kancheepuram

Preemptive Scheduling Algorithm

- **SRTF**
- RR
- Priority

Different Time based Parameters w.r.t. Process

- Arrival Time (AT) - Time the Process comes to the Ready State
- Burst Time (BT) – Execution time of the process – also referred as Service Time
- Completion Time (CT) – Time at which process completes its execution.
- Turn Around Time (TAT) - Time required for an application (process) to give an output to the end user
- TAT = CT - AT
- Waiting Time (WT) - Time Difference between turn around time and burst time.
- WT = TAT – BT
- Response Time – Time for the System to Respond to Process or User (First Response time on System Clock)
  - Time Since the Request is Submitted (AT) and the First Response Time

Shortest-First Job-With Preemption
SRTF-Shortest Remaining Time First

- If the new process arrives with a shorter burst than remaining of current process then schedule a new process
- Reducing average waiting time and average response time
- Not Practical

SRTF

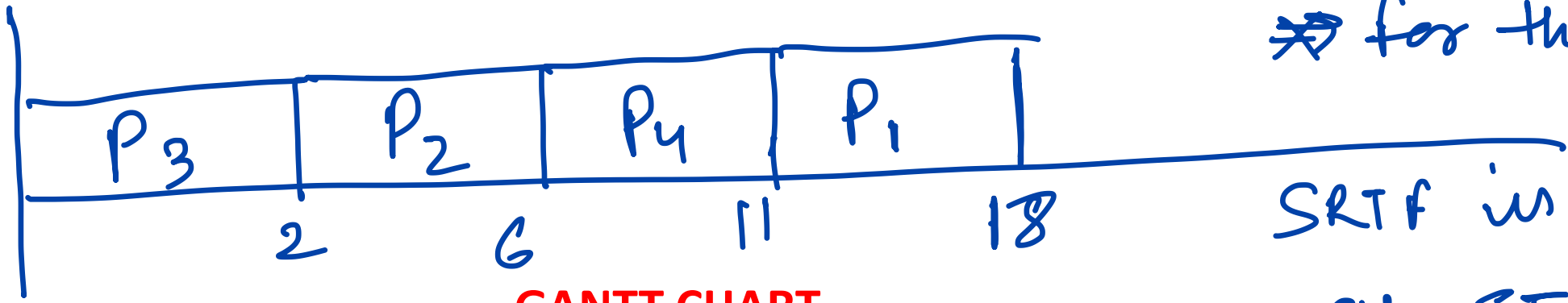
PID/P#	AT	BT	CT/FT	TAT	WT
1	0	7			
2	0	4			
3	0	2			
4	0	5			

Ready queue

P_1	P_2	P_3	P_4

Choose the shortest Burst -time.

⇒ for this example



GANTT CHART

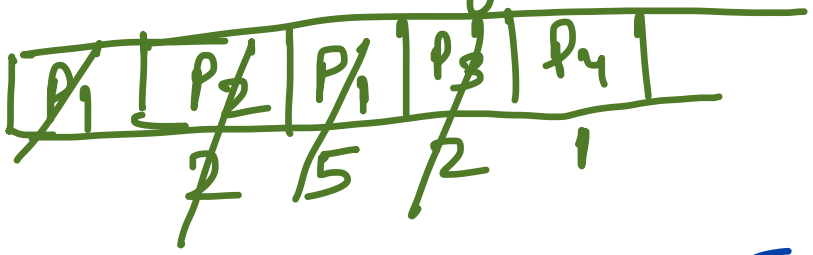
SRTF is same as SJF

* In every time point scheduler is running to check which processes has SRT
* If current running process and a new process in ready queue has same SRT, continue the same current process. Current process has chosen over new based on its arrival and process id.

SRTF

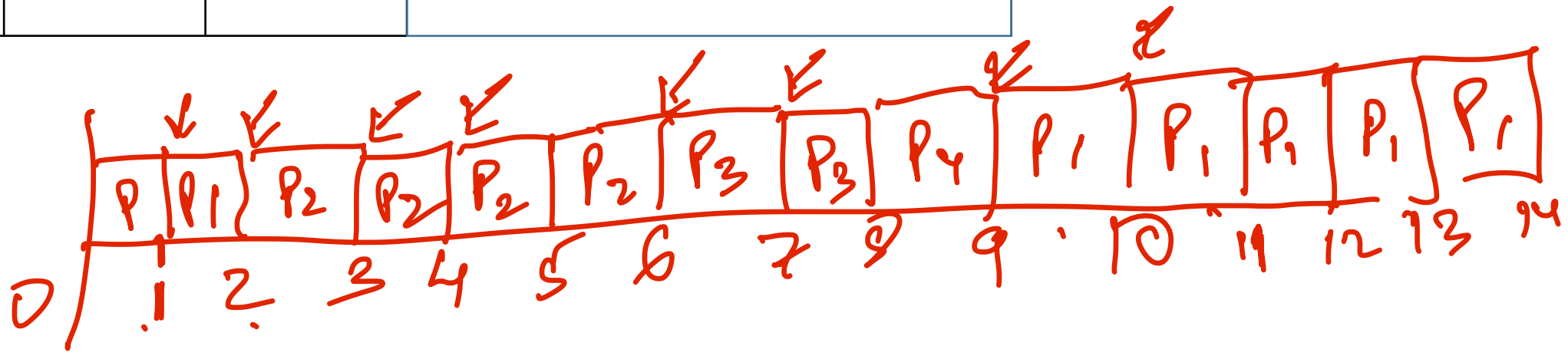
PID/P#	AT	BT	CT/FT	TAT (FT-AT)	WT	RT
1	0	7	14	14	7	0
2	2	4	6	4	0	0
3	4	2	8	4	2	2
4	7	1	9	2	1	1
				Aw = 6	2.5	0.75

