COMPS267F Chapter 4

Process Scheduling

Dr. Andrew Kwok-Fai LUI

Aim of the Chapter

- Discuss how to manage the CPU
  - Perhaps the most precious resource on computer systems
  - Study CPU scheduler
- Select a process to use an idle CPU
  - There are different methods or algorithms
  - Each results in different CPU usage patterns
  - Each leads to different performance characteristics

Example: Customer Service in a Bank

Example: Customer Service in a Bank

Example 1: Customer Service in a Bank

In a particular branch of Open Bank, there are ten customers queuing for teller service. The Open Bank is suffering from financial tsunami and can afford to make one teller counter available. Consider the following methods of selecting a customer from the queue and evaluate the impact on the business of the bank.

- The first customer arrived will be served first.
- The customer with the higher account balance will be served first.
- The customer wanting one simple transaction (such as deposit cheques, buying foreign currencies) will be served first.

Example: Customer Service in a Bank

- First-come-first-served method is most neutral to all customers.
  - Random arrival
  - No other factors
  - The bank may be perceived as being neutral and welcome everybody.

Example: Customer Service in a Bank

Example: Customer Service in a Bank

- The rich-over-poor method gives priority to the rich
  - The rich encouraged to do more business with the bank
  - Expect a higher turnover in the business
  - Antagonize the poor

Example: Customer Service in a Bank

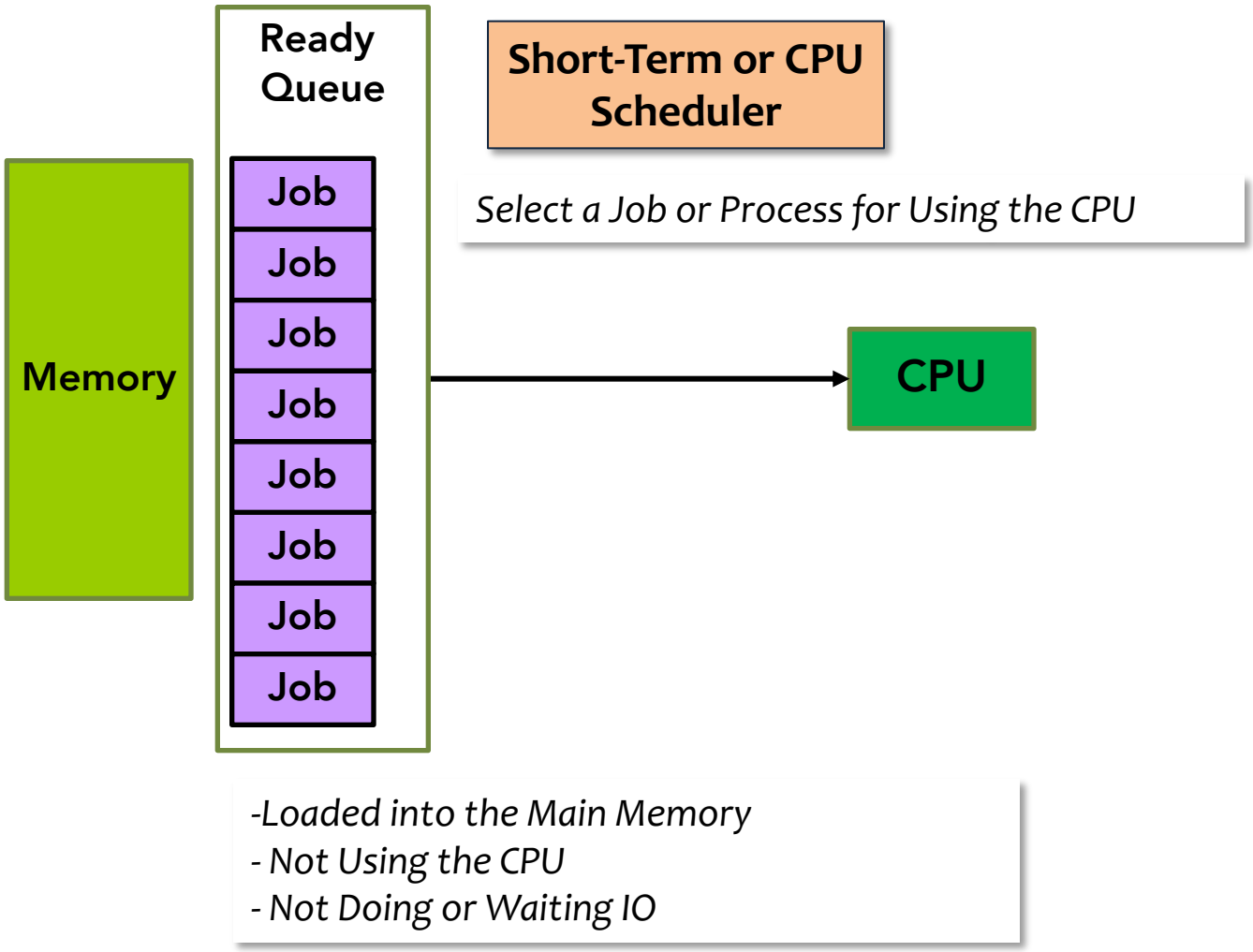
- The short-over-long method gives priority to the short transactions.
  - The long transactions are disadvantaged
  - These customers may never get served if there are new customers keep coming in
  - Keep majority of customers happy
    - A bank can satisfy more customers doing short transactions

performance objectives

CPU Scheduler

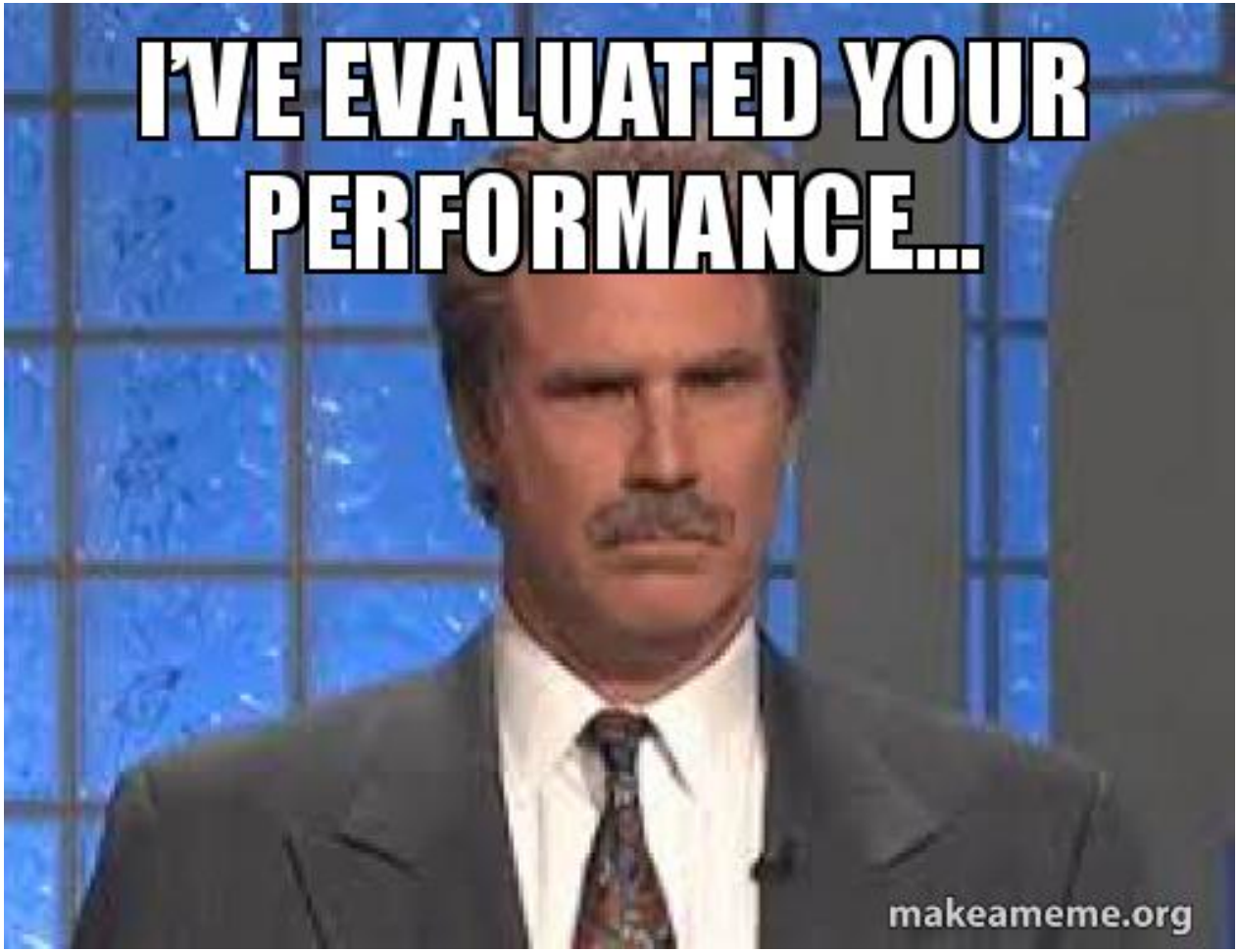
- The component responsible for selecting a process from the queue of ready processes
  - These processes are in the Ready state
  - These processes are in the Ready Queue
- Characteristics of processes in the Ready Queue
  - Loaded in the main memory
  - Do not have control of the CPU right now
  - Not waiting or doing any I/O

