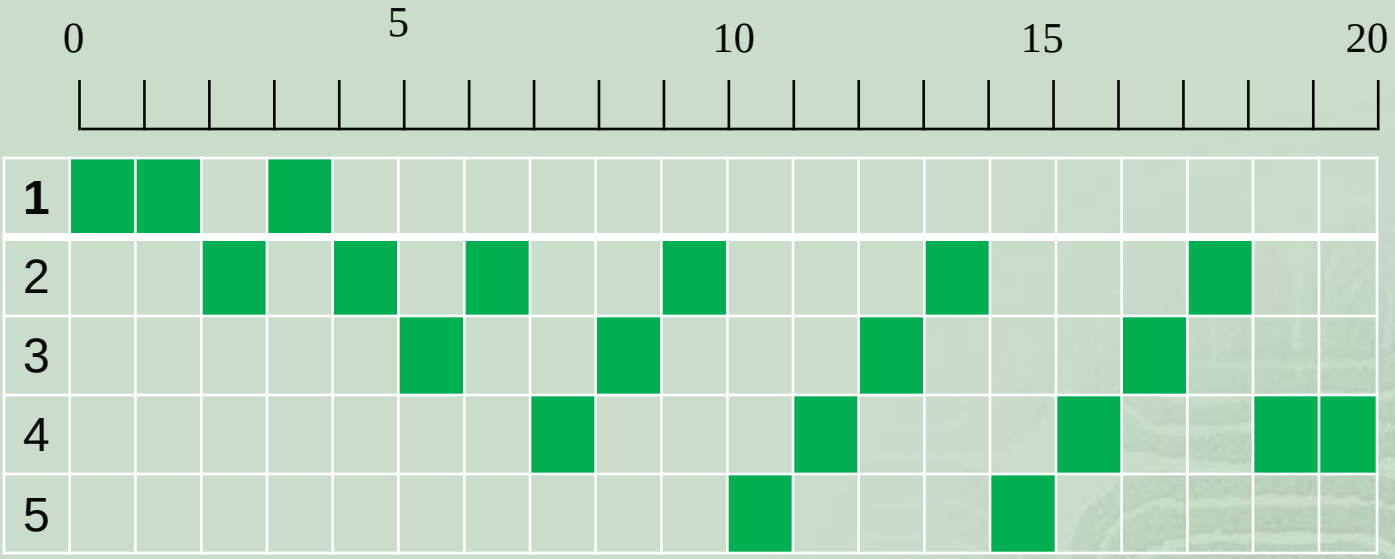
CPU Scheduling

Round Robin (RR)

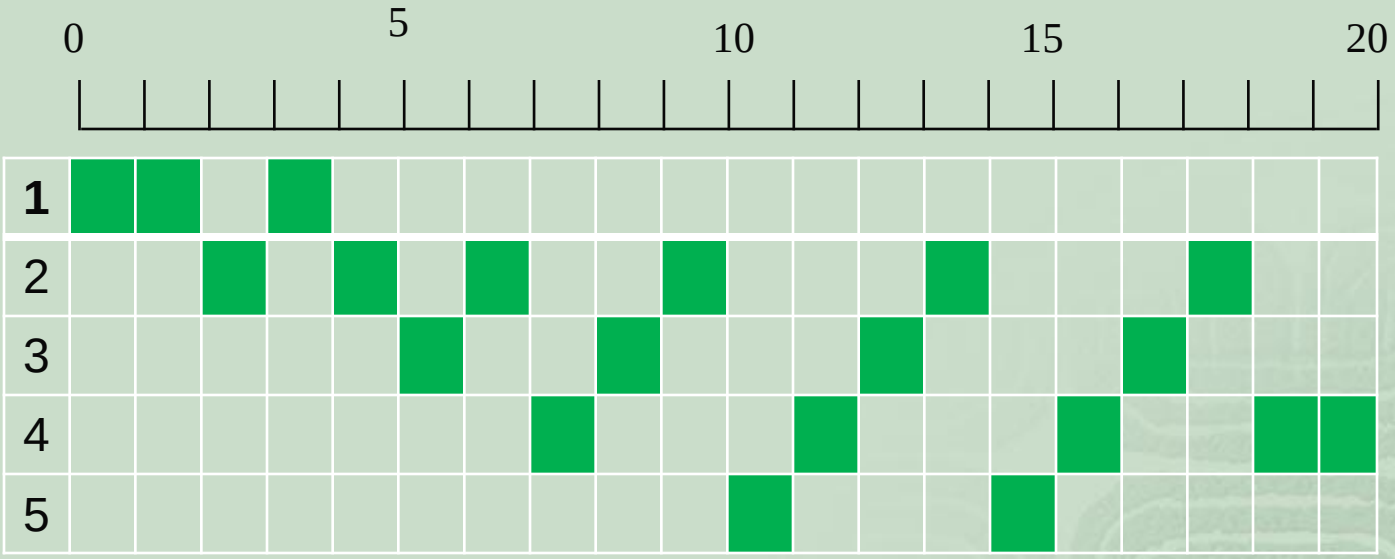
- Each process gets a small amount of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the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