(ST-AT) . Ready queue



Av. wt = 2.5

Av rt = 0.75



Round-Robin Scheduling approach

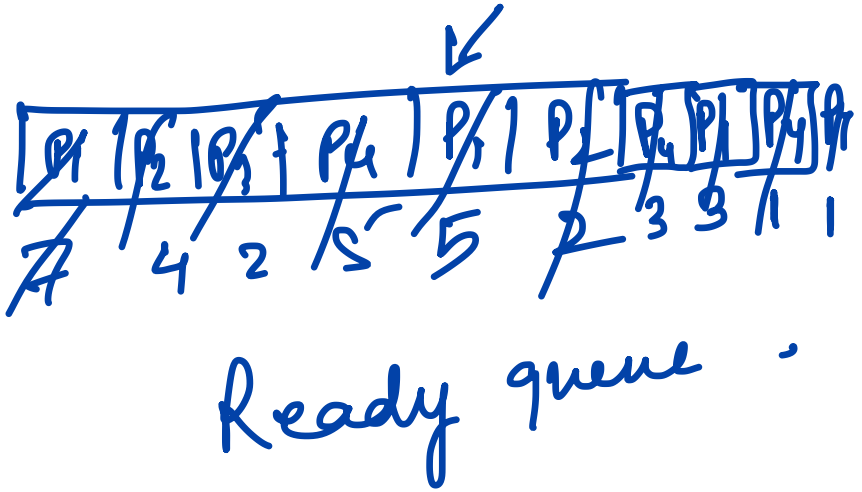
- Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than (*n*-1)*q* time units.
- Preemptive approach
- Performance
  - *q* large ⇒ FIFO
  - *q* small ⇒ *q* must be large with respect to context switch, otherwise overhead is too high

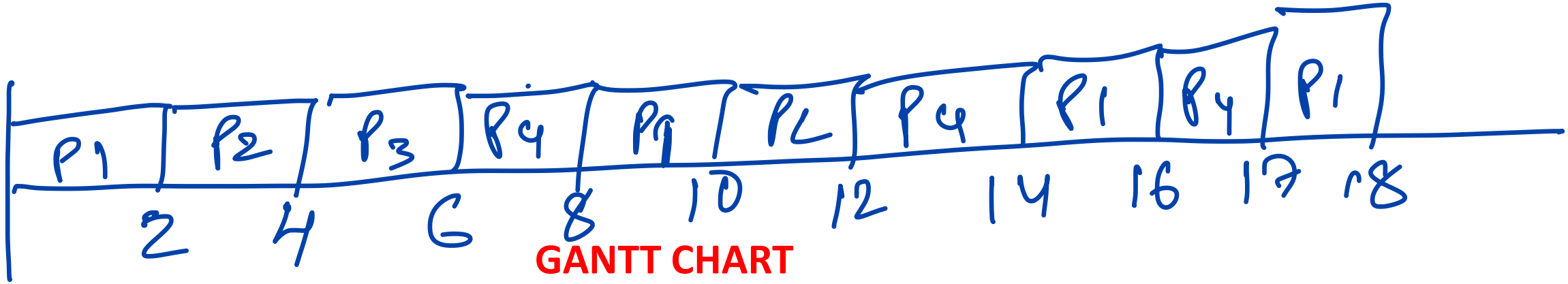
* In every 'q' time quantum, RR scheduler runs