CPU Scheduler

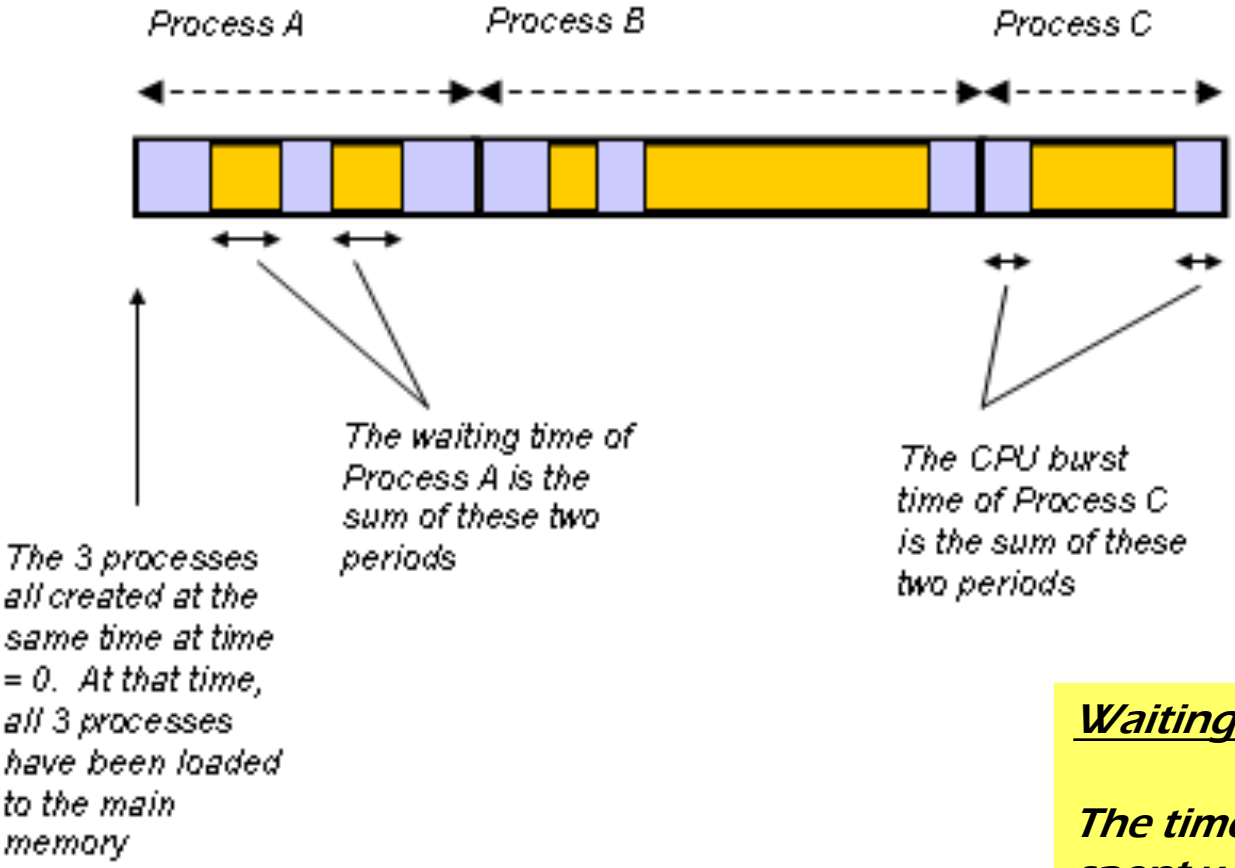


Performance Characteristics

- No best method in CPU scheduling
  - Choose according to the desired performance outcome
  - Common performance measurements (metrics) used :
    - Process resource consumption
    - Resource Utilization
    - Multi-user support
    - No starvation
    - Through-put
    - Turn-around time
    - Waiting time
    - Response time


