CSCI 4730/6730 OS
(Chap #5 CPU Scheduling – Part II)

In Kee Kim

Department of Computer Science

University of Georgia

Where are we?

- Scheduling for Single Processor (w/ Single Core)
  - FCFS (First-Come First-Served)
  - Shortest-Job-First (SJF)
    - Non Preemptive and Preemptive
  - Round Robin (RR)
  - Priority Scheduling
  - Multilevel Queue Scheduling
  - Multilevel Feedback Queue Scheduling

FCFS

Case #1

Process	CPU Burst Time
P_1	24
P_2	3
P_3	3

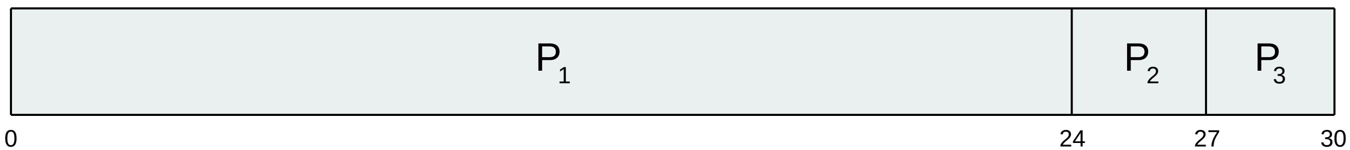
Case #2

Process	CPU Burst Time
P_2	3
P_3	3
P_1	24

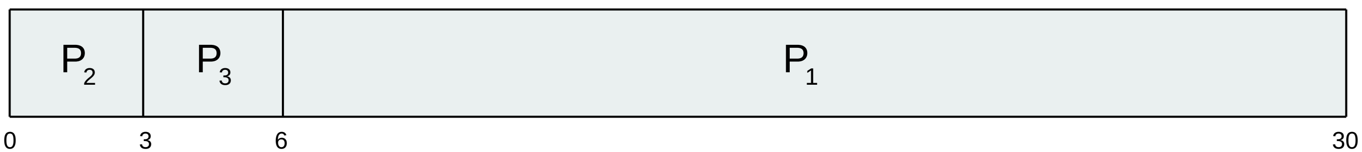
- Wait time for P_1, P_2, and P_3? And Average wait time?
- Turnaround time and Throughput for P_1, P_2, and P_3?
- Average Turnaround time and Throughput?

FCFS

- Case #1



- Case #2



- Wait time?
- Turnaround time?
- Throughput?

FCFS

- Pros and Cons

Pros	Cons
• Intuitive • Easy to implement	• Waiting time not likely to be minimal • “Head of line” blocking: Lots of small processes can get stuck behind big one

- How to improve FCFS?

SJF (Shorted-Job-First)

- As the name suggests, scheduling/processing the shortest job (CPU burst) first
- Preemptive version called **shortest-remaining-time-first**
- SJF is ***optimal*** in terms of giving minimum average waiting time for a given set of processes
- Assumption is "Scheduler knows the process execution time" – **too strong!**

SJF (Shorted-Job-First)

- Question - How do we determine the length of the next CPU burst?
  - Could ask the user
  - Prediction

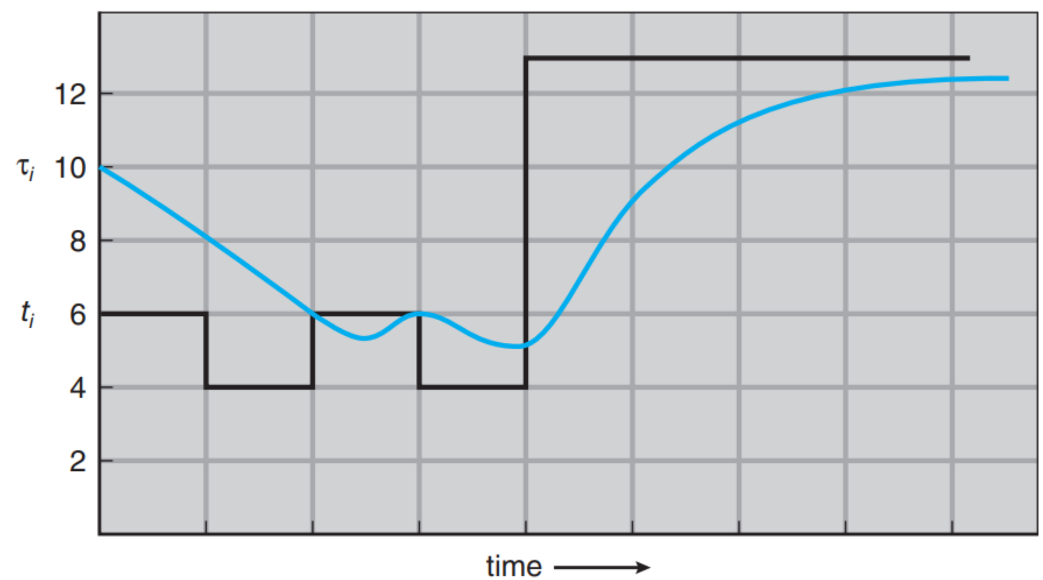
Execution Time Prediction

- Open-ended problem
- Exponential Average (as per text book)

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

$\boldsymbol{True_n}$ $\boldsymbol{Pred_n}$

How to determine α?