$t_q = 2$

RR

PID/P#	AT	BT	CT/FT	TAT	WT	RT
1	0	7	18	18	11	0
2	0	4	12	12	8	2
3	0	2	6	6	4	4
4	0	5	17	17	12	6
				Av	8.75	3

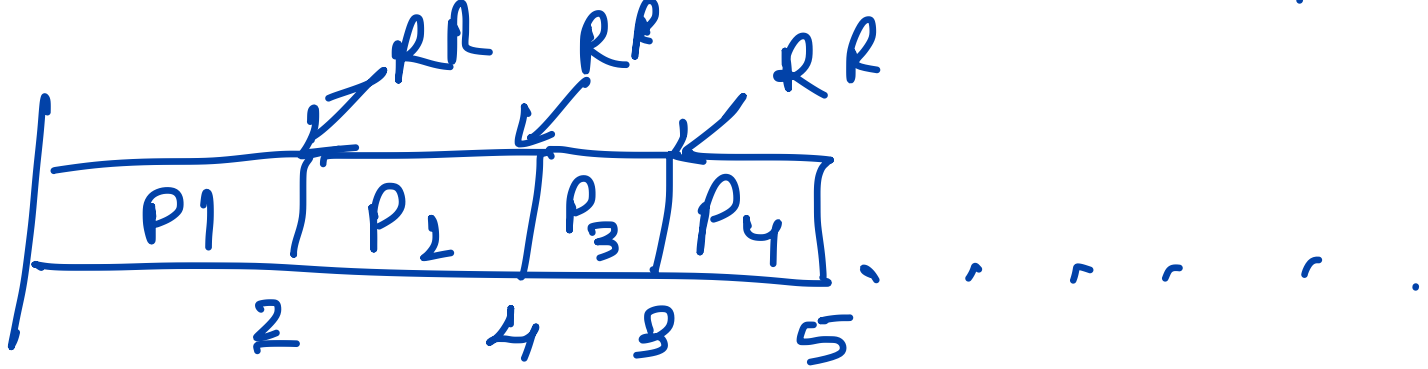




GANTT CHART

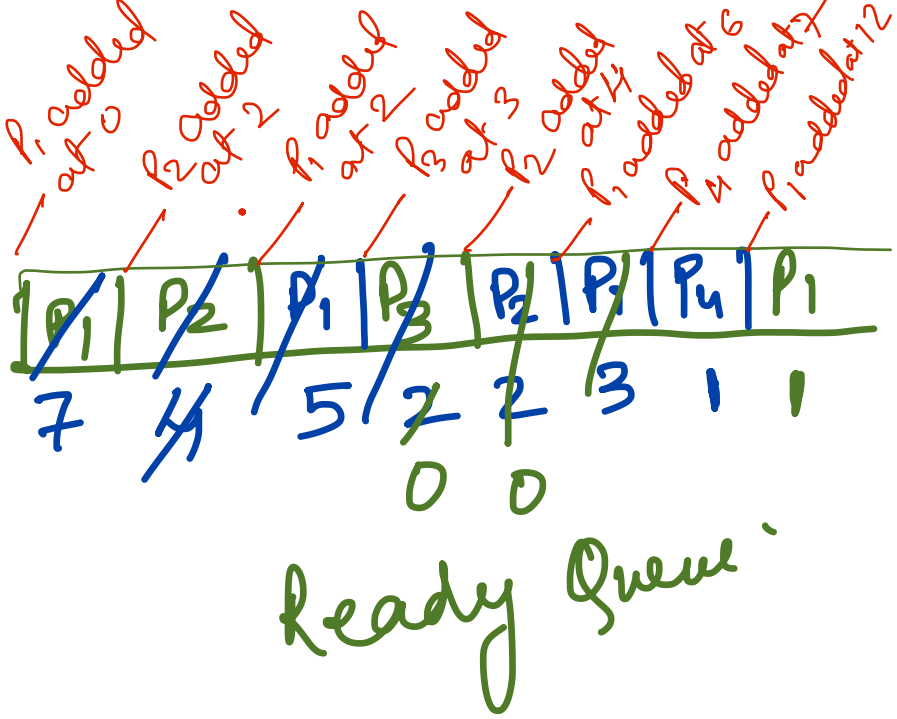
Note : (Special Scenario)

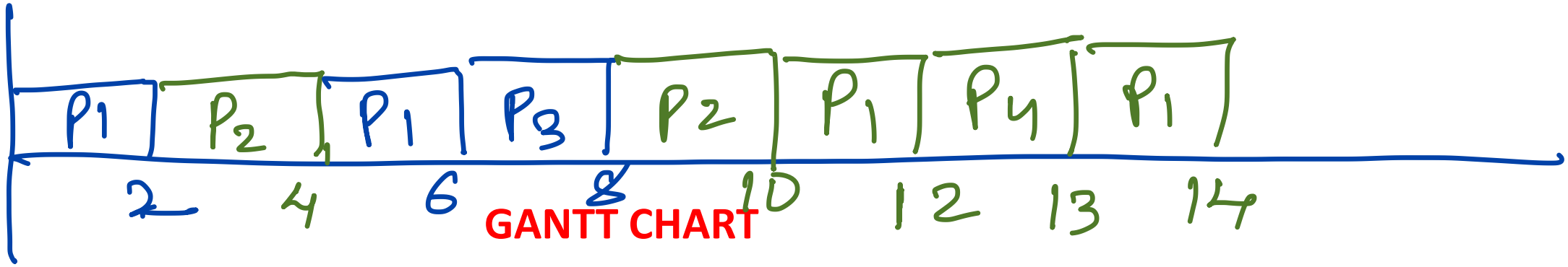
* If we have given 'q' time quantum for each processes, but suppose a process has completed before its quantum, then we remove that process from the queue and execute the RR Scheduler to choose another process.



quantum = 2 time in RR

PID/P#	AT	BT	CT/FT	TAT	WT	RT
1	0	7	14	14	7	0
2	2	4	10	8	4	0
3	3	2	8	5	3	3
4	7	1	13	6	5	5
			Avg		4.75	2





GANTT CHART