ECE437/CS481

M03A: CPU SCHEDULING
SCHEDULING CONCEPT & ALGORITHMS

CHAPTER 6.1-6.3

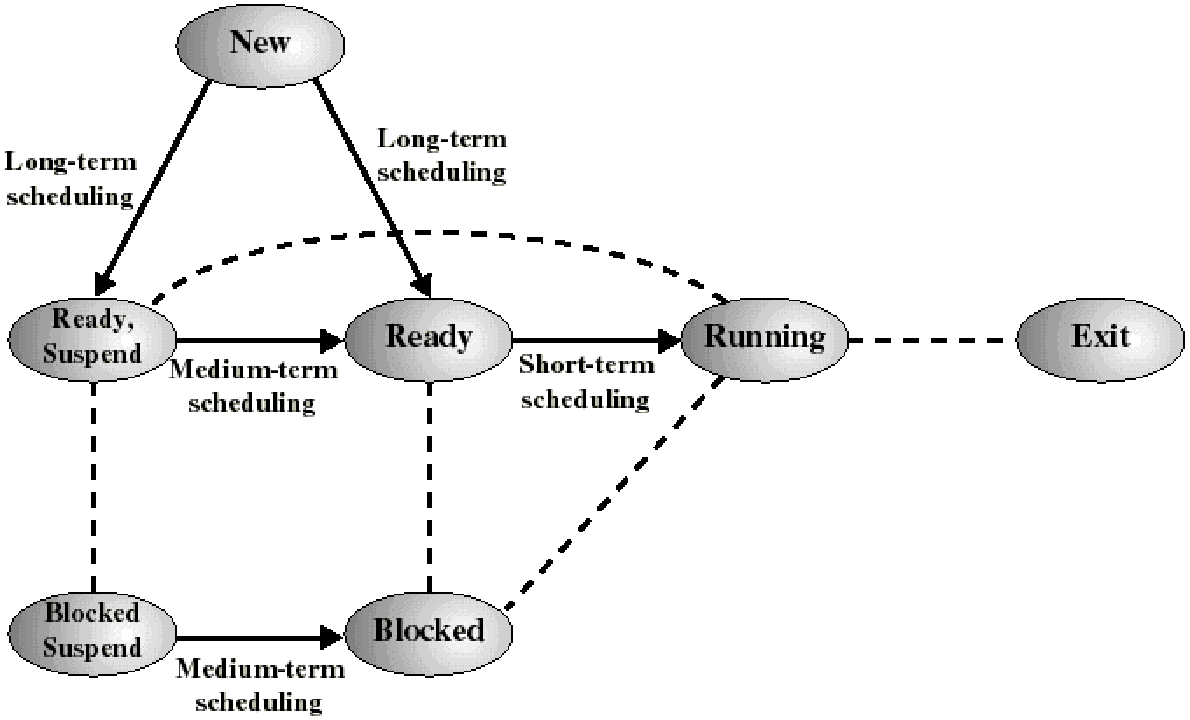
Xiang Sun

The University of New Mexico

Levels of Scheduling

❑ Long-term scheduling/Job scheduler (High-level)

- A long-term scheduler determines which programs are admitted to the system for processing (i.e., loaded into the main memory).
- Coarse-grained control of the degree of multiprogramming
- Should balance different types of processes:
  - I/O-intensive process: spends more time doing I/O than computations, many short CPU bursts.
  - CPU-intensive process: spends more time doing computations; few very long CPU bursts.