I'VE EVALUATED YOUR PERFORMANCE...
makeameme.org

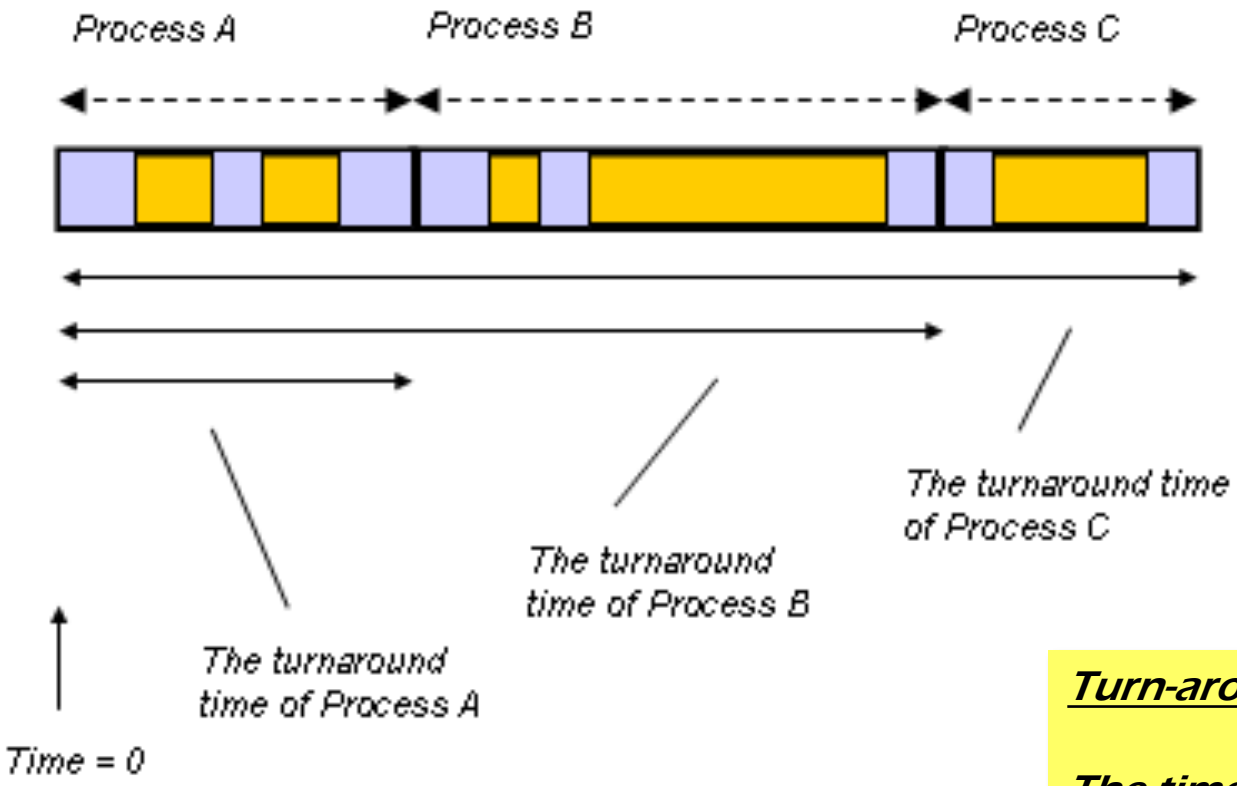
Performance Metrics



Waiting Time

The time a process spent waiting for IO operations and Ready queue

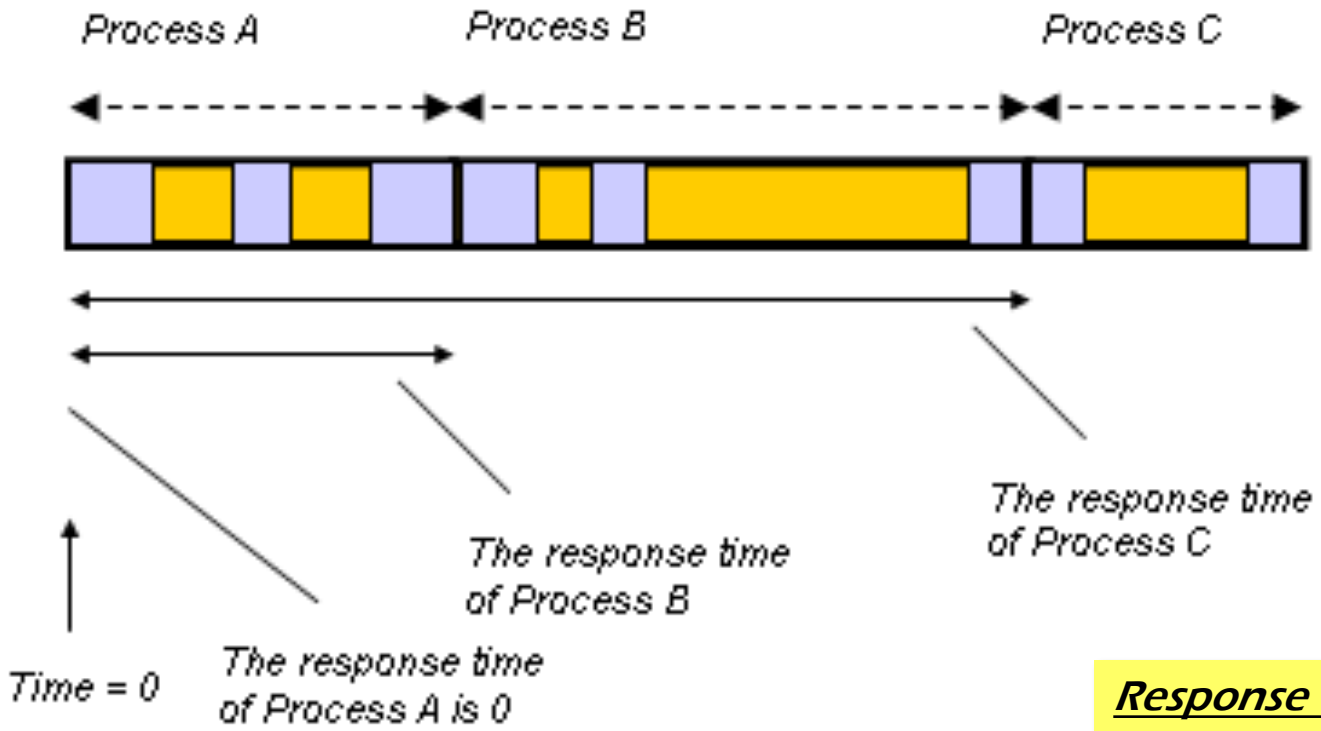
Performance Metrics



Turn-around Time

The time taken for the process to complete execution

Performance Metrics

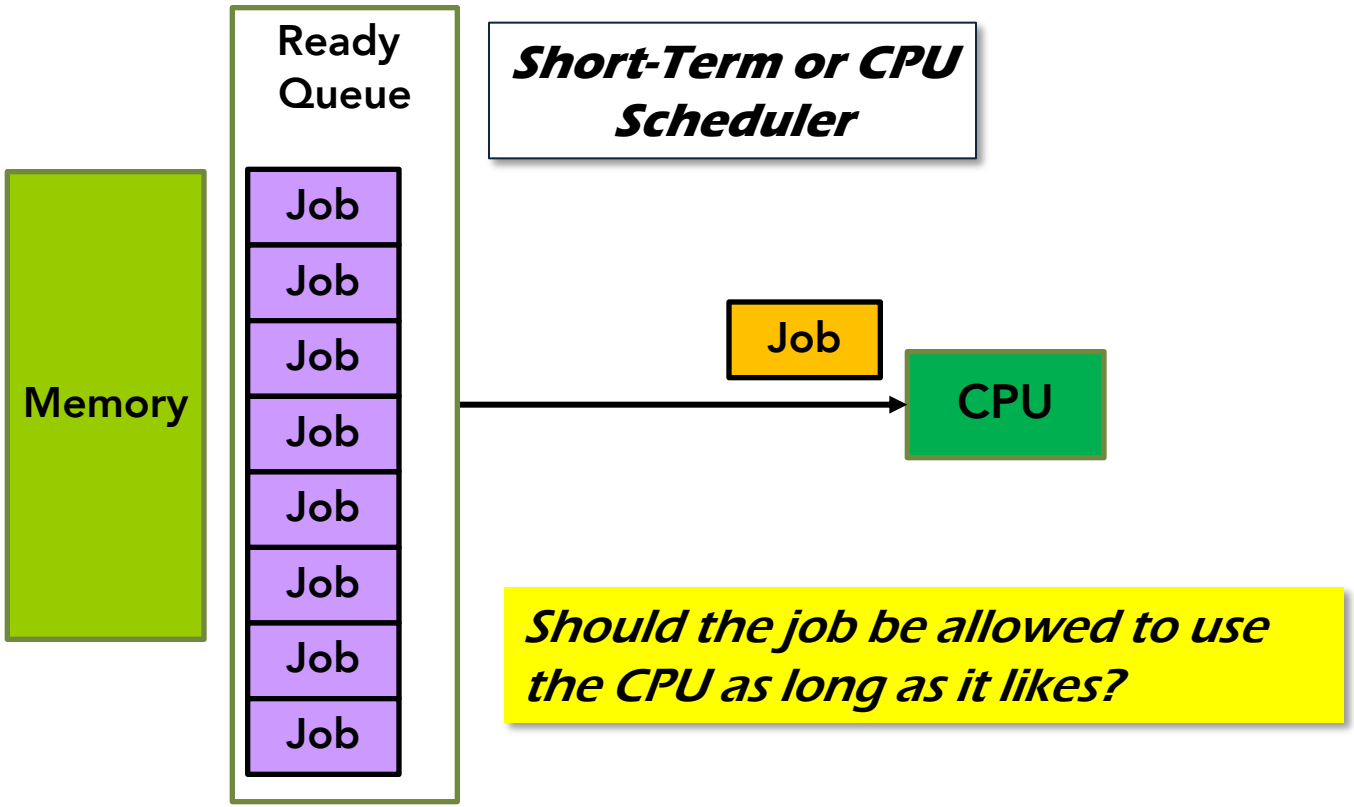


Response Time

The time a process spent waiting for the first execution by CPU

concepts of cpu scheduling

Pre-Emption


Ready Queue
Short-Term or CPU Scheduler
Memory
Job
Job
Job
Job
Job
Job
Job
Job
Job
CPU
Should the job be allowed to use the CPU as long as it likes?

Pre-Emption

- Stopping the process while it is using the CPU
  - No pre-emption means allowing the process to use as long as it likes (possibly until termination)
- Pre-emptive scheduling
  - Process may transit from Running back to Ready state
- Non Pre-emptive scheduling
  - Process will complete the execution and move to Terminate
  - Waiting for IO will move to Waiting
  - Never transit from Running to Ready


"SIR, AN AMERICAN PAPER, THE ONION, REPORTS OBAMA IS PLANNING AN ATTACK"
"TELL THEM WE ARE PREPARING A PREEMPTIVE STRIKE"
quickmeme.com

Pre-emptive Scheduling

- Conditions of pre-emption
  - A period of time has lapsed
    - The process not yet terminated will move back to Ready Queue
  - Priority has become lowered
    - Another process with higher priority can pre-empt existing running process
    - Remember new processes are being created in a running computer

Pre-emptive Scheduling in Banking Case

Concepts	*Bank Examples*
Non pre-emptive scheduling	Customers are allowed to stay at the teller counter until the transaction is completed or the customers cannot continue the transaction any further because of lack of documents.
Pre-emptive scheduling	Customers are asked to leave the teller if a time quota is up, or if another customer of a higher priority has arrived.

Pre-emptive Scheduling

Example 2: Pre-emptive Customer Service

In the same branch of Open Bank, the manager decides to impose a rule that each customer can use the teller service for at most 5 minutes. If the transaction cannot complete at the end of the period, then the customer has to leave the teller and join the end of the queue again. Consider the impact of this new rule.

Answer:

This is a form of pre-emptive scheduling. Customers wanting to do short transactions will be happier because they would not be delayed by another customer with an exceedingly long transaction.

Characteristics of Pre-emptive Scheduling

- Prevents a process to control the CPU exceedingly
  - Long processes can be pre-empted
  - Short processes can complete execution quicker
    - Better services to shorter processes
- The scheduler has more flexibility
  - Scheduling methods can move processes from Running to Ready

Characteristics of Pre-emptive Scheduling

- Increases the overhead of process execution
  - More context switching
  - Reduces CPU utilization
- Causes problems of data inconsistency when a process is updating a set of data and is pre-empted.
  - Disastrous if system data is involved
  - Some OS disallow a process to be pre-empted until a system call is completed

Scheduling Priority

- A systematic and unified means to select process
  - Highest priority is the next process to run
- Common factors to determine priority
  - Arrival or creation time
  - Length of CPU burst time (if known)
  - Time waiting in the queue
  - Priority can change with time

CPU Scheduling Algorithms

- A scheduling algorithm is a method of selecting process.
  - Specific method of assigning priorities to processes
  - Supported by data structures (e.g. ready queue)
  - CPU usage pattern and other system characteristics

Performance Requirements

Business Objectives (e.g. Deter Poor Customers?)

↓

Bank Manager

↓

Set up Metrics for Performance Measurement

↓

Select Scheduling Algorithm

?

Desired System Performance (e.g. Banking Operation?)