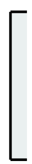
CPU burst (t_i)		6	4	6	4	13	13	13	…
"guess" (τ_i)	10	8	6	6	5	9	11	12	…

Figure 5.4 Prediction of the length of the next CPU burst.

SJF Example

Process	CPU Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

Assume that all arrive at 0.

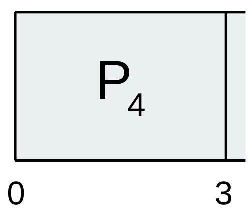

0

At time 0, among P1, P2, P3, and P4, which one has the shortest CPU time?

SJF Example

Process	CPU Burst Time
P_1	6
P_2	8
P_3	7
~~P_4~~	~~3~~

Assume that all arrive at 0.



At time 3, among P1, P2, and P3, which one has the shortest CPU time?

SJF Example

Process	CPU Burst Time
~~P_1~~	~~6~~
P_2	8
P_3	7
~~P_4~~	~~3~~

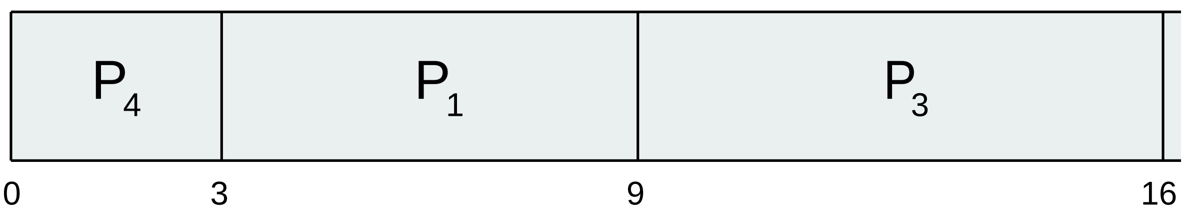
Assume that all arrive at 0.

At time 9, between P2, and P3, which one has the shortest CPU time?

SJF Example

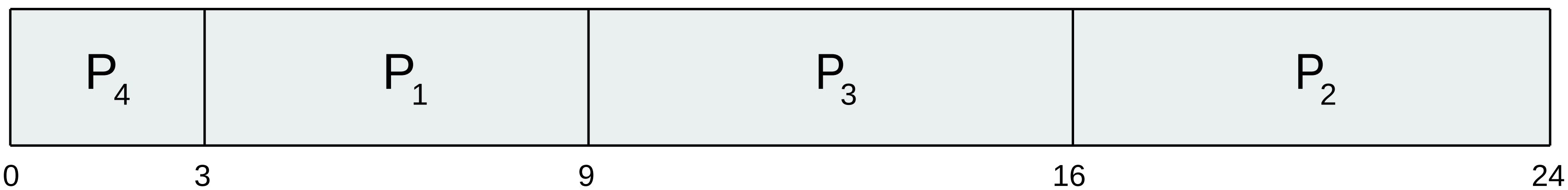
Process	CPU Burst Time
~~P_1~~	~~6~~
P_2	8
~~P_3~~	~~7~~
~~P_4~~	~~3~~

Assume that all arrive at 0.



At time 16, P2 is the only remaining job

SJF Example



- Waiting time for P_1, P_2, P_3 and P_4?
  - $P_1 = 3$, $P_2 = 16$, $P_3 = 9$, $P_4 = 0$
- Average waiting time?
- Throughput?
- Turnaround time?

Process	CPU Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

Shortest Remaining Time First

- **Preemptive** version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
P_1	0	8	
P_2	1	4	
P_3	2	9	
P_4	3	5	

SRTF Example

- Preemptive version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
$\boldsymbol{P_1}$	**0**	**8**	**8**
P_2	1	4	
P_3	2	9	
P_4	3	5	

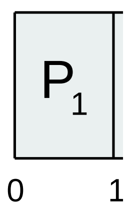
0

At time 0, P1 is the only process in the ready queue.

SRTF Example

- Preemptive version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
P_1	**0**	**8**	
P_2	**1**	**4**	
P_3	2	9	
P_4	3	5	

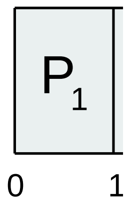


At time 1, P2 arrives, which is one has shorter remaining time?

