Chapter 4
Scheduling

-- By R.G.B

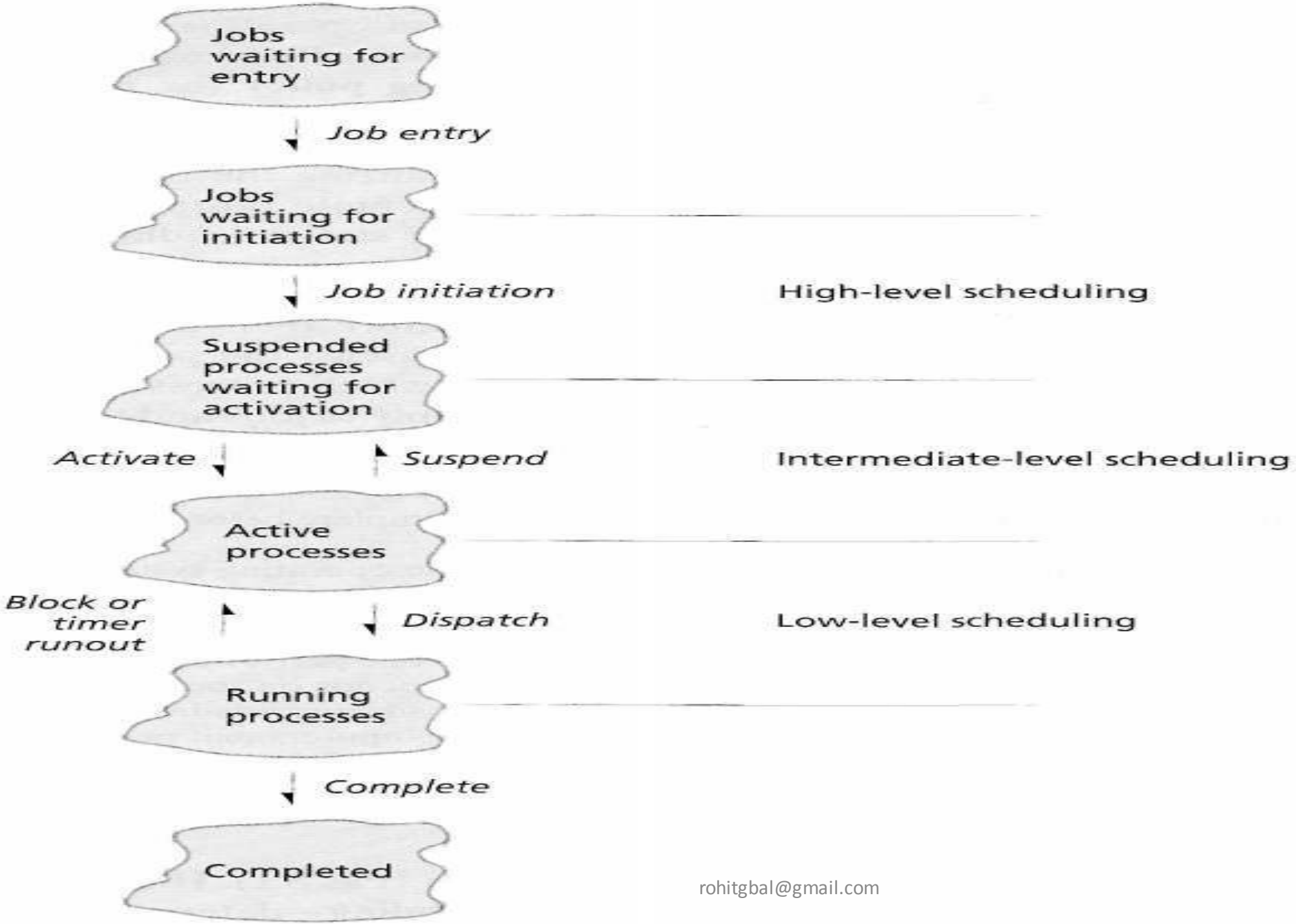
Scheduling

- Frequently multiple process competes for the CPU at the same time
- Two or more process are simultaneously in the ready state
- Choice has to be made which process is to run next
- Part of the OS is called Scheduler
- Algorithm is called scheduling algorithm
- Process execution consists of cycles of CPU execution and I/O wait. Processes alternate between these two state