↓

Computer System Designer

↓

Set up Metrics for Performance Measurement

↓

Select Scheduling Algorithm

Common Metrics

- System performance requirements are expressed using these metrics or measurements.
  - CPU utilization
  - Turn-around time
  - Waiting time
  - Response time
  - No Starvation
  - Throughput

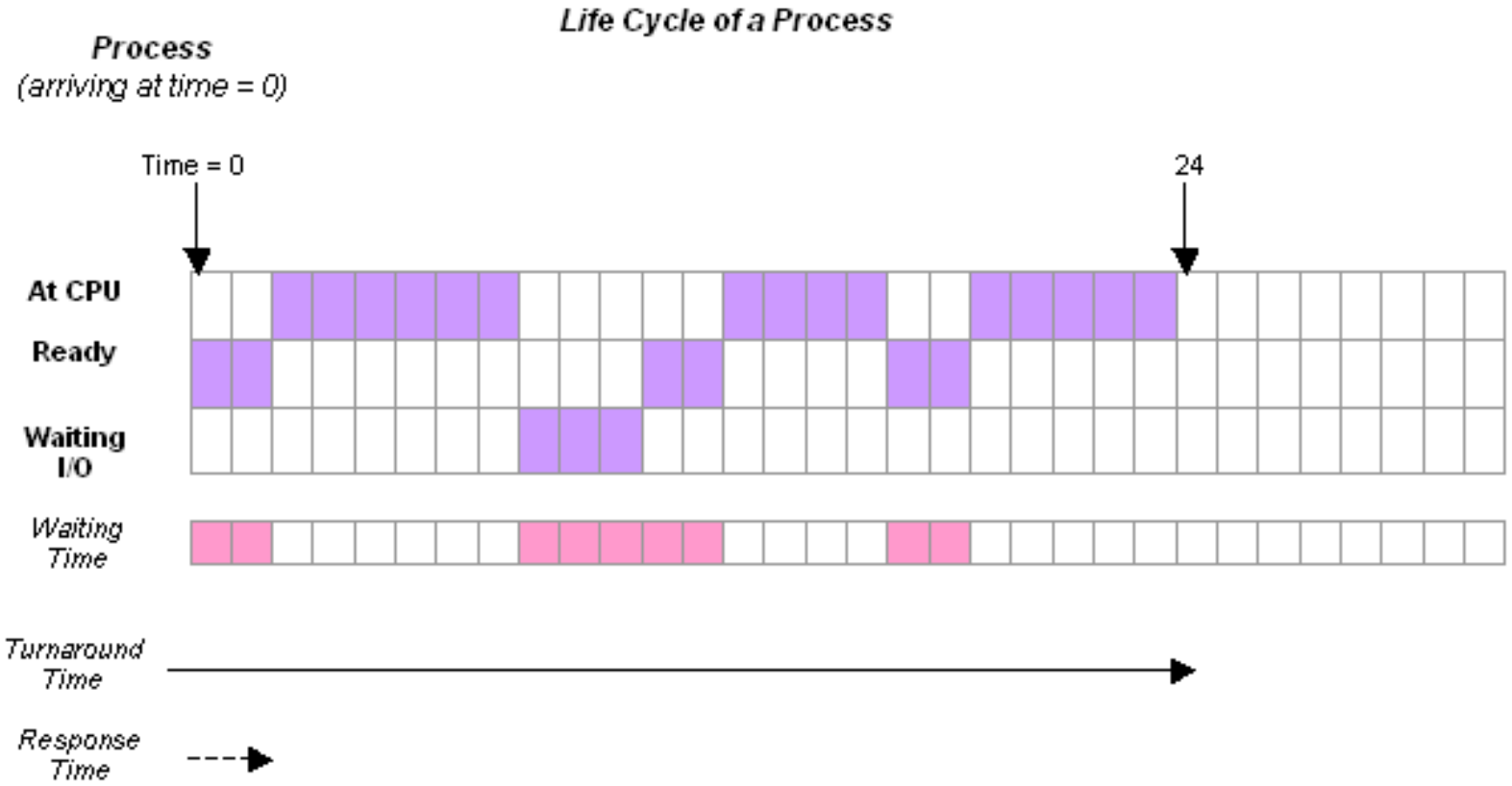
Common Metrics

Objectives	*Bank Examples*
CPU Utilization	The proportion of time that the teller is busy
Turn-around time	The time between a customer joining the queue and the completion of transaction.
Waiting time	The time spending in the queue
Response time	The time between a customer joining the queue and start being served by the teller
No Starvation	All customers will eventually get the transaction finished
Throughput	The number of customers that can be served by the teller per time unit.

Evaluation of Scheduling Algorithms

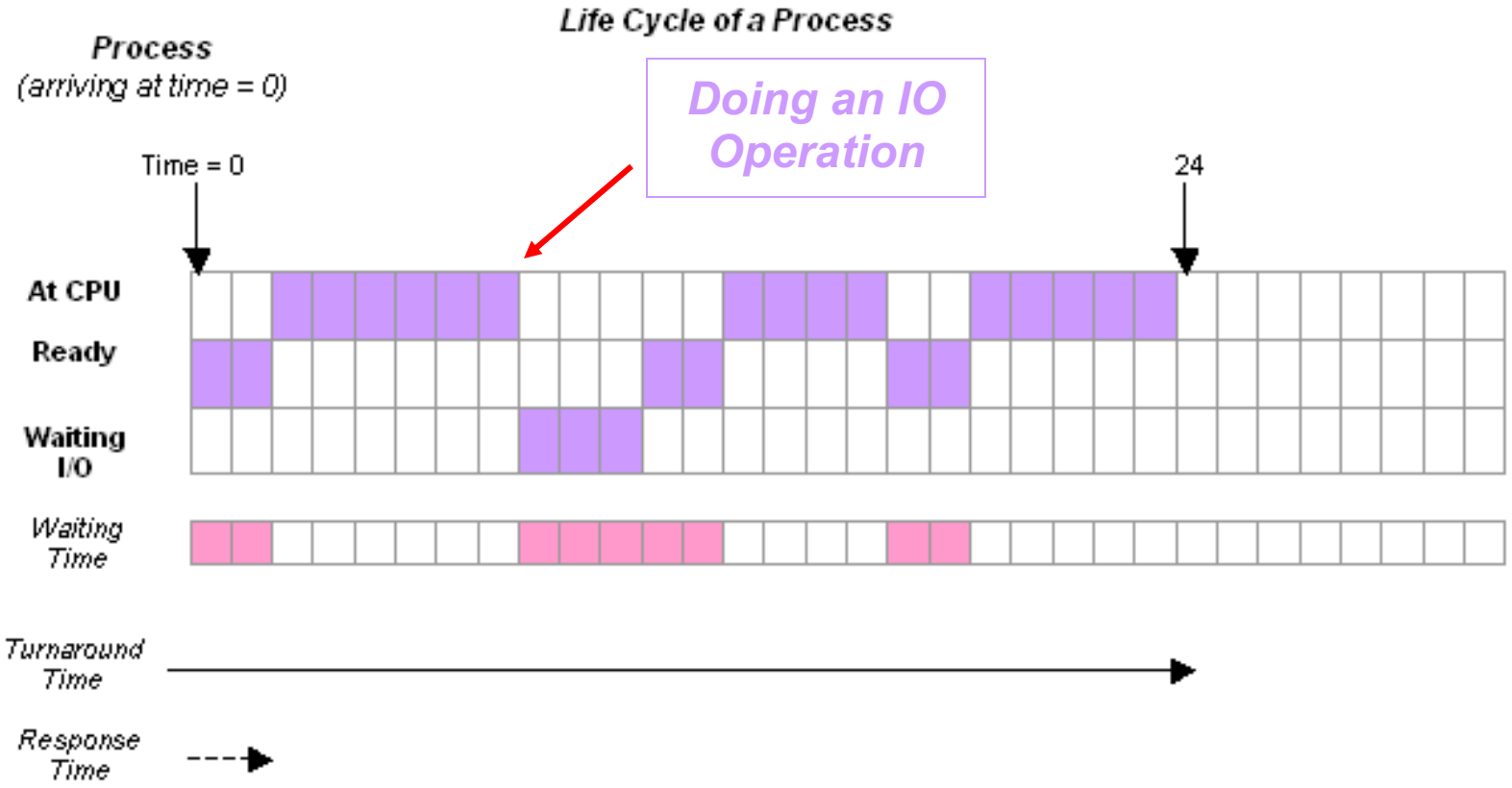
- Evaluation based on a set of processes of mixed characteristics
  - All Long processes
  - All Short processes
  - Mixed Long and Short processes
  - Different orders of Long and Short processes
- Three main metrics in evaluation of algorithms
  - Turnaround Time: From Birth to Death
  - Waiting Time: Waiting instead of doing work
  - Response Time: From Birth to the time first doing work