SRTF Example

- Preemptive version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
P_1	**0**	**8**	**7**
P_2	**1**	**4**	**4**
P_3	2	9	
P_4	3	5	

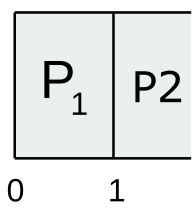


At time 1, P2 arrives, which is one has shorter remaining time?

SRTF Example

- Preemptive version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
P_1	**0**	**8**	
P_2	**1**	**4**	
P_3	**2**	**9**	
P_4	3	5	

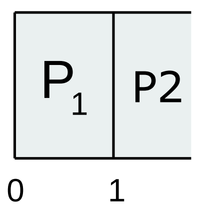


At time 2, P3 arrives, which is one has shorter remaining time?

SRTF Example

- Preemptive version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
P_1	0	8	7
P_2	1	4	3
P_3	2	9	9
P_4	3	5	

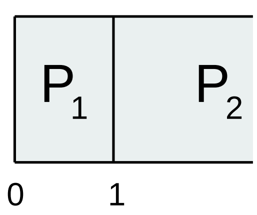


At time 2, P3 arrives, which is one has shorter remaining time?

SRTF Example

- Preemptive version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
P_1	0	8	
P_2	1	4	
P_3	2	9	
P_4	3	5	

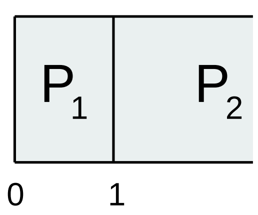


At time 3, P4 arrives, which is one has shorter remaining time?

SRTF Example

- Preemptive version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
P_1	**0**	**8**	**7**
P_2	**1**	**4**	**2**
P_3	**2**	**9**	**9**
P_4	**3**	**5**	**5**

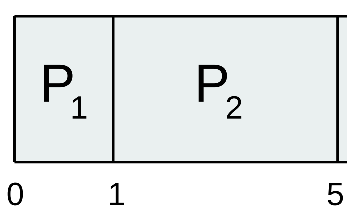


At time 3, P4 arrives, which is one has shorter remaining time?

SRTF Example

- Preemptive version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
P_1	0	8	
~~P_2~~	~~1~~	~~4~~	
P_3	2	9	
P_4	3	5	

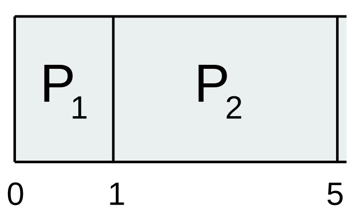


At time 5, which is one has shorter remaining time?

SRTF Example

- Preemptive version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
P_1	0	8	7
~~P_2~~	~~1~~	~~4~~	~~0~~
P_3	2	9	9
P_4	3	5	5

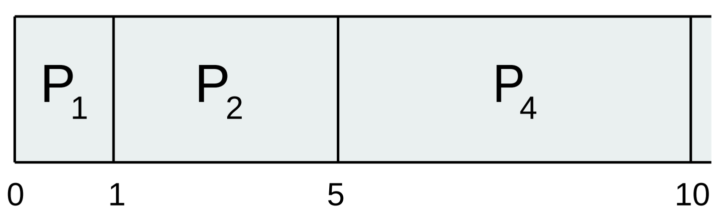


At time 5, which is one has shorter remaining time?

SRTF Example

- Preemptive version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
P_1	0	8	
~~P_2~~	~~1~~	~~4~~	
P_3	2	9	
~~P_4~~	~~3~~	~~5~~	

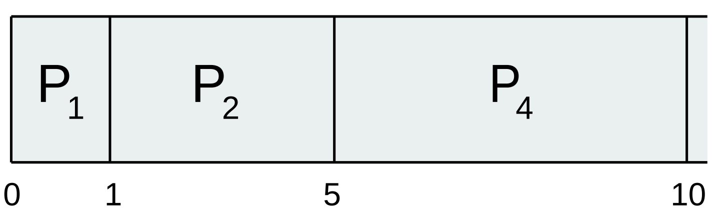


At time 10, which is one has shorter remaining time?

SRTF Example

- Preemptive version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
P_1	0	8	7
~~P_2~~	~~1~~	~~4~~	~~0~~
P_3	2	9	9
~~P_4~~	~~3~~	~~5~~	~~0~~

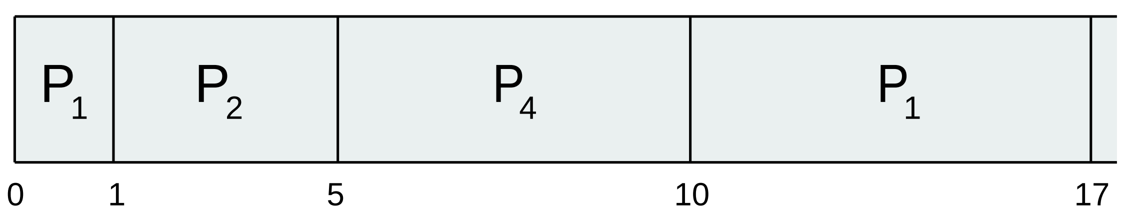


At time 10, which is one has shorter remaining time?