Round Robin (continued…)



Process	Arrival	Service
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2

Round Robin (continued…)



Process	Arrival	Service
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2

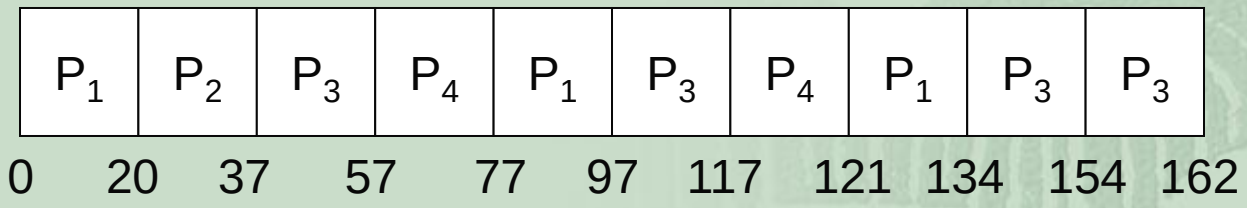
Suppose q = 1;

Turnaround time: p1: 4-0 = 4; p2: 18-2 = 16; p3: 17-4 = 13;
P4: 20 -6 = 14; p5: 15-8 = 7;
Ave TAT: (4+16+13+14+7)/5 = 8.8

Example of RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

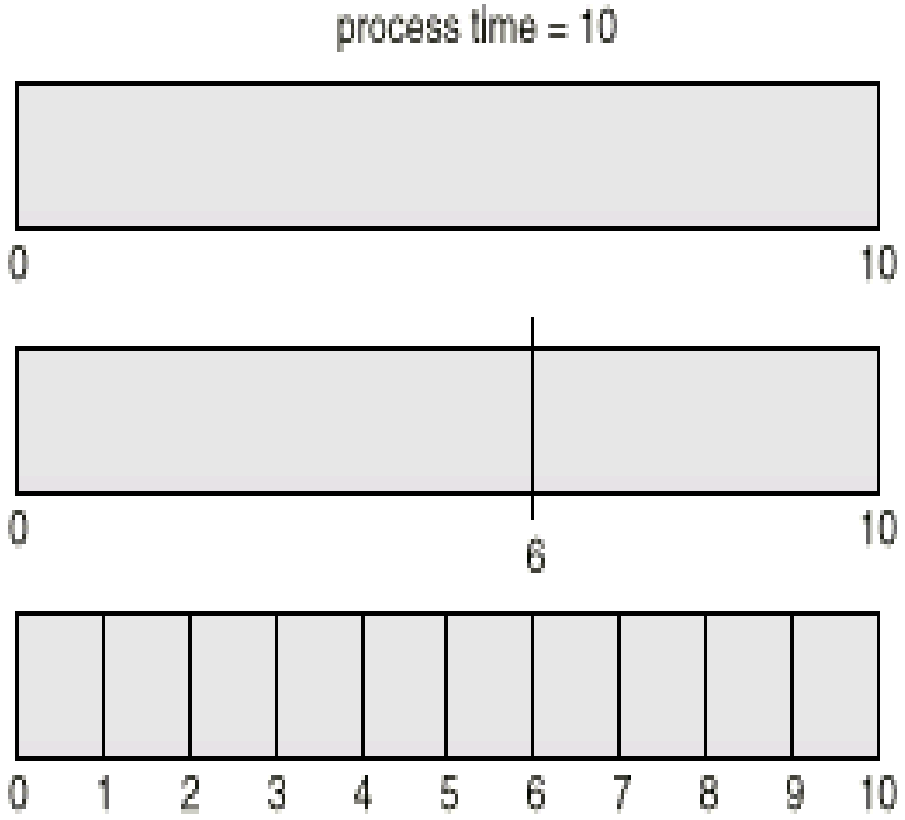
- The Gantt chart is:



Round Robin (continued…)

- Particularly effective in time-sharing or transaction processing system.
- Generally favors CPU bound processes as I/O bound give up their time slice while CPU bound use their complete time quantum.
- Could be improved using a virtual round robin (VRR) scheduling scheme – implement an auxiliary I/O queue which gets preference over the main queue.