Scheduling Levels

- Three levels of scheduling

1. High Level Scheduling
2. Intermediate Level Scheduling
3. Low Level Scheduling


Jobs waiting for entry
Job entry
Jobs waiting for initiation
Job initiation
High-level scheduling
Suspended processes waiting for activation
Activate
Suspend
Intermediate-level scheduling
Active processes
Block or timer runout
Dispatch
Low-level scheduling
Running processes
Complete
Completed

High Level Scheduling

- Job Scheduling/Long term Scheduling/Admission
- Determines which jobs the system allows to compete actively for system resources
- Too much Process admitted→ Poor performance

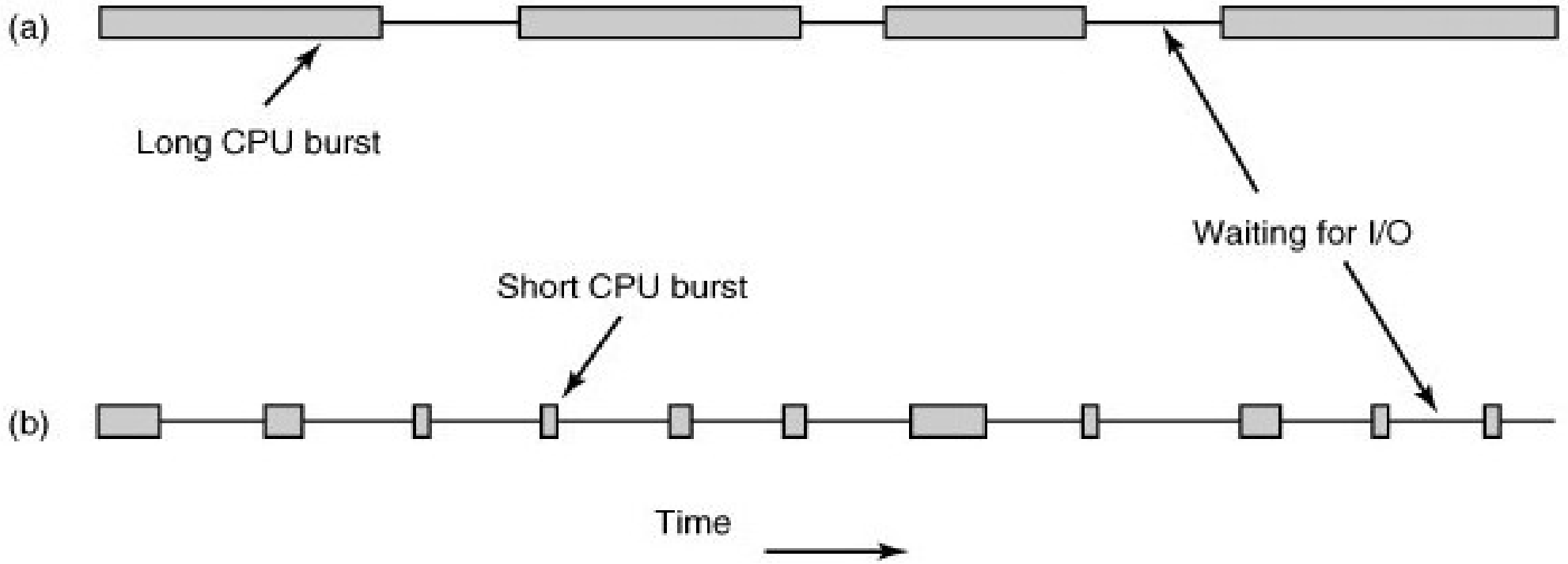
Intermediate Level Scheduling

- Determines which processes shall be allowed to compete for processors
- Responds to short-term fluctuations in system
- Temporarily suspends and resumes processes to achieve smooth system operation
- Intermediate-level scheduler acts as a buffer between the admission of jobs to the system and the assignment of processors

Low Level Scheduling

- Determines which active process the system will assign to a processor
- The short-time scheduler will be executing frequently
- Mostly at least once every 10 milliseconds
- It has to be very fast in order to achieve a better processor utilization
- It often assign