SRTF Example

- Preemptive version of SJF

Proc	Arrival Time	CPU Burst Time	Remaining Time
~~P_1~~	~~0~~	~~8~~	~~0~~
~~P_2~~	~~1~~	~~4~~	~~0~~
~~P_3~~	~~2~~	~~9~~	~~9~~
~~P_4~~	~~3~~	~~5~~	~~0~~



At time 17, only P3 is in the ready queue

SRTF Example

P_1	P_2	P_4	P_1	P_3

0 1 5 10 17 26

- Waiting time for P_1, P_2, P_3 and P_4?
  - $P_1 = (0\text{-}0) + 9$, $P_2 = 1\text{-}1$, $P_3 = 17\text{-}2$, $P_4 = 5\text{-}5$
- Average waiting time?
- Throughput?
- Turnaround time?

Proc	Arrival Time	CPU Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

SJF vs. SRTF

- Job (process) arrival sequence

Proc	Arrival Time	CPU Burst Time
P_1	0	3
P_2	1	8
P_3	2	6
P_4	4	4
P_5	5	2

- Average Wait Time, Average Turnaround Time?

SJF and SRTF

- Pros and Cons

Pros	Cons
• Minimal waiting time (preemptive SJF) • Better response time (over FCFS)	• Hard to predict execution time (remaining time) • Hard to implement (not practically feasible) • Mostly (but not all) existing in research papers and textbook