Evaluation of Scheduling Algorithms



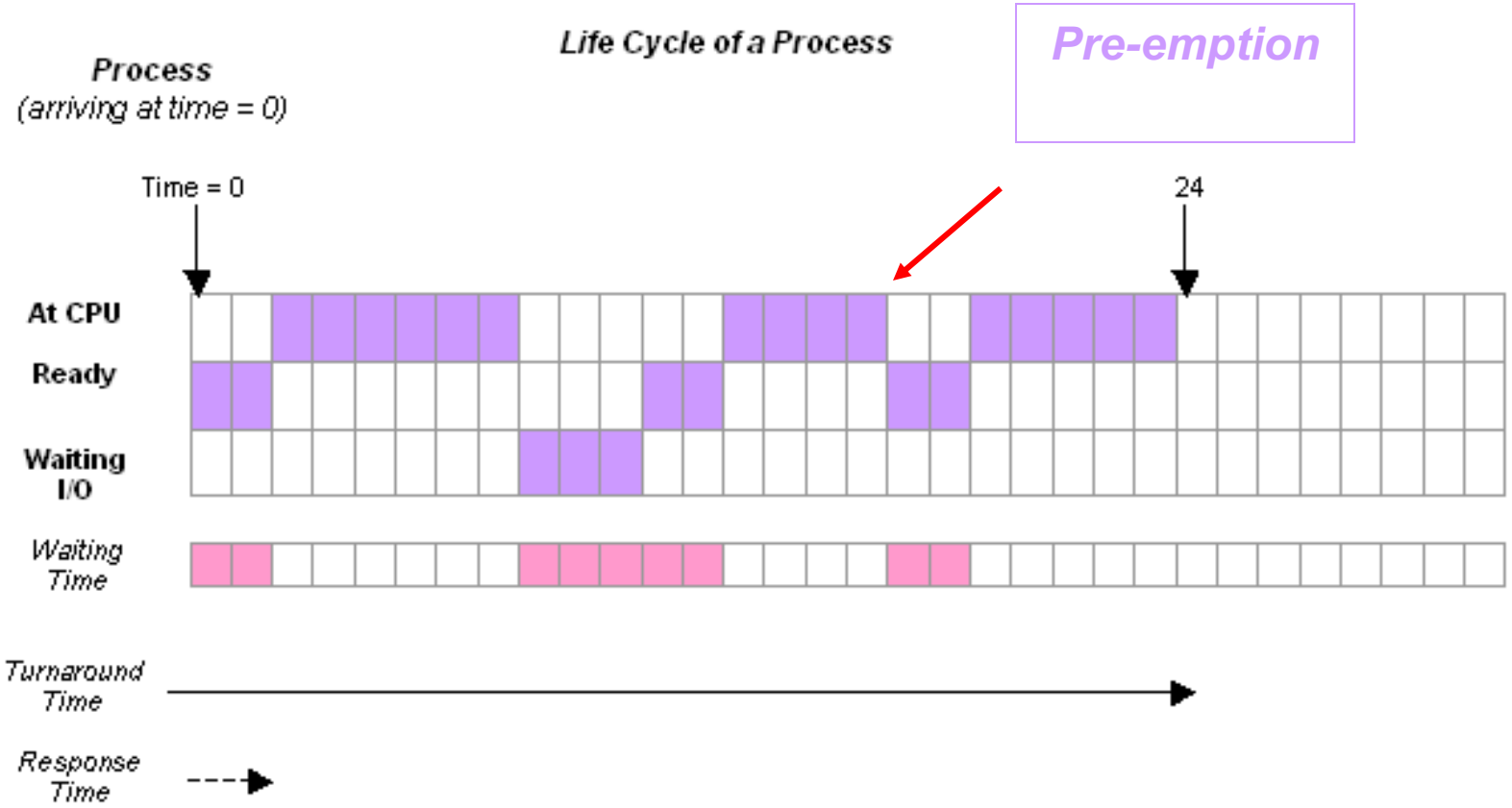
CPU Burst Time (ms)	15
Waiting Time	9
Turnaround Time	24
Response Time	2

Evaluation of Scheduling Algorithms



CPU Burst Time (ms)	15
Waiting Time	9
Turnaround Time	24
Response Time	2

Evaluation of Scheduling Algorithms



CPU Burst Time (ms)	15
Waiting Time	9
Turnaround Time	24
Response Time	2

Evaluation of Scheduling Algorithms

- Relations of the factors
  - Turnaround time = CPU burst time + Waiting time
    - Waiting time should be worked out from Turnaround time and CPU burst time
  - In non pre-emptive scheduling, Response time = Waiting time

cpu scheduling algorithms

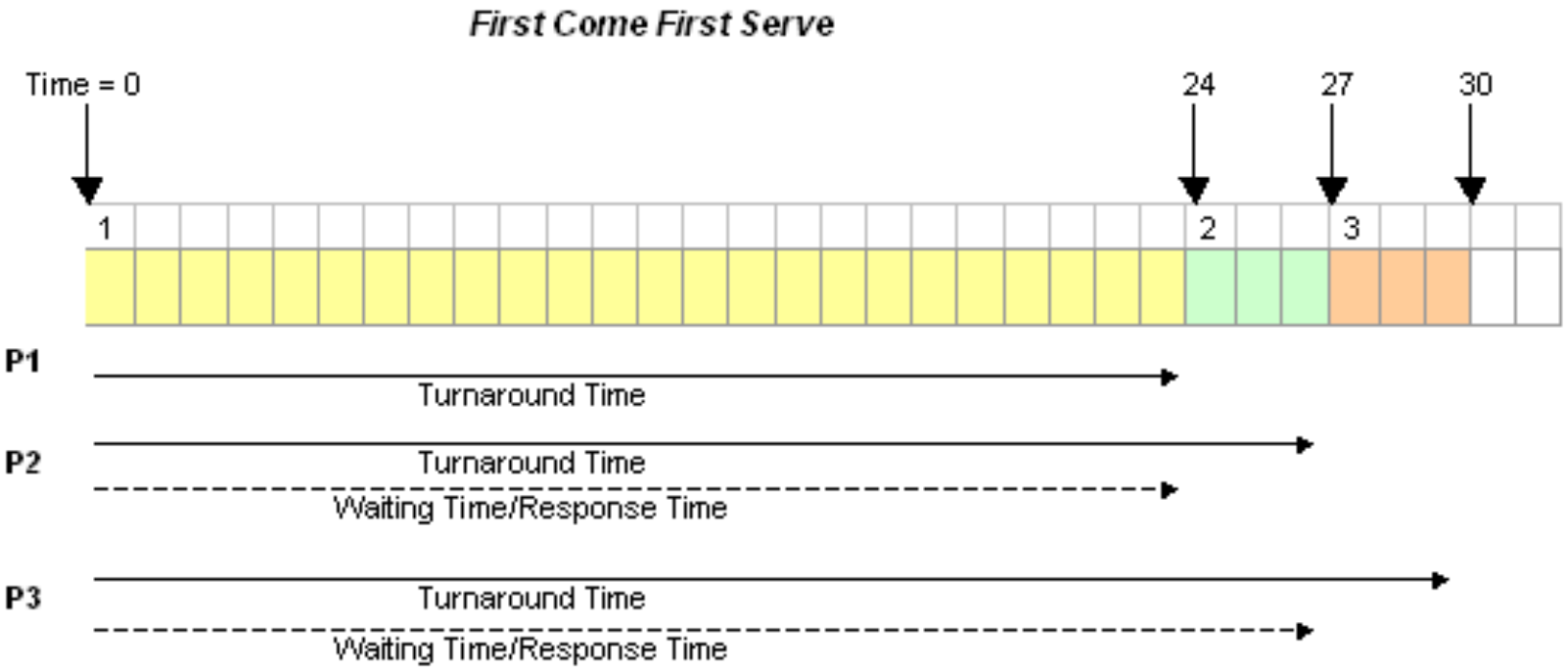
First-Come-First-Serve (FCFS)

- Selects a process based on the order of arrival or creation.
  - Assigns higher priority to the older processes
  - Non pre-emptive
  - Processes can continue until termination
- Strength
  - Simple implementation and low overhead
    - Few context switching
- Weakness
  - Short processes are waiting, starved of CPU
  - No guarantee of response time

First-Come-First-Serve (FCFS): Gnatt Chart

Process *(arriving at this order)*

Process	CPU Burst Time (ms)	Arrival Time
P1	24	0
P2	3	0
P3	3	0



First-Come-First-Serve (FCFS): Gnatt Chart

Example 3: Evaluating the characteristics of FCFS

Given the following processes, their CPU burst time and arrival time, calculate the average turnaround time, waiting time, and response time.

Process	*CPU Burst Time (ms)*	*Arrival Time*
P1	*24*	*0*
P2	*3*	*0*
P3	*3*	*0*

Answer:

The Gnatt chart is shown above. Turnaround time (TT) is clear from the chart and then we can work out the waiting time (WT) from the turnaround time. FCFS is non pre-emptive and so the waiting time is the same as the Response time (RT).