Process Behaviour



- Bursts of CPU usage alternate with periods of waiting for I/O. (a) A CPU-bound process. (b) AnI/O bound process

Process Behaviour

- **CPU Bound(or compute bound):** Some process such as the one shown fig a. above spend most of their time computing
- These processes tend to have long CPU burst and thus infrequent I/O waits
- **Example: Matrix multiplication**
- **I/O Bound:** Some process such as the one shown in fig. b above spend most of their time waiting for I/O
- They have short CPU bursts and thus frequent I/O waits
- **Example: Firefox**

Scheduling Criteria

- **CPU utilization**
- **Throughput**
- **Turnaround Time**
- **Waiting time**
- **Response time**

Scheduling Criteria

- **CPU utilization**: Keep the CPU as busy as possible.(range 0 to 100%)
- Goal of Scheduler Design →Less Idle Time
- In a real system it should range from 40 – 90 % for lightly and heavily loaded system.
- **Throughput:** It is the measure of work in terms of number of process completed per unit time.
- For long process this rate may be 1 process per hour, for short transaction, throughput may be 10 process per second.

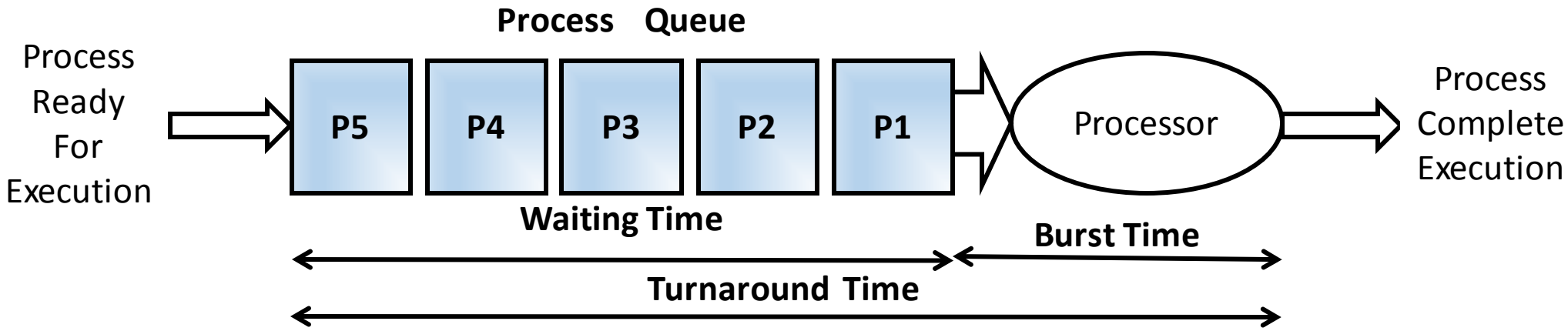
Scheduling Criteria

- **Turnaround Time:** It is the sum of time periods spent in waiting to get into memory, waiting in ready queue, execution on the CPU and doing I/O.
- The interval form the time of submission of a process to the time of completion is the turnaround time.
- **Turnaround time** = Time of completion of job - Time of submission of job. (waiting time + service time or burst time)

Scheduling Criteria

- **Waiting time:** its the sum of periods waiting in the ready queue.
- **Response time:** The amount of time it takes to start responding, not the time taken to output that response
- **Burst Time:** Actual time required to complete execution of particular task

Scheduling Criteria



Scheduling Types

- Scheduling algorithms can be divided into two categories with respect to how they deal with clock interrupts

1. **Pre-emptive Scheduling**
2. **Non pre-emptive Scheduling**

Pre-emptive Scheduling

- Picks a process and lets it run for a maximum of some fixed time
- This time interval is called **Quantum time/Time slice**
- If it is still running at the end of the time interval, it is suspended and the scheduler picks another process to run (if one is available).
- Clock interrupt occur at the end of time interval to give control of the CPU back to the scheduler

Non Pre-emptive Scheduling(*Greedy one*)