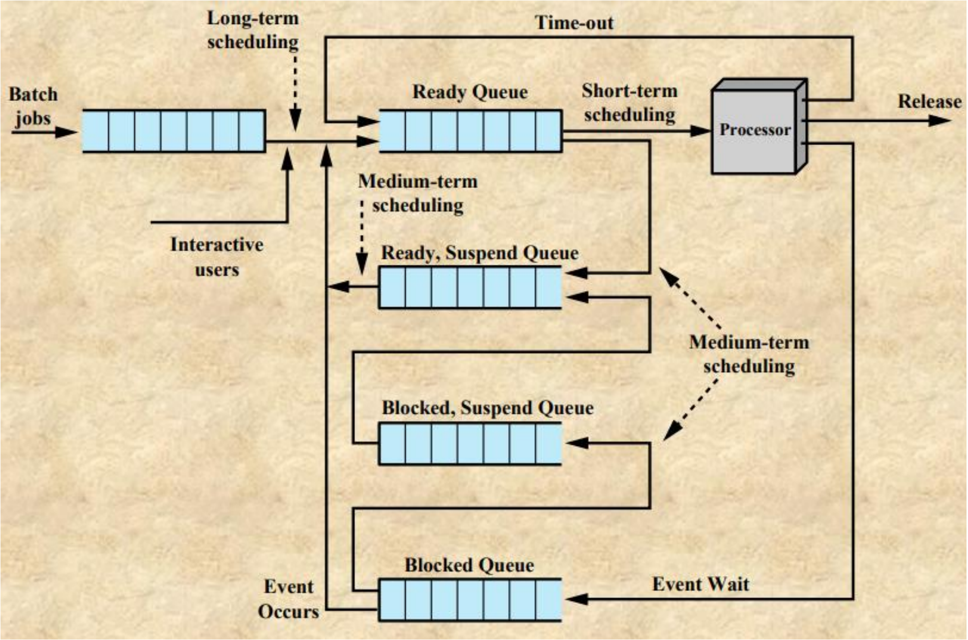
Levels of Scheduling

❑ Medium-term scheduling (swapping in/out)

- Adjust the degree of multiprogramming by swapping.

- Swapping: removes a process from memory, stores on disk, and bring back in from disk to continue execution later on.

  - ✓ Swap out: memory-to-disk
  - ✓ Swap in: disk-to-memory



- Normally, the medium-term scheduler may decide to swap out a process which has not been active for some time, or a process which has a low priority, or a process which is taking up a large amount of memory in order to free up main memory for other processes, etc.