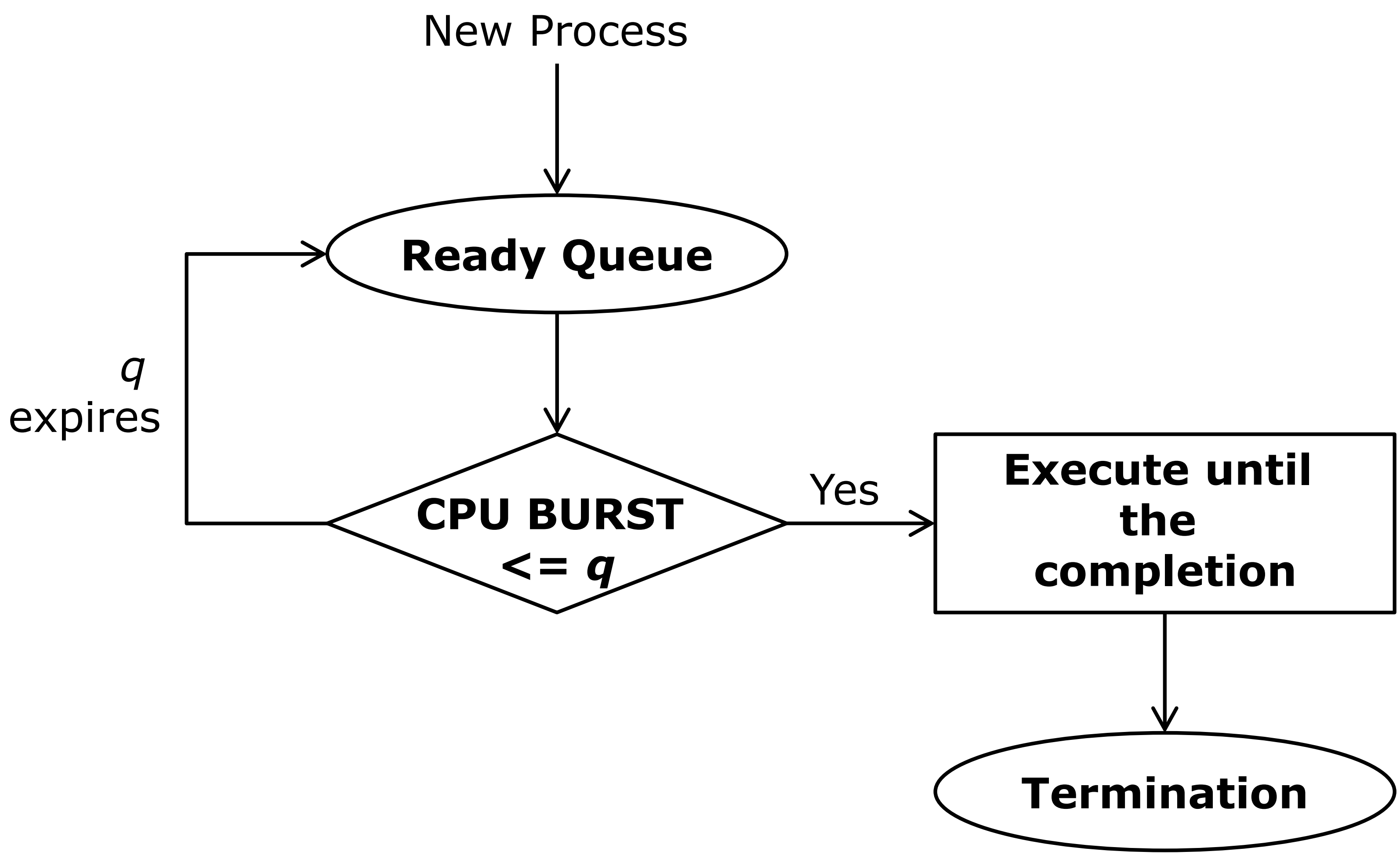
Round Robin

- **FCFS + Preemption + Time Quantum (*time slice*, q)**
- Each process gets a small unit of CPU time (*q*), usually 10 -- 100 milliseconds.
- After this time has elapsed, the process is preempted and added to the end of the ready queue.

Round Robin

- If there are n processes in the ready queue and the time quantum is q
  - 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.
- Timer interrupts every quantum to schedule next process

Round Robin


New Process
Ready Queue
q
expires
CPU BURST
<= q
Yes
Execute until
the
completion
Termination

Round Robin

Proc	CPU Burst Time
P_1	24
P_2	3
P_3	3

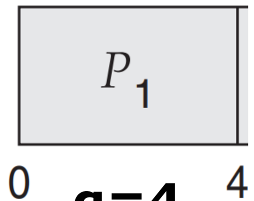
All processes arrive at 0
q (time quantum) = 4

P_1
0

Round Robin

Proc	CPU Burst Time
P_1	24
P_2	3
P_3	3

All processes arrive at 0
q (time quantum) = 4

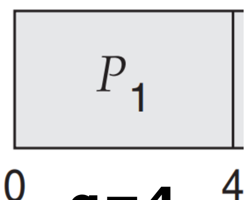


q=4

Round Robin

Proc	CPU Burst Time
P_1	21
P_2	3
P_3	3

All processes arrive at 0
q (time quantum) = 4

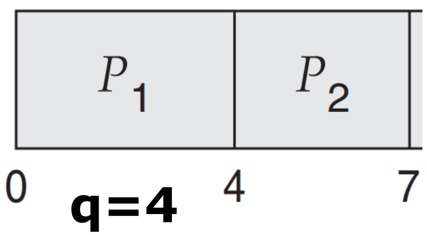


q=4

Round Robin

Proc	CPU Burst Time
P_1	21
~~P_2~~	~~3~~
P_3	3

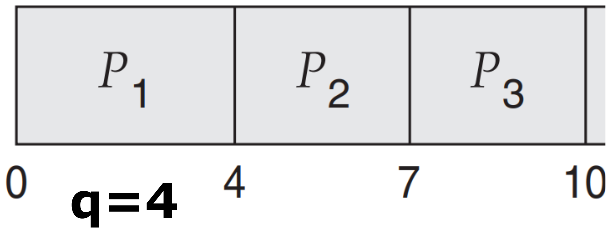
All processes arrive at 0
q (time quantum) = 4



Round Robin

Proc	CPU Burst Time
P_1	21
~~P_2~~	~~3~~
~~P_3~~	~~3~~

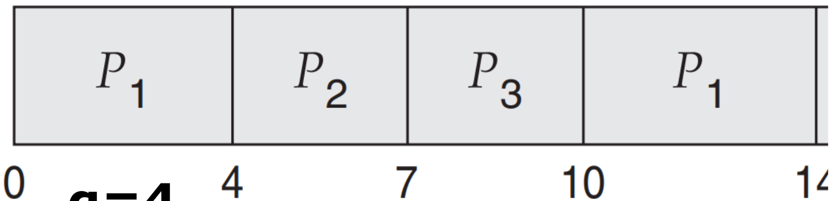
All processes arrive at 0
q (time quantum) = 4