Process	*TT*	*WT*	*RT*
P1	*24*	*0*	*0*
P2	*27*	*24*	*24*
P3	*30*	*27*	*27*
Average	*27*	*17*	*17*

Shortest Job First (SJF)

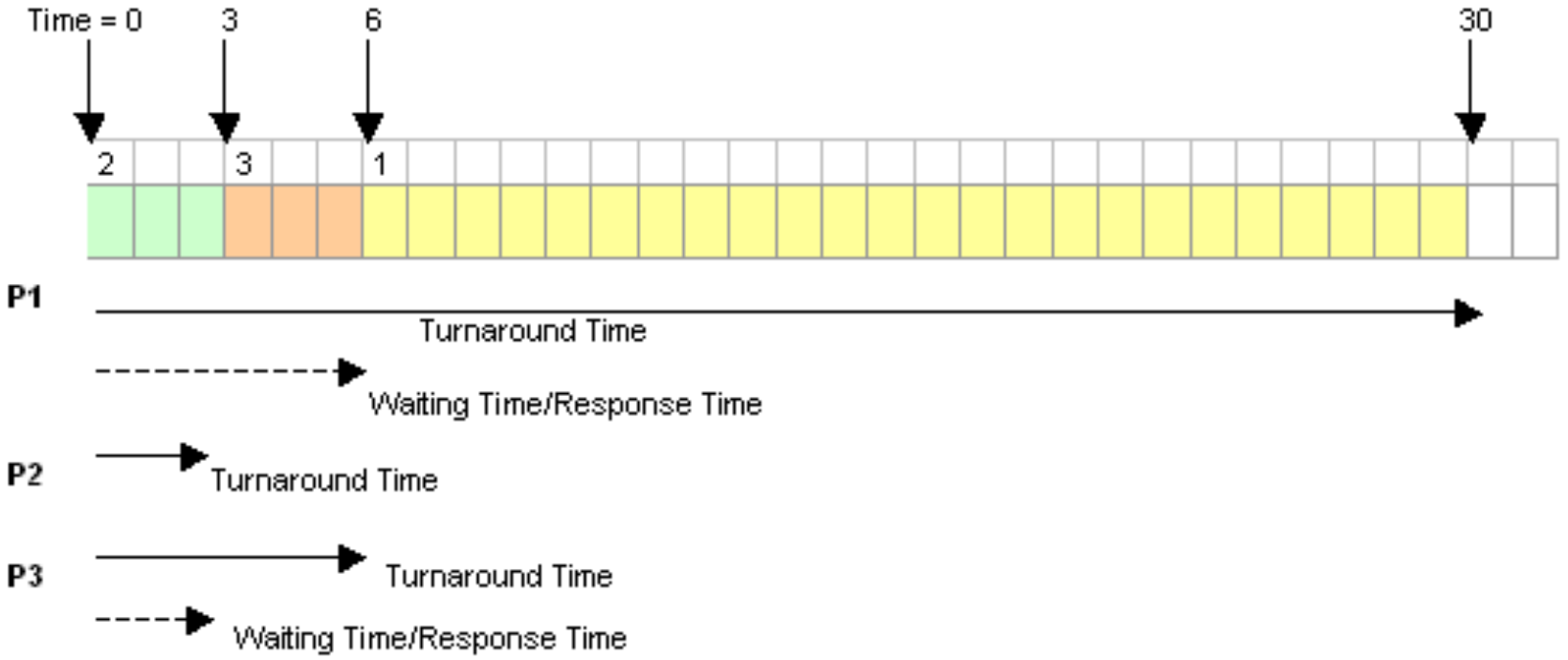
- Selects a process based on CPU burst time
  - Assigns higher priority to shorter CPU burst time processes
  - Non pre-emptive
  - Favour short processes
- Strength
  - Minimizing average waiting time
- Weakness
  - Actual turnaround time and response time for an individual process is difficult to predict
  - Long processes may wait forever

Shortest Job First (SJF)

Process *(arriving at this order)*

Process	*CPU Burst Time (ms)*	*Arrival Time*
P1	*24*	*0*
P2	*3*	*0*
P3	*3*	*0*

Shortest Job First (SJF) (Non-Pre-emptive)



Shortest Job First (SJF)

Example 4: Evaluating the characteristics of SJF

Given the following processes, their CPU burst time and arrival time, calculate the average turnaround time, waiting time, and response time.

Process	*CPU Burst Time (ms)*	*Arrival Time*
P1	*24*	*0*
P2	*3*	*0*
P3	*3*	*0*

Answer:

The Gnatt chart is shown above. Turnaround time (TT) is clear from the chart and then we can work out the waiting time (WT) from the turnaround time. SJF is non pre-emptive and so the waiting time is the same as the Response time (RT).

Process	*TT*	*WT*	*RT*
P1	*30*	*6*	*6*
P2	*3*	*0*	*0*
P3	*6*	*3*	*3*
Average	*13*	*3*	*3*

Overall, the SJF performs significantly better than FCFS in all three measurements. The fact that two out of three processes are short ones helps to produce this performance for SJF.

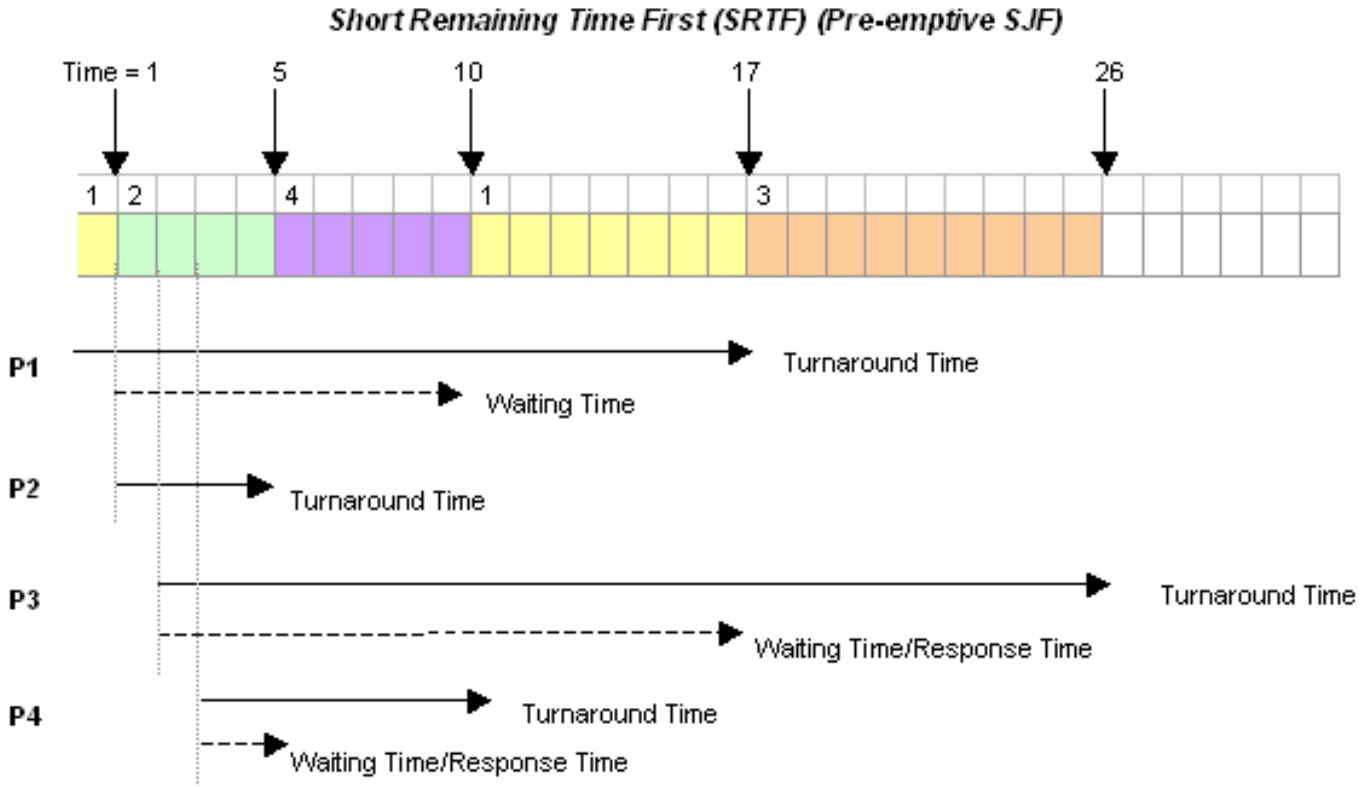
Shortest Remaining Time First (SRTF)

- Selects a process based on CPU burst time
  - Pre-emptive version of SJF
  - The priority is re-calculated when new process arrives
- Strength
  - New processes will not be waiting forever even when a long process is running
- Weakness
  - Greater overhead because of the more frequent context switching
  - Long processes may wait forever

Shortest Remaining Time First (SRTF)

Process (arriving at this order)

Process	CPU Burst Time (ms)	Arrival Time
P1	8	0
P2	4	1
P3	9	2
P4	5	3



Shortest Remaining Time First (SRTF)

Example 4: Evaluating the characteristics of SRTF

Given the following processes, their CPU burst time and arrival time, calculate the average turnaround time, waiting time, and response time.

Process	*CPU Burst Time (ms)*	*Arrival Time*
P1	*8*	*0*
P2	*4*	*1*
P3	*9*	*2*
P4	*5*	*3*

Answer:

The Gnatt chart is shown above. SRTF is pre-emptive and so the response time is not always the same as the waiting time. Another important point is that in this scenario the arrival time of the processes varied.

First we work out the TT. Be careful about the arrival time is not always zero.

Then work out the WT = CPU Burst – TT.

Finally the RT is worked out by checking the chart, finding when is the first time quantum of a process and then minus the arrival time.