- Picks a process to run and then just lets it run until it blocks or until it voluntarily releases the CPU
- Even it runs for hours, it will not be forcibly suspended
- No scheduling decisions are made during clock interrupts

Scheduling Techniques

1. First In First Out Scheduling
2. Shortest Job First Scheduling
3. Priority Scheduling
4. Round Robin Scheduling
5. Deadline Scheduling
6. Shortest Remaining Time Scheduling
7. Highest Response Ratio Next Scheduling
8. Multilevel Feedback Queues

First In First Out (FIFO)Scheduling

- First Come First Serve(FCFS)
- Non Pre-emptive Type
- Processes are assigned the CPU in the order they request it
- When the first job enters the system from the outside it is started immediately and allowed to run as long as it wants
- Other jobs come in, they are put onto the end of the queue
- When the running process blocks, the first process on the queue is run next
- When a blocked process becomes ready, like a newly arrived job, it is put on the end of the queue.

FIFO Advantages & Disadvantages

- **Advantages**
  - Easy to understand and program
  - Equally fair
  - Suitable specially for Batch Operating system
- **Disadvantages**
  - FCFS is not suitable for time-sharing systems where it is important that each user should get the CPU for an equal amount of arrival time.

Scheduling

- Consider the following set of processes having their burst time mentioned in milliseconds. CPU burst time indicates that for how much time, the process needs the CPU.

Process	Burst Time
P1	13
P2	14
P3	12
P4	41

Calculate the average waiting time and turn around time if the processes arrive in the order of

a)P1,P3,P2,P4

b)P4,P3,P1,P2

Shortest Job First Scheduling

- Non pre-emptive
- Scheduler selects process with the smallest estimated run-time
- SPF reduces average waiting time over FIFO
- When several equally important jobs are sitting in the i/p queue waiting to be started, the scheduler picks the shortest jobs first
- Best approach to minimize waiting time.
- Impossible to implement
- Processer should know in advance how much time process will take to complete

Priority Based Scheduling

- Each process is assigned a priority Process with highest priority is to be executed first and so on
- Processes with same priority are executed on FCFS basis
- Priority can be decided based on memory requirements, time requirements or any other resource requirement

Priority Based Scheduling

Assigning priority:

- To prevent high priority process from running indefinitely the scheduler may decrease the priority of the currently running process at each clock interrupt
- If this causes its priority to drop below that of the next highest process, a process switch occurs
- Each process may be assigned a maximum time quantum that is allowed to run
- When this quantum is used up, the next highest priority process is given a chance to run

Priority Based Scheduling

- **Types of Priority**

1. **Static priorities**

- Remain fixed, so static-priority-based mechanisms are relatively easy to implement
- Not responsive to changes in environment

2. **Dynamic priority**

- Mechanisms are responsive to change
- High overhead

Round Robin Scheduling

- Pre-emptive Scheduling
- Each process is provided a fix time(Time Interval) to execute called quantum/Time Slice
- Once a process is executed for given time period
- Process is pre-empted and other process executes for given time period
- Context switching is used to save states of pre-empted processes
- The system then places the pre-empted process at the back of the ready queue