Round Robin

Proc	CPU Burst Time
P_1	21
~~P_2~~	~~3~~
~~P_3~~	~~3~~

All processes arrive at 0
q (time quantum) = 4

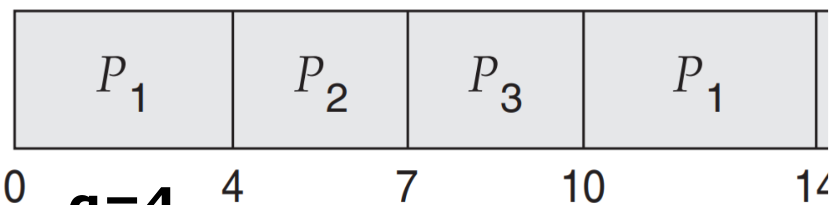


q=4

Round Robin

Proc	CPU Burst Time
P_1	17
~~P_2~~	~~3~~
~~P_3~~	~~3~~

All processes arrive at 0
q (time quantum) = 4

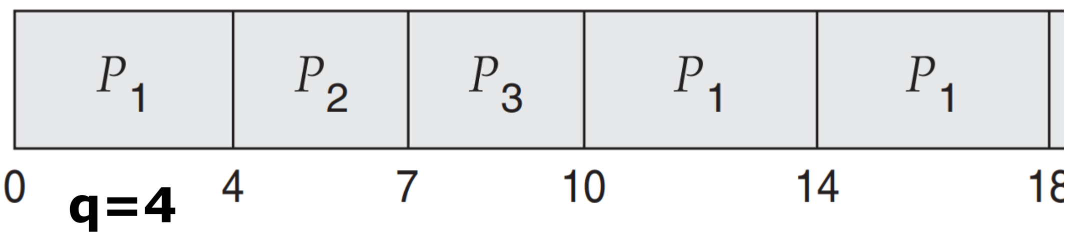


q=4

Round Robin

Proc	CPU Burst Time
P_1	17
~~P_2~~	~~3~~
~~P_3~~	~~3~~

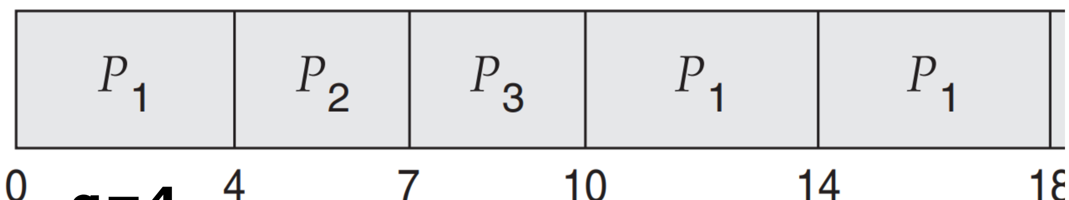
All processes arrive at 0
q (time quantum) = 4



Round Robin

Proc	CPU Burst Time
P_1	13
~~P_2~~	~~3~~
~~P_3~~	~~3~~

All processes arrive at 0
q (time quantum) = 4

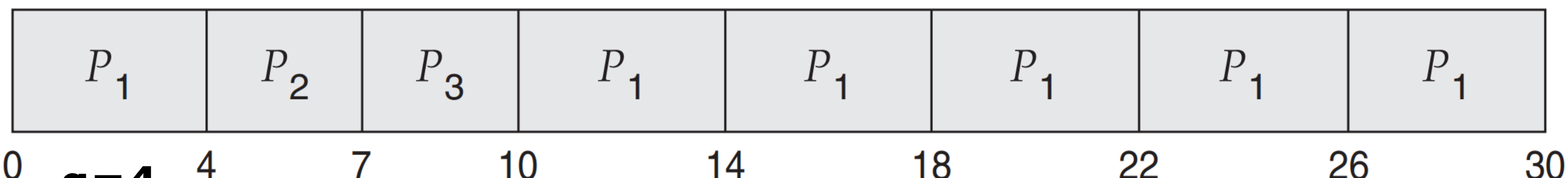


q=4

Round Robin

Proc	CPU Burst Time
~~P_1~~	~~13~~
~~P_2~~	~~3~~
~~P_3~~	~~3~~

All processes arrive at 0
q (time quantum) = 4



q=4

Another Example (Round Robin)

Proc	Arrival Time	CPU Burst Time
P_1	0	8
P_2	1	7
P_3	5	4
P_4	6	2
P_5	8	5

q=3

Round Robin

- Performance
  - q large $\Rightarrow$
  - q small $\Rightarrow$

Round Robin

- Performance
  - q large $\Rightarrow$ **FIFO**
  - q small $\Rightarrow$

Round Robin

- Performance
  - q large $\Rightarrow$ **FIFO**
  - q small $\Rightarrow$ ***q* must be large with respect to context switch, otherwise overhead is too high**