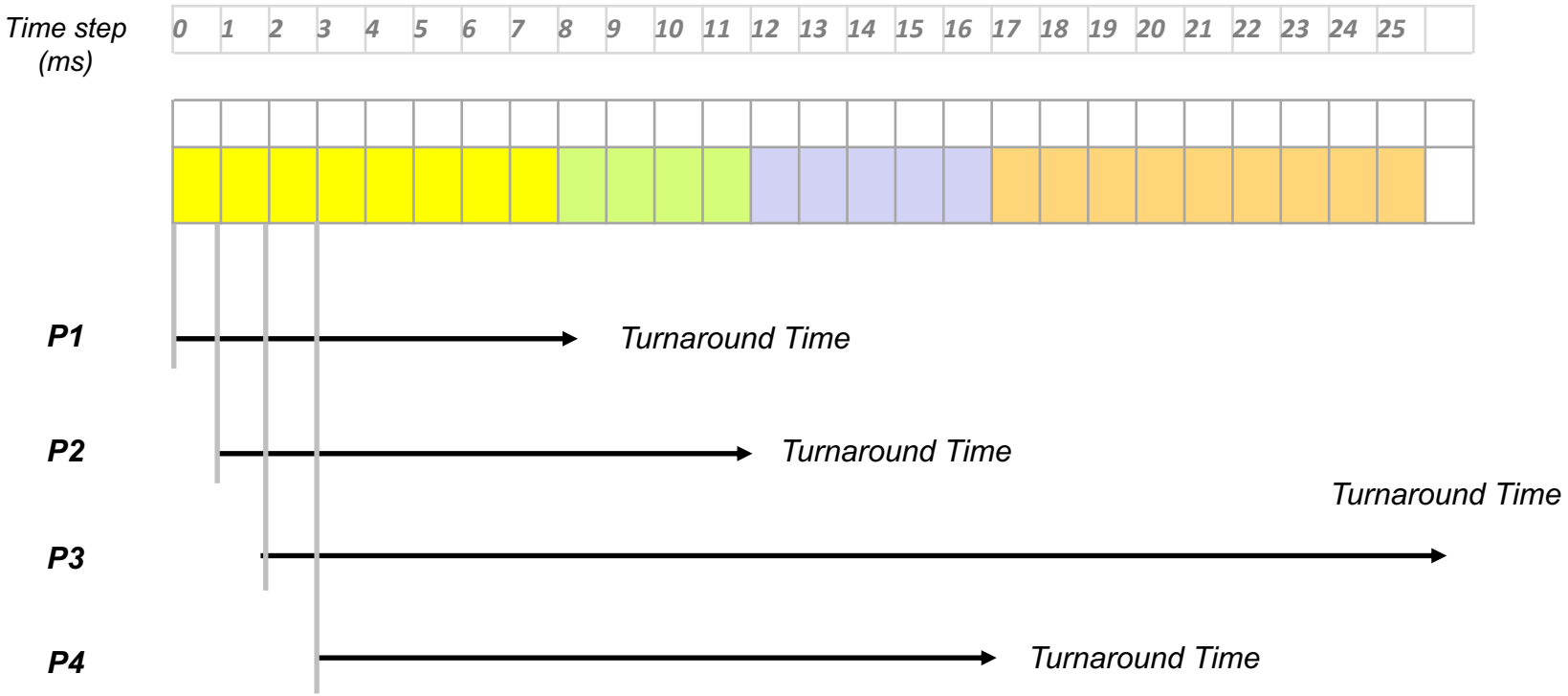
Process	*TT*	*WT*	*RT*
P1	*17*	*9*	*0*
P2	*4*	*0*	*0*
P3	*24*	*15*	*15*
P4	*7*	*2*	*2*
Average	*13*	*6.5*	*4.25*

Shortest Job First (SJF)

Processes *(arriving at this order)*

Process	CPU Burst Time (ms)	Arrival Time
P1	8	0
P2	4	1
P3	9	2
P4	5	3

TT	WT	RT
8	0	0
12-1 = 11	8-1=7	7
26-2 = 24	17-2=15	15
17-3 = 14	12-3=9	9
14	8	8



Highest Response Ratio Next (HRN)

- Solve the problem of starvation of long processes in SJF and SRTF
- Selects a process based on CPU burst time and waiting time
  - More sophisticated way to calculate priority

$$priority = (waiting\ time + cpu\text{-}burst\text{-}time) / cpu\text{-}burst\text{-}time$$

Highest Response Ratio Next (HRN)

- Strength
  - Long processes will have a higher priority as time passes.
  - The longer is the waiting time, the higher is the priority.
- Weakness
  - Greater overhead because of the need to keep accounting info of waiting time
  - A possible weakness is that it stills favour short jobs

Round Robin (RR)

- Ensures that each process has a chance to control the CPU
  - Assigning a time quota to each Running process
  - The time quota is known as the time quantum
  - Processes are queuing up for the use of CPU
  - A process is pre-empted when the assigned time quantum is expired, moved to the end of Ready queue or Terminate
  - RR scheduler will select the next process from the Ready queue

Round Robin (RR)

- Strength
  - Not biased towards long or short processes
  - Each process has a fair use of the CPU
- Weakness
  - High overhead due to the very frequent context switching
    - pre-emption.

Round Robin (RR)

- Time quantum size is important
  - Typical time quantum ranges from 10 to 100 milliseconds
  - Impression of processor sharing
    - Very short time quantum

Round Robin (RR)

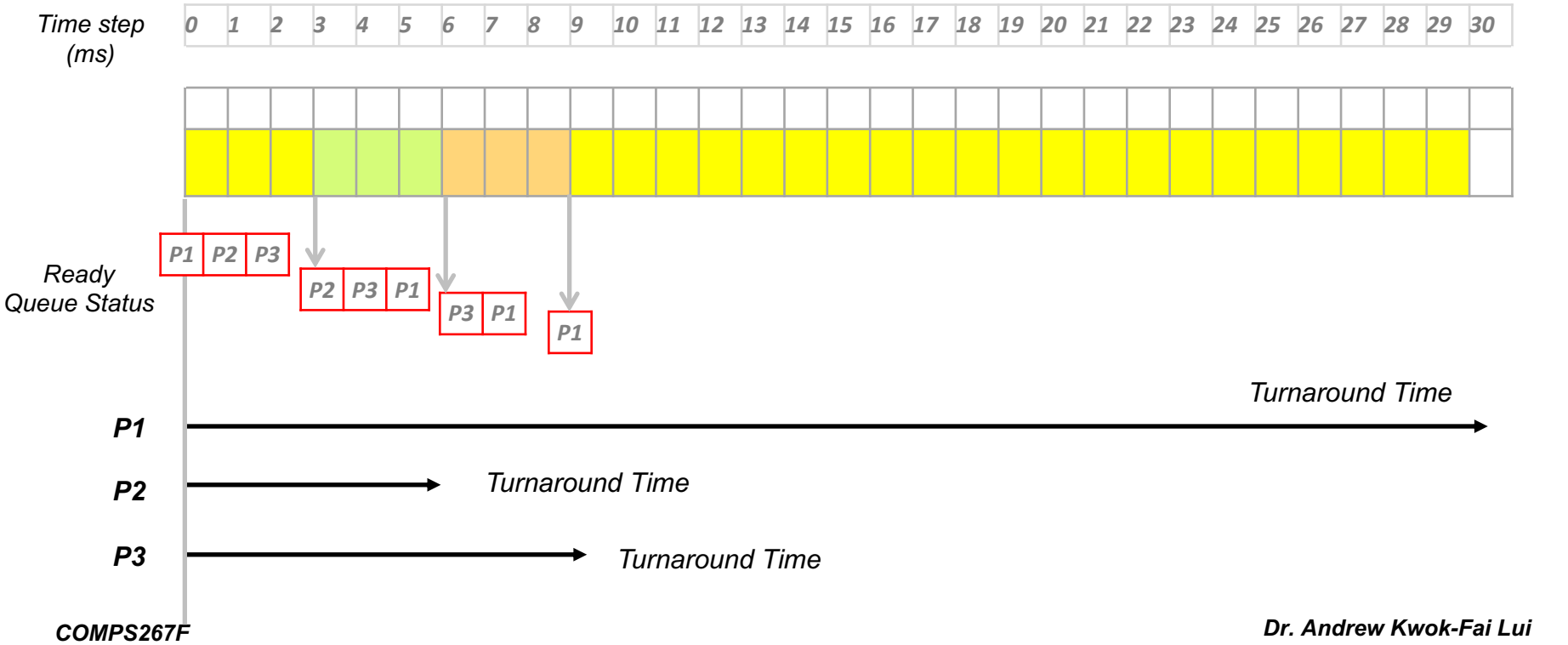
- Implementation is based on a circular queue or a linked list
  - System timer is used
    - Raises an interrupt causes the Running process to be put back to the Ready queue

Round Robin (RR)

Processes *(arriving at this order)*

Process	CPU Burst Time (ms)	Arrival Time
P1	24	0
P2	3	0
P3	3	0

TT	WT	RT
30	6	0
6	3	3
9	6	6
15	5	3



Round Robin (RR)

Example 5: Evaluating the characteristics of RR

Given the following processes, their CPU burst time and arrival time, calculate the average turnaround time, waiting time, and response time.

Consider that the time quantum is 3 ms.