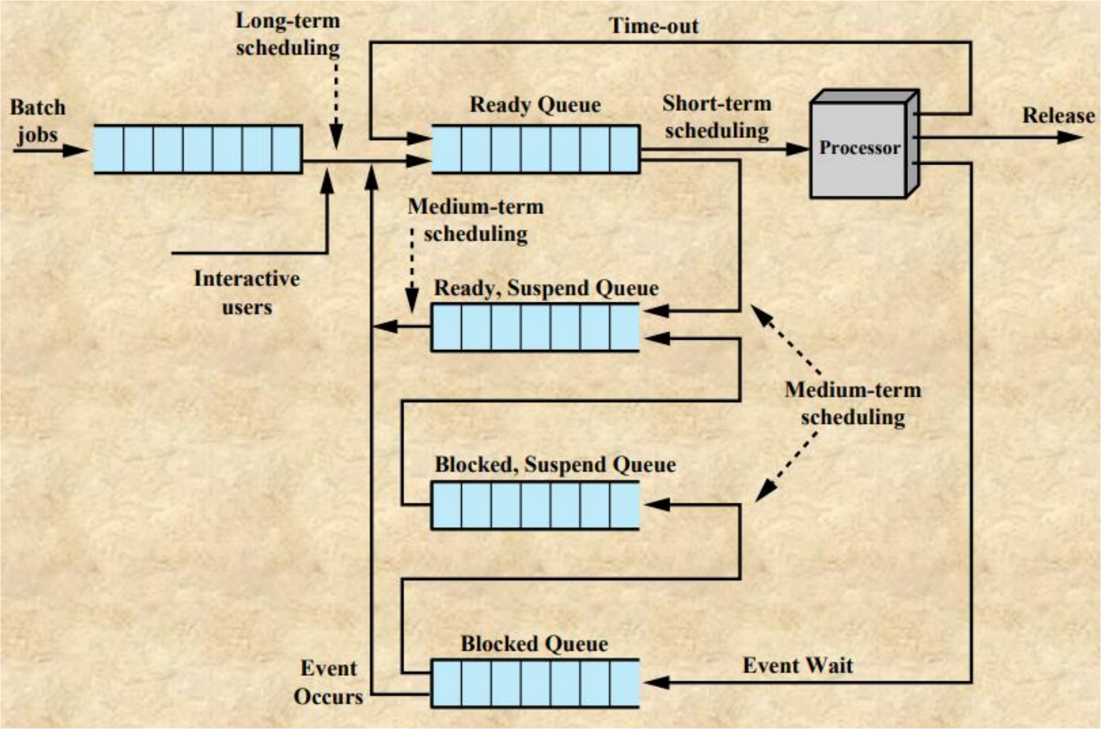
Levels of Scheduling

❑ Short-term scheduling/CPU scheduling

- Determine the execution order of the processes in the ready queue.

- Invoked when an event occurs that may lead to the blocking/termination of the current process or that may provide an opportunity to preempt a currently running process in favor of another.
  - ✓ Clock interrupts
  - ✓ I/O interrupts
  - ✓ System calls
  - ✓ Signals, etc.

- Scheduler is consuming CPU too! It executes most frequently, must be very careful about its computation overhead.



Scheduling Objectives

- ❑ Metrics—system-wide

  - Maximize processor/CPU utilization
    - ✓ percentage of time that CPU is running users' processes, to keep system as busy as possible

  - Maximize throughput
    - ✓ number of processes completed per time unit
    - ✓ number of instructions executed per time unit

  - Fairness
    - ✓ don't starve any processes—treat the all the same

Scheduling Objectives

❑ Metrics—per process/user-oriented

- Minimize waiting time in the ready queue

- Minimize real time/wall clock time
  - ✓ equal to sum of the waiting time in the ready and waiting queue, plus running time.

- Minimize response time
  - ✓ for interactive job, time from the submission of a request until the first response happens.

Scheduling Objectives

❑ Metrics

- Achieve a balance between response time and utilization

- Minimize overhead (system level)
  - ✓ Context switching
  - ✓ Scheduling complexity

- It is difficult to find the optimal solution of the scheduling since the scheduling problem is mostly an NP-hard/NP-complete problem (e.g., job shop scheduling). Instead, looking for heuristic approaches is the common way.

Process Scheduling

❑ Non-preemption V.S. Preemption

- **Non-preemption**
  - ✓ The running process continues execution until either it is terminates/blocked/yield, even though a new process is scheduled by the scheduling algorithm.
- **Preemption**
  - ✓ The running process is immediately suspended when a new process is scheduled by the scheduling algorithm.

❑ Concurrency V.S. Parallelism

- **Concurrency**
  - ✓ schedule multiple processes onto a single CPU—time multiplex manner
- **Parallelism**
  - schedule multiple processes onto multiple CPUs—spatial multiplex

Process Scheduling

Scheduling algorithm		Dispatcher
Policy: to decide who gets to run	+	Mechanism: how to do the context switch

- Decision mode
  - non-preemption
  - preemption—high priority or periodically (the process's time slice expires)

- Priority function
  - static information, e.g., CPU-intensive or I/O intensive, memory requirement, service time
  - dynamic information, e.g., relative deadline, recent CPU consumption

Process Scheduling

<u>Scheduling algorithm</u>		<u>Dispatcher</u>
Policy: to decide who gets to run	+	Mechanism: how to do the context switch

- Events affect/trigger the scheduling algorithm:

1) Current process goes from running to waiting state (e.g., wait for I/O)

2) Current process terminates, running to termination state

3) Current process goes from running to ready state (e.g., time slice is up or yield)