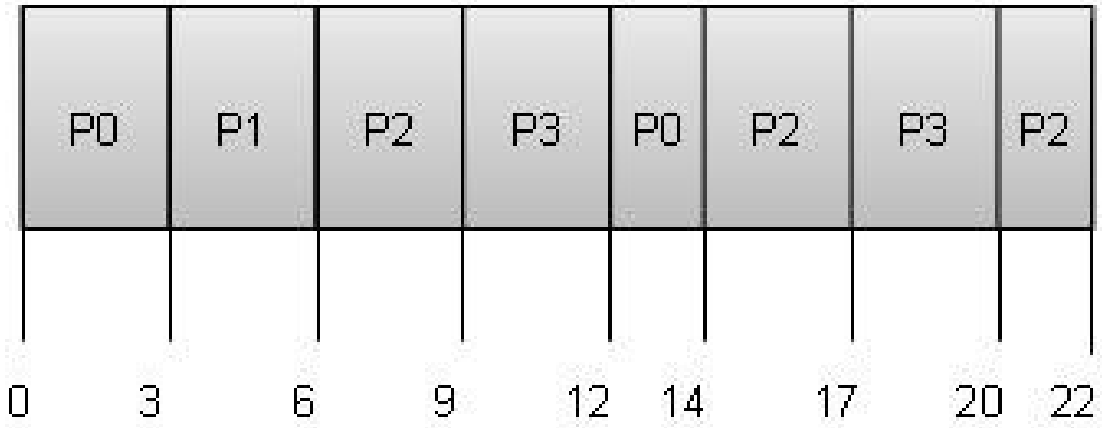
Round Robin Scheduling

- Round-robin is effective for interactive environments in which the system needs to guarantee reasonable response times
- The system can minimize pre-emption overhead through efficient context-switching mechanisms and by keeping waiting processes in main memory

Round Robin Scheduling

Consider Process arrives in order P0,P1,P2,P3 and burst times 5,3,8,6 and initially all process arrives at time 0

Quantum = 3



Process	Wait Time : Service Time - Arrival Time
P0	(0-0) + (12-3) = 9
P1	(3-0) = 2
P2	(6-0) + (14-9) + (20-17) = 14
P3	(9-0) + (17-12) = 14

Deadline Scheduling

- Certain processes are scheduled to be completed by a specific time or deadline
- These processes may have high value if delivered on time and little or no value otherwise
- the priority of the process or thread may need to be increased as its completion deadline approaches

Deadline Scheduling

Deadline scheduling is complex

1. User must supply the precise resource requirements in advance to ensure that the process is completed by its deadline
2. The system should execute the deadline process without severely degrading service to other users
3. New processes may arrive, making unpredictable demands
4. Many deadline processes are active at once, scheduling could become extremely complex