Quantum Size – Context Switch

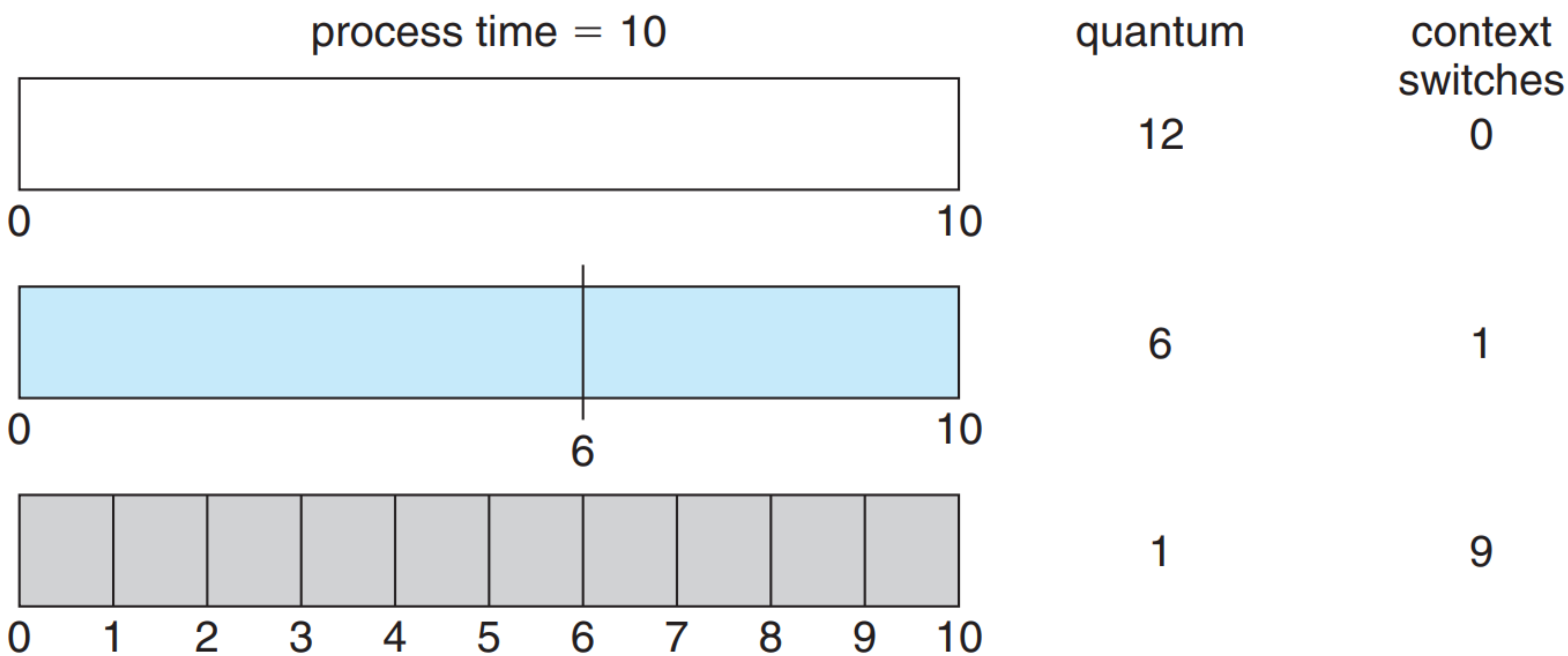


Figure 5.5 How a smaller time quantum increases context switches.

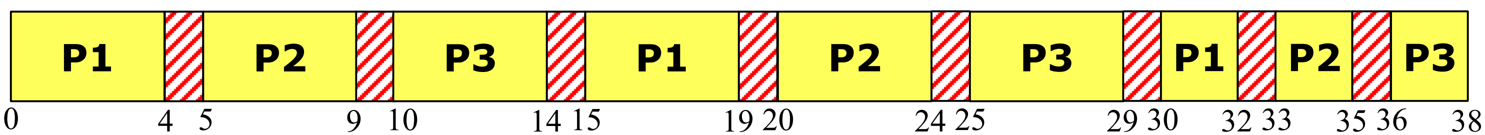
CPU Utilization??

Proc	CPU Burst Time
P_1	10
P_2	10
P_3	10

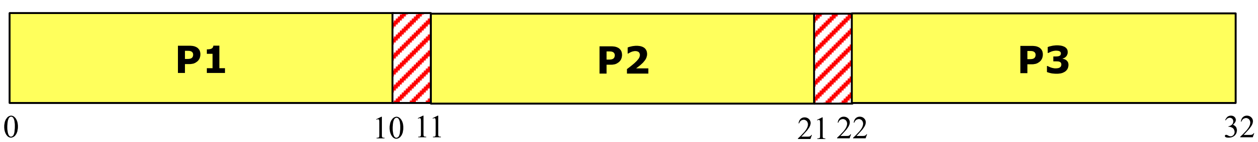
All processes arrive at 0
q (time quantum) = 4
Context switch time = 1

What is throughput?