Process Scheduling

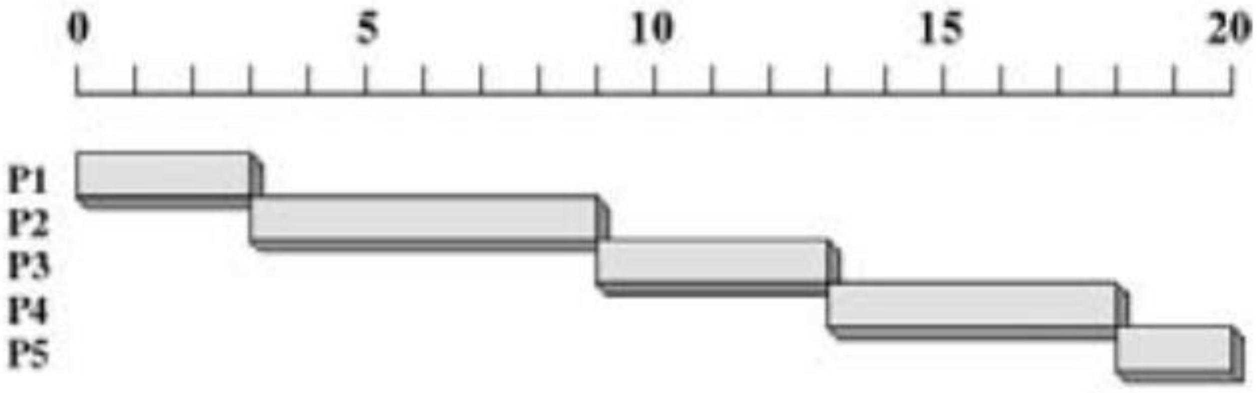
Scheduling algorithm		Dispatcher
Policy: to decide who gets to run	+	Mechanism: how to do the context switch

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  1) switching context
  2) switching between user mode and kernel mode

- Dispatch latency: time consumption of the dispatcher to stop one process and start another.

- **First-Come, First-Served (FCFS)**
  - General specification
    - ✓ Decision mode: non-preemption
    - ✓ Priority: arrival time
  - As a process become ready, it join the ready queue, scheduler always selects process from the front of the ready queue.

Process	Arrival time	Service time (Burst time)
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Waiting time for P1 = 0; P2 = 1; P3 = 5, P4= 7, P5 = 10,
Average waiting time: (0 + 1 + 5 + 7 + 10)/5 = 4.6

Scheduling Algorithms

❑ First-Come, First-Served (FCFS)

- ➢ Simple to implement; low overhead, since no priority calculation, no extra context switch.
- ➢ Average waiting time may be long and suffer from convoy effect.
  - ✓ Convoy effect: long waiting time due to the slow processes.

Case 1:

- ✓ Suppose that there are three processes already in the ready queue, and their arrival order: P1, P2, P3.
- ✓ The Gantt Chart for the schedule is

- ✓ Waiting time P1=0; P2=24; P3=27
- ✓ Average waiting time: (0+24+27)/3=17

Process	Service time
P1	24
P2	3
P3	3

Case 2:

- ✓ Suppose that there are three processes already in the ready queue, and their arrival order: P2, P3, P1.
- ✓ The Gantt Chart for the schedule is

- ✓ Waiting time P1=6; P2=0; P3=3
- ✓ Average waiting time: (6+0+3)/3=3

Scheduling Algorithms

Shortest Job Next (SJN)/Shortest Job First(SJF)

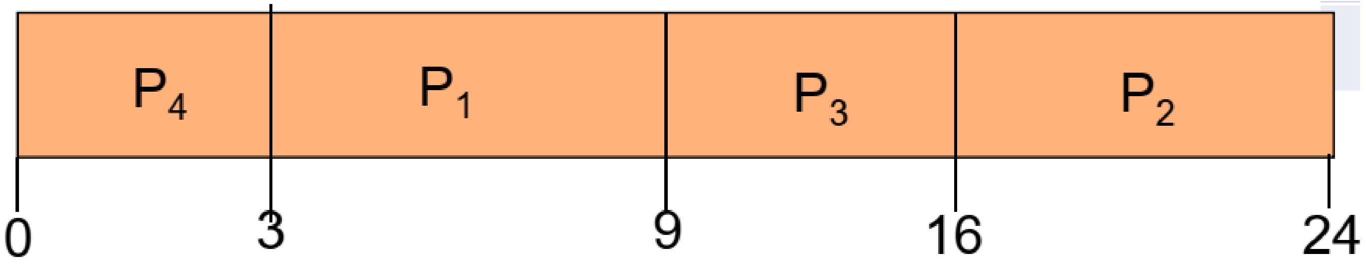
- General specification
  - Decision mode: non-preemption by default (could be implemented as preemption)
  - Priority: 1/service time—a process with a shorter service time has a higher priority
- Pros:
  - Decrease the average waiting time as compared to FCFS
- Cons:
  - Higher complexity as compared to FCFS
  - Difficult to predict service time
  - Risk to starve slow process, as long as there are fast processes around