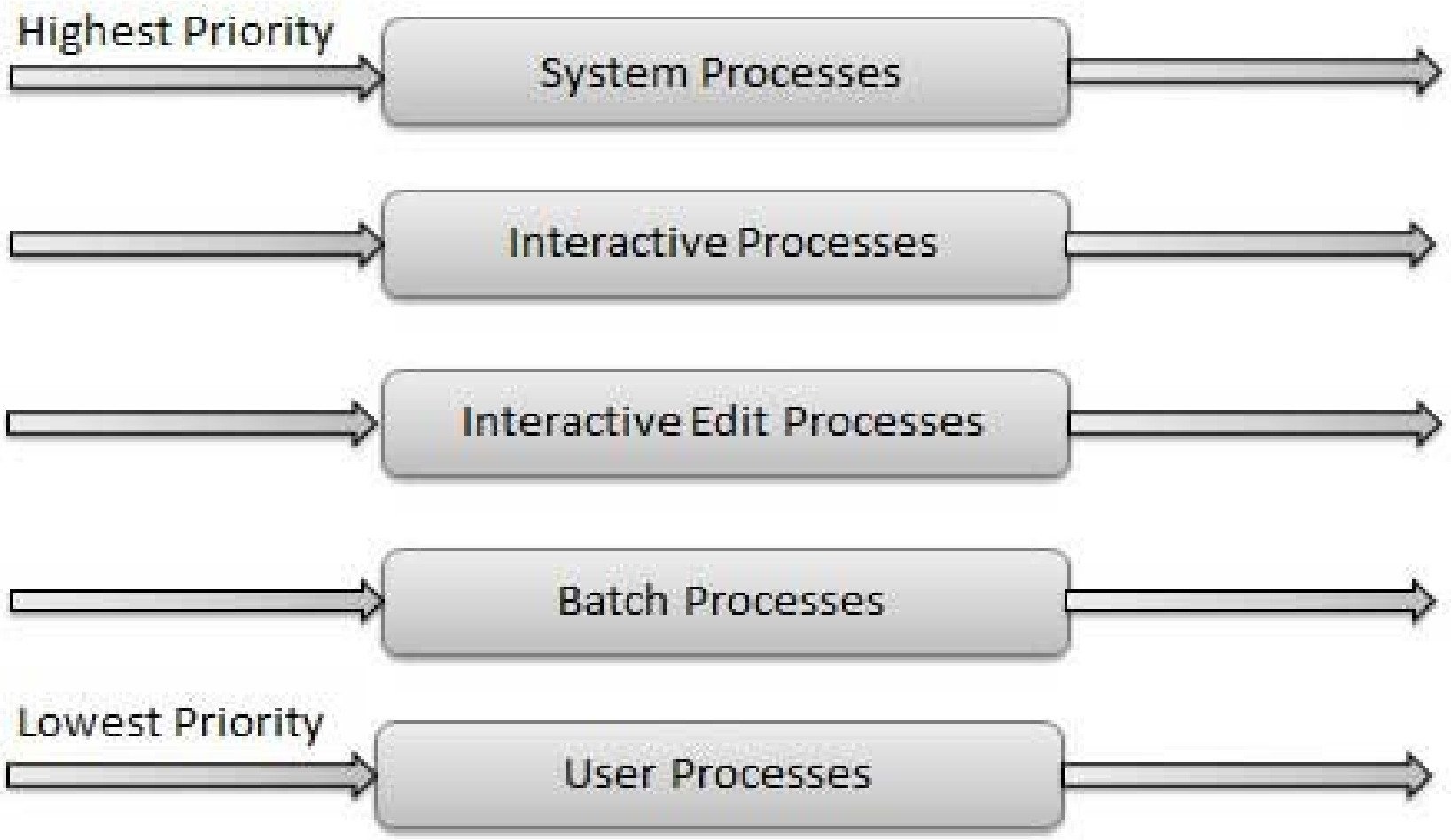
Shortest Remaining Time Scheduling

- **Shortest-remaining-time (SRT)** scheduling is the pre-emptive counterpart of SPF that attempts to increase throughput by servicing small arriving processes
- SRT was effective for job-processing systems that received a stream of incoming jobs
- The scheduler selects the process with the smallest estimated run-time-to-completion

Highest-Response-Ratio-Next (HRRN)

- Policy that corrects some of the weaknesses in SPF, particularly the excessive bias against longer processes and the excessive favouritism toward short processes
- HRRN is a non pre-emptive scheduling discipline in which each process's priority is a function not only of its **service time but also of its time spent waiting for service**
- HRRN calculates dynamic priorities according to the formula
- **Priority=(service time + waiting time)/service time**
- Longer processes that have been waiting will also be given favourable treatment

Multi Queue Scheduling



Multi Queue Scheduling

- Scheduling processes are classified into different groups
- Multiple queues are maintained for processes
- Each queue can have its own scheduling algorithms
- Priorities are assigned to each queue

Guaranteed Scheduling

- Make real promises to the users about performance
- N users logged in while you are working, you will receive about 1 /n of the CPU power.
- Single-user system with n processes running, all things being equal, each one should get 1 /n of the CPU cycles
- System must keep track of how much CPU each process has had since its creation
- CPU Time entitled= (Time Since Creation)/n

Lottery Scheduling

- Lottery Scheduling is a probabilistic scheduling algorithm for processes in an operating system
- Processes are each assigned some number of lottery tickets for various system resources and the scheduler draws a random ticket to select the next process
- The distribution of tickets need not be uniform; granting a process more tickets provides it a relative higher chance of selection
- This technique can be used to approximate other scheduling algorithms, such as Shortest job next and Fair share scheduling.
- Lottery scheduling solves the problem of starvation

Two-Level Scheduling

- Performs process scheduling that involves swapped out processes
- Two-level scheduling is needed when memory is too small to hold all the ready processes
- Consider this problem: A system contains 50 running processes all with equal priority. However, the system's memory can only hold 10 processes in memory simultaneously. Therefore, there will always be 40 processes swapped out written on virtual memory on the hard disk
- **Uses two different schedulers**
- **One lower-level scheduler** which can only select among those processes in memory to run.
- **Other scheduler is the higher-level scheduler** whose only concern is to swap in and swap out processes from memory.

Scheduling in Real Time System

- Time is crucial and plays an essential role
- Cd→ music
- Auto pilot in Aircraft
- Robot control in automated factory.
- Patient monitoring in Hospital
- Two types:
  1. **Hard Real Time system:** Absolute deadline that must be met.
  2. **Soft Real Time system:** Missing an occasional deadline is undesirable but tolerable