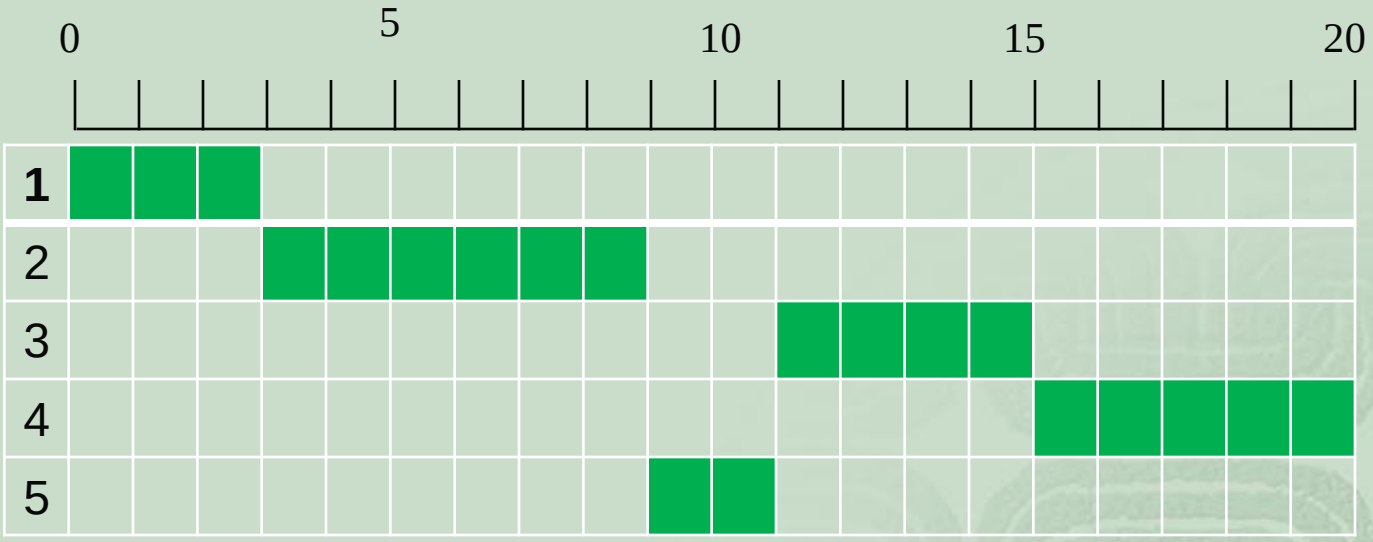
Time Quantum and Context Switch Time



quantum	context switches
12	0
6	1
1	9

Shortest Process Next (SPN)

Process	Arrival	Service
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2

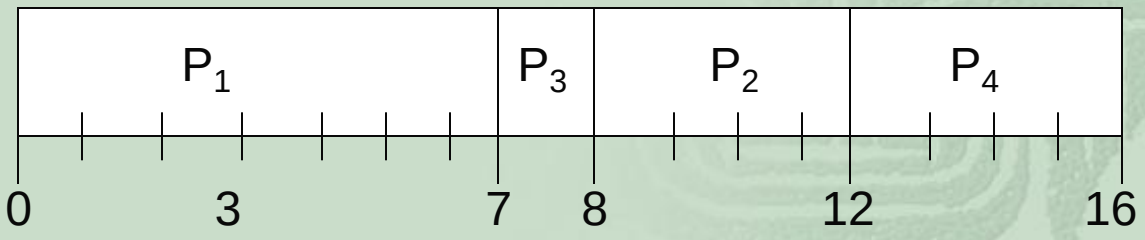


- Shortest Process Next - Non-preemptive policy
- Process with shortest expected processing time is selected next
- Short process jumps ahead of longer processes

Example of Non-Preemptive SPN

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SPN (non-preemptive)