Shortest Remaining-time First (SRF)

- General specification
  - Decision mode: preemption
  - Priority: 1/(service time-running time)
- Still needs to estimate service time and can potentially starve slow processes.
- May give less average waiting time than SJN/SJF but lead to more context switch.

Shortest Job Next (SJN)/Shortest Job First(SJF)

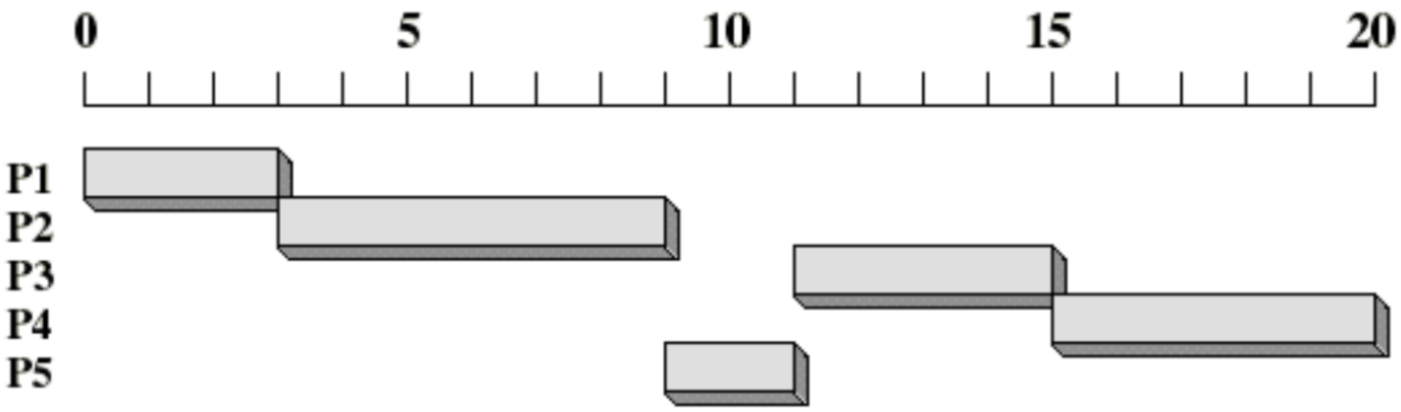
- Suppose that there are four processes already in the ready queue.
- The Gantt Chart for the schedule is:



Process	Service time
P1	6
P2	8
P3	7
P4	3

- Waiting time for P1=3 , P2=16 , P3=9 , P4=0
- Average waiting time: (3+16+9+0)/4=7
- Suppose FCFS is applied, average waiting time: (0+6+14+21)/4=10.25

Shortest Job Next (SJN)/Shortest Job First(SJF)