- RR



- FCFS

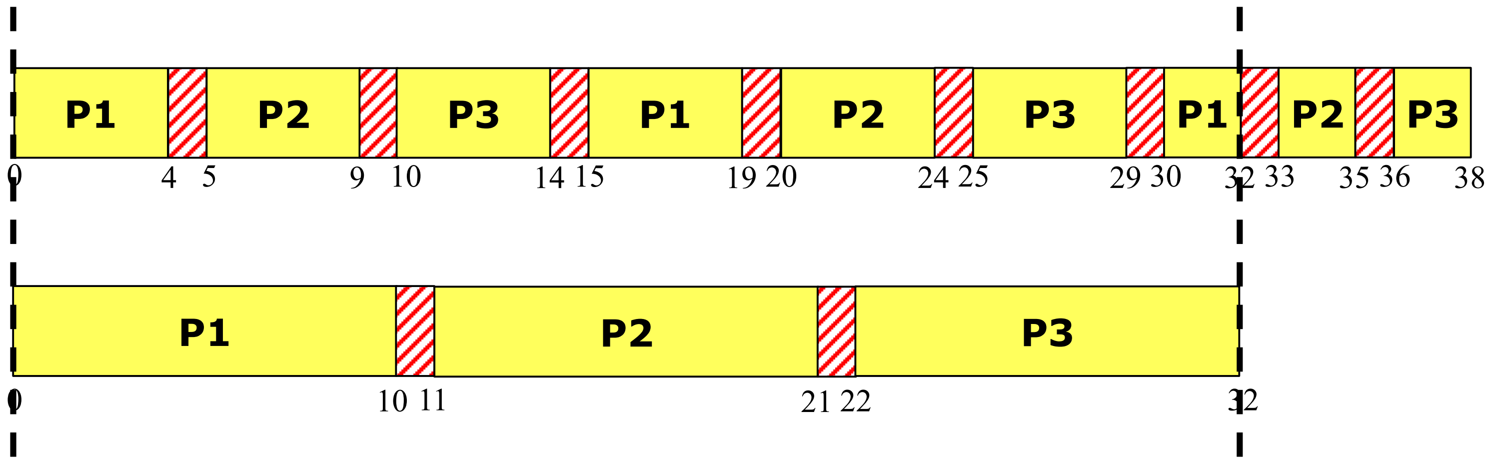


CPU Utilization??

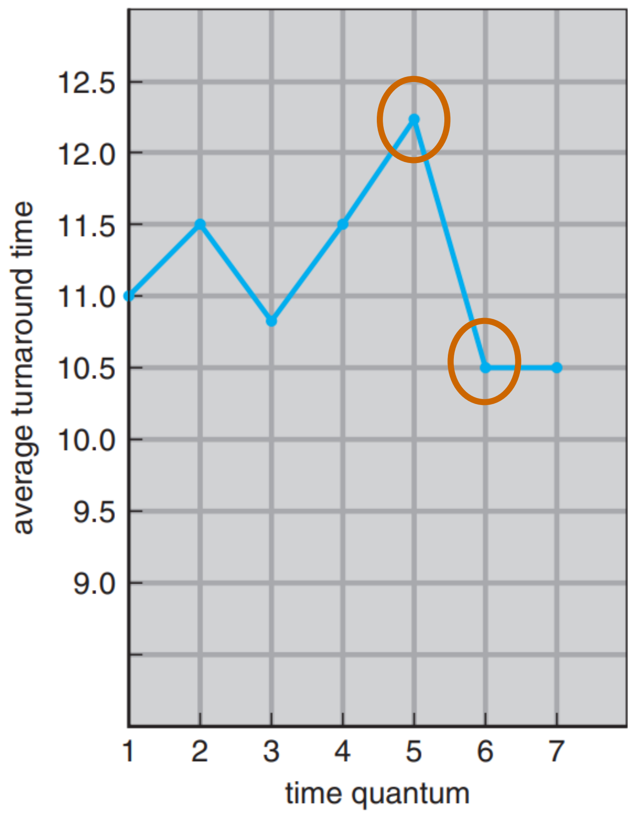
Proc	CPU Burst Time
P_1	10
P_2	10
P_3	10

All processes arrive at 0
q (time quantum) = 4
Context switch time = 1

- RR
- FCFS



Quantum Size – Turnaround Time



process	time
P_1	6
P_2	3
P_3	1
P_4	7

Figure 5.6 How turnaround time varies with the time quantum.

Round Robin

- Pros and Cons

Pros	Cons
• Better responsiveness • No “Head-of-Line” blocking • Easy to implement • Equal priority (??) • Fairness	• Performance replies on *q* • Determining the optimal *q* is difficult • Small *q* – High overhead, Low CPU utilization • Large *q* – FCFS