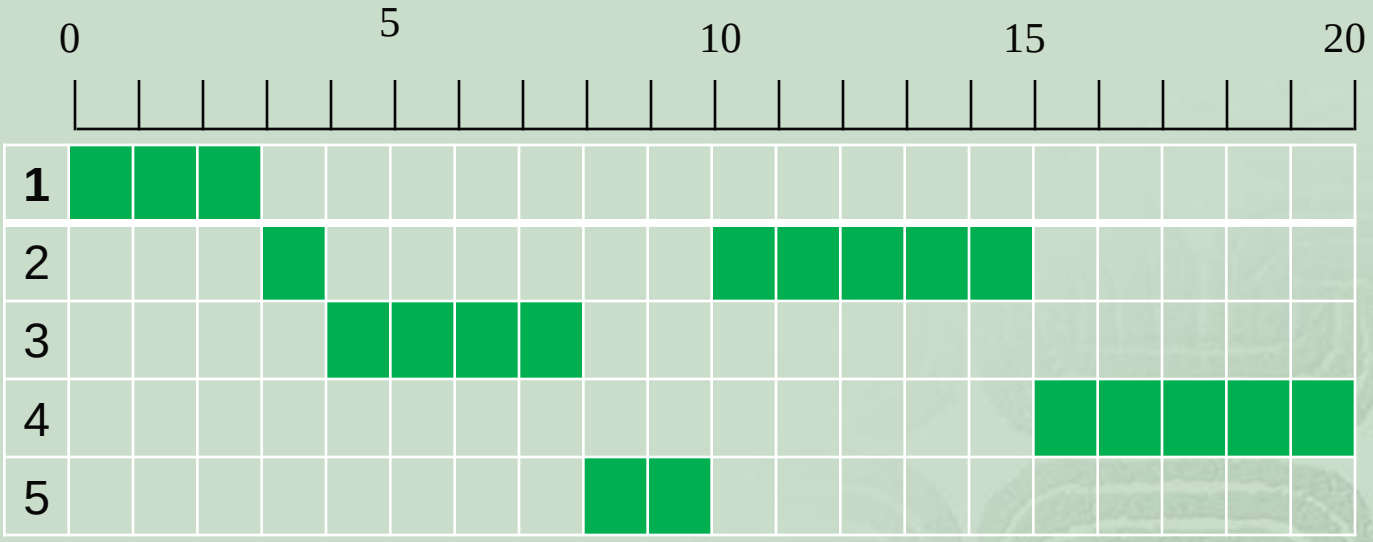
- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

Shortest Process Next (continued…)

- May be impossible to know or at least estimate the required processing time of a process.
  - For batch jobs, require a programmer's estimate.
    - If estimate is substantially off, system may abort job.
  - In a production environment, the same jobs run frequently and statistics may be gathered.
  - In an interactive environment, the operating system may keep a running average of each "burst" for each process.
- SPN could result in starvation of longer processes if there is a steady supply of short processes.
- Not suitable for time-sharing or transaction processing environment because of lack of preemption.

Shortest Remaining Time (SRT)

Process	Arrival	Service
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2

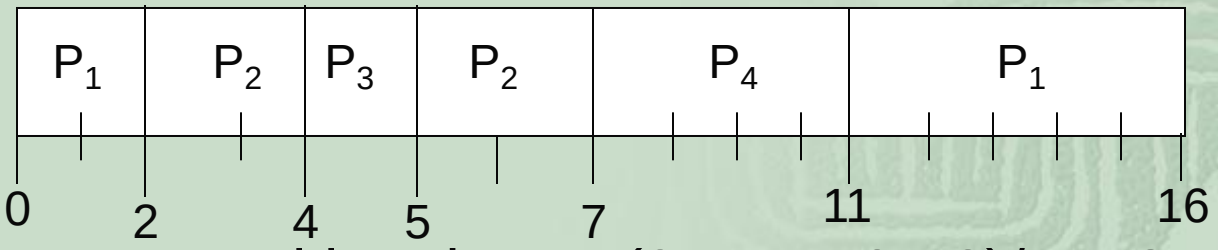


- The shortest remaining time (SRT) policy is a preemptive version of SPN.
- Must estimate expected remaining processing time

Example of Preemptive SRT

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SRT (preemptive)



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

Shortest Remaining Time
(continued…)

- When a new process joins the ready queue, it may have a shorter remaining time and preempts the current process.
- SRT does not bias in favor of long processes (as does FCFS)
- Unlike RR, no additional interrupts are generated reducing overhead.
- Superior turnaround performance to SPN, because a short job is given immediate preference to a running longer job.