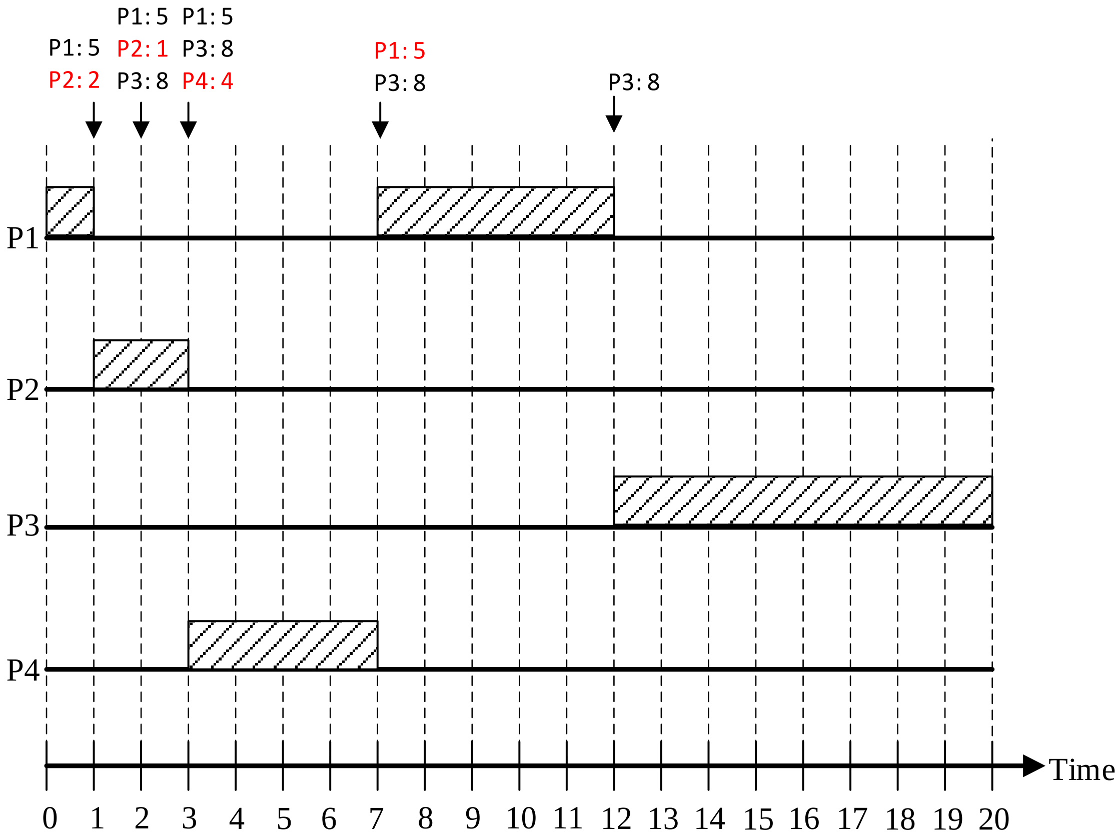
Scheduling diagram

Process	Arrival time	Service time
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2

- Waiting time for P1=0 , P2=1, P3=7 , P4=9, P5=1.
- Average waiting time: (0+1+7+9+1)/5=3.6
- Suppose FCFS is applied, average waiting time: (0+1+5+7+10)/5=4.6

❑ Shortest Remaining-time First (SRF) VS Shortest Job Next (SJN)