Process	*CPU Burst Time (ms)*	*Arrival Time*
P1	*24*	*0*
P2	*3*	*0*
P3	*3*	*0*

Answer:

The Gnatt chart is shown above. RR is pre-emptive and the response time has to be worked out separately.

Process	*TT*	*WT*	*RT*
P1	*30*	*6*	*0*
P2	*6*	*3*	*3*
P3	*9*	*6*	*6*
Average	*15*	*5*	*3*

Comparing to other algorithms, the measurements are not as good as SJF but close enough. The key characteristic is a small RT for all processes and no process would perform especially poorly.

The Ready Queue

Arrival at the Queue

Round Robin (RR)

Example 6: Evaluating the characteristics of RR

Given the following processes, their CPU burst time and arrival time, calculate the average turnaround time, waiting time, and response time.

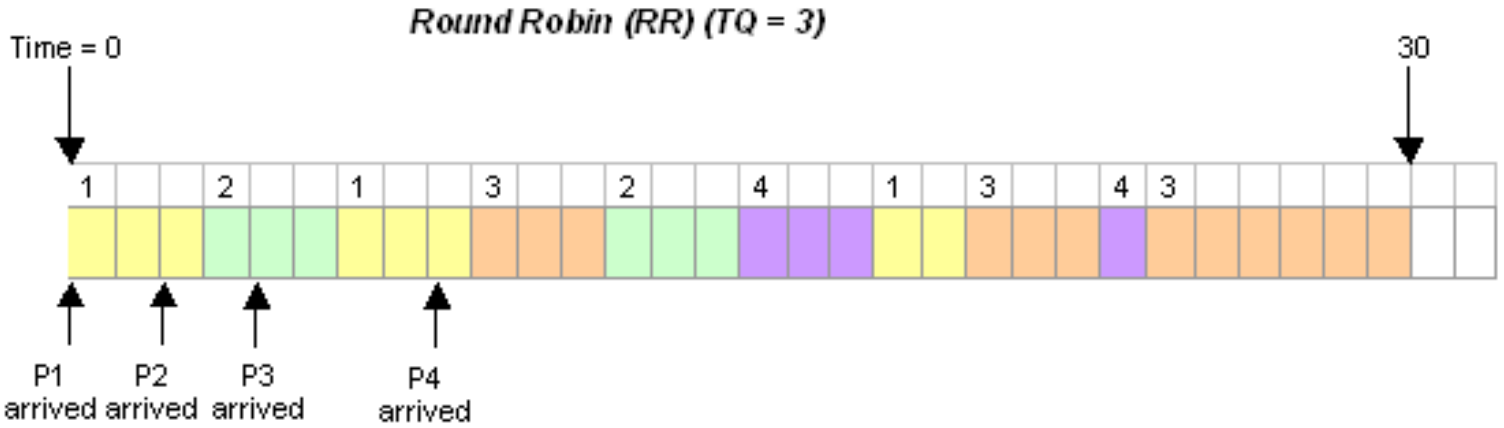
Consider that the time quantum is 3 ms.

Process	CPU Burst Time (ms)	Arrival Time
P1	8	0
P2	6	2
P3	12	4
P4	4	8

Round Robin (RR)

Answer:

Draw the Gnatt chart first. Except for the simplest scenarios, evaluating RR scheduling would involve keeping track of the state of the ready queue.



The Gnatt chart is shown above. The RT is the time when a process controls the CPU the first time, minus the arrival time.

Process	*TT*	*WT*	*RT*
P1	*20 – 0 = 20*	*20 – 8 = 12*	*0*
P2	*15 – 2 = 13*	*13 – 6 = 7*	*3 – 2 = 1*
P3	*30 – 4 = 26*	*26 – 12 = 14*	*9 – 4 = 5*
P4	*24 - 8 = 16*	*16 – 4 = 12*	*15 – 8 = 7*
Average	*18.75*	*11.25*	*3.25*

Round Robin (RR)

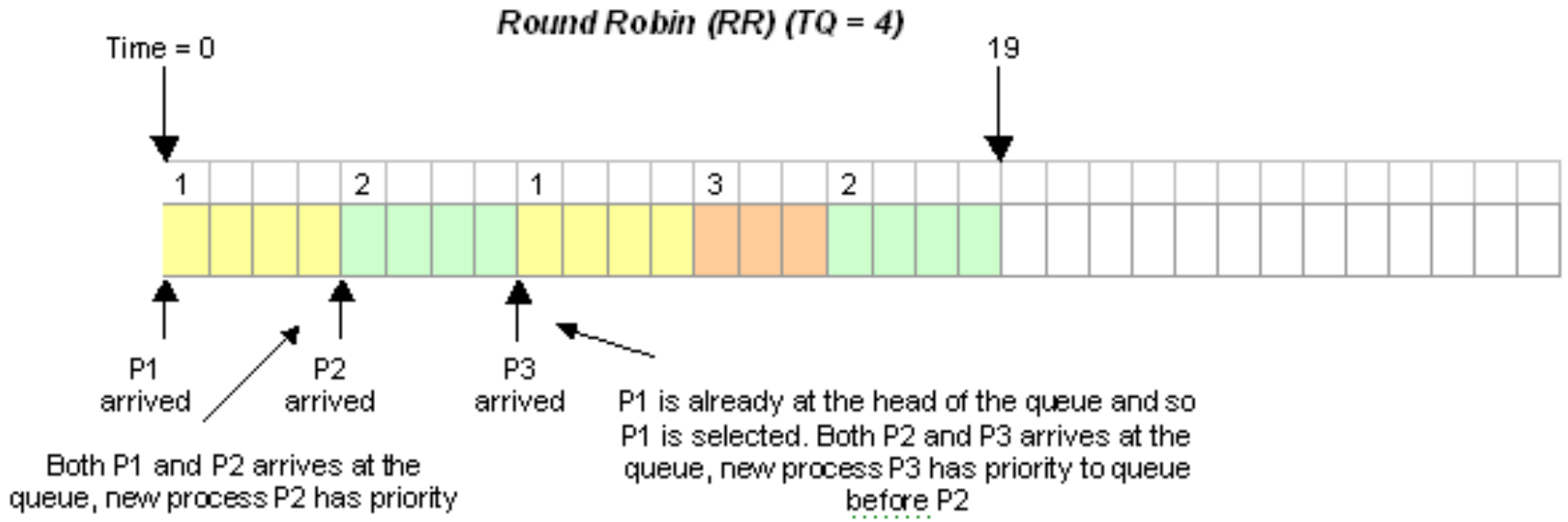
Example 7: Evaluating the characteristics of RR

Given the following processes, their CPU burst time and arrival time, calculate the average turnaround time, waiting time, and response time. Consider that the time quantum is 4 ms.

Process	*CPU Burst Time (ms)*	*Arrival Time*
P1	*8*	*0*
P2	*8*	*4*
P3	*3*	*8*

(new processes take priority than old processes)

Round Robin (RR)



The Gnatt chart is shown above. There are two instances when there is a clash of two processes joining the ready queue. At time = 4 and time = 8. In both cases, the new process joins the queue before the old process.

Process	*TT*	*WT*	*RT*
P1	*12 - 0 = 12*	*12 – 8 = 4*	*0*
P2	*19 – 4 = 15*	*15 – 8 = 7*	*4 – 4 = 0*
P3	*15 – 8 = 7*	*7 – 3 = 4*	*12 – 8 = 4*
Average	*11.33*	*5*	*1.33*

Multi-Level Queue Scheduling (MLQ)

- Processes can be characterised into several groups of distinctive behaviours
  - Each group of processes is put into a separate queue
  - Each queue is managed by a scheduling algorithm
  - A composite algorithm can be designed to exploit knowledge about different groups of processes

Multi-Level Queue Scheduling (MLQ)

- Typical characterisation of processes
  - Interactive processes are scheduled by RR scheduler with small time quantum.
    - Minimize the response time.
  - Short processes are scheduled by RR scheduler with larger time quantum.
    - Reasonable performance for majority of processes
  - Long processes are put in a group that is not scheduled until there is no other process in the ready queue
    - Long processes will not block short processes

Multi-Level Queue Scheduling (MLQ)

- Typical classification of processes
  - System processes.
  - Interactive processes.
  - Interactive editing processes
  - Batch processes.
  - Student processes.

Multi-Level Queue Scheduling (MLQ)

- Allocation of CPU time to the queues
  - One possible way is allocation by percentage
    - For example, a high priority queue is given 60 percent, and a lower priority queue gets 10 percent.

Multi-Level Queue Scheduling (MLQ)

- Strength
  - Able to handle a mixed class of processes suitably
  - Generally acceptable performance.
- Weakness
  - Complex implementation
  - Difficulty to know the actual characteristics of each process.
    - Process is placed in the correct queue not easy

Multi-Level Feedback Queue Scheduling (MLFQ)

- Based on MLQ allows processes to move from one queue to another queue dynamically
  - MLFQ has the opportunity to observe the characteristics of a process
  - Then move the process to the appropriate queue

Multi-Level Feedback Queue Scheduling (MLFQ)

- Example
  - If a process is found to use a lot of CPU time, it