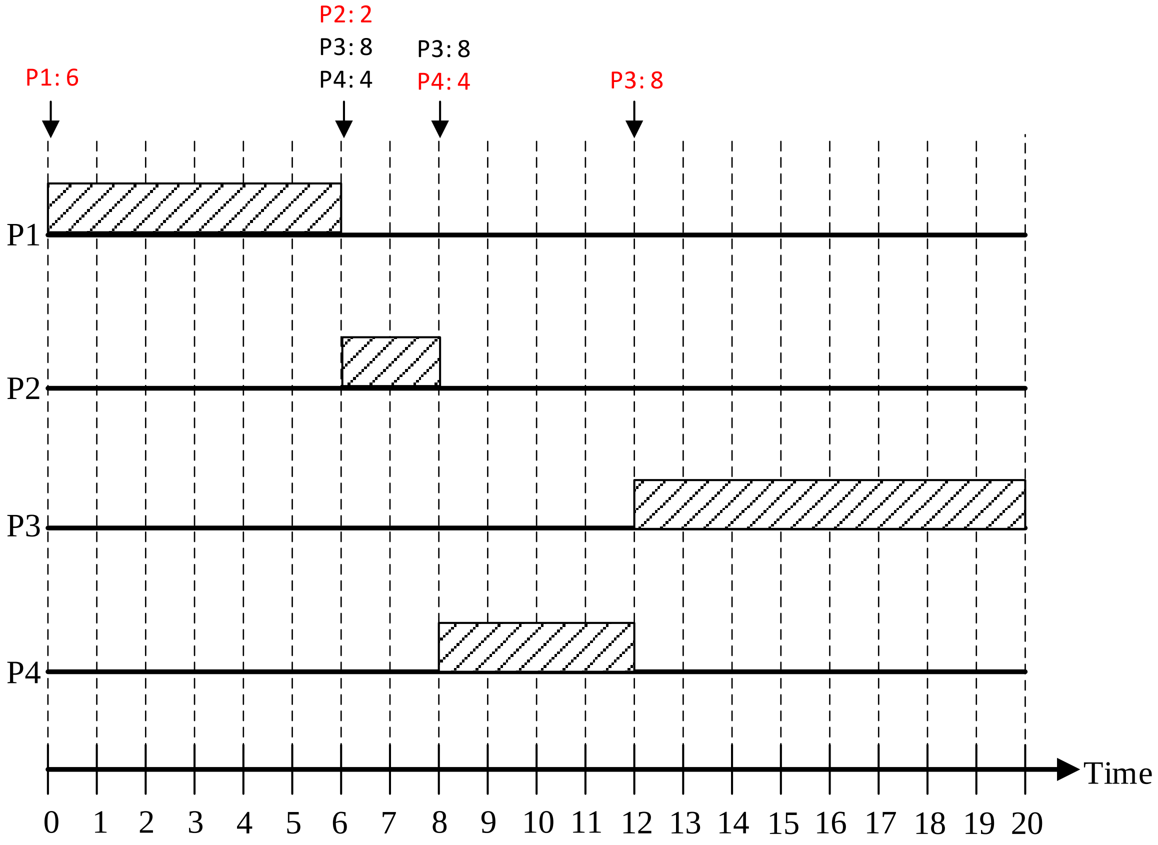
- ➢ Suppose we have four processes arriving based on the order showing in the table.
- ➢ The scheduling diagram for SRF is



Process	Arrival time	Service time
P1	0	6
P2	1	2
P3	2	8
P4	3	4

- ➢ The priorities of the processes are recalculated for each arrival.
- ➢ Waiting time for P1=6 , P2=0 , P3=10 , P4=0.
- ➢ Average waiting time: (6+0+10+0)/4=4

- Shortest Remaining-time First (SRF) VS Shortest Job Next (SJN)
  - Suppose we have four processes arriving in the order showing in the table.
  - The scheduling diagram for SJN is



Process	Arrival time	Service time
P1	0	6
P2	1	2
P3	2	8
P4	3	4

  - The priorities of the processes is static.
  - The average waiting time: (0+5+10+5)/4=5

Scheduling Algorithms

Round Robin (RR)

- General specification
  - decision: preemption, periodically with time slice
  - priority: equal
- Use preemption based on clock - time slicing (time quantum), generate interrupt at periodic intervals.
- When an interrupt occurs, the running process is placed back to the end of Ready queue, the next process is selected to run based on FCFS.
- Designed especially for interactive jobs.
- What's the right length of a time slice?
  - short time slice leads to processes move through quickly, thus having high overhead to deal with process scheduling and context switching.
  - long time slice makes RR degenerate into FCFS.
  - should be slightly greater than average service time.

Scheduling Algorithms

❑ Round Robin (RR)

- Suppose that there are three processes in the ready queue, and their arrival order: P1, P2, P3. The time quantum length is 4.

- The Gantt Chart for the schedule is:

P_1	P_2	P_3	P_1	P_1	P_1	P_1	P_1

0 4 7 10 14 18 22 26 30

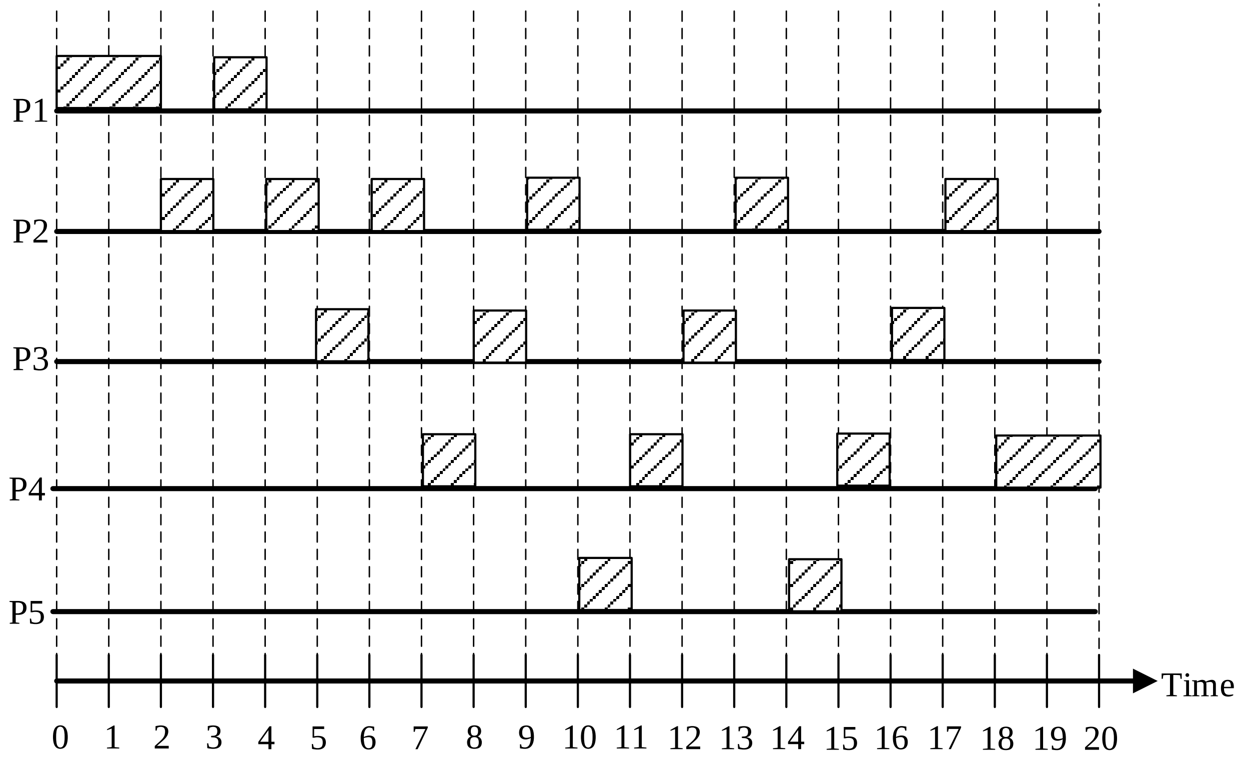
Process	Service time
P1	24
P2	3
P3	3

- Wait time for P1=6; P2=4; P3=7

- Average waiting time: (6+4+7)/3=5.67.
  - ✓ Recall that the average waiting time for FCFS is 17.
  - ✓ However, average waiting time of RR < average waiting time of FCFS is not always true.

- The number of context switching in RR >= The number of context switching in FCFS

Round Robin (RR)- time quantum=1



Process	Arrival time	Service time (Burst time)
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2

Waiting time for P1 = 1; P2 = 10; P3 = 9, P4= 9, P5 = 5,
Average waiting time: (1+ 10 + 9 + 9 + 5)/5 = 6.8
Recall that the average waiting time for applying FCFS is 4.6 in Slide 12, but waiting_time (RR) > waiting_time (FCFS) is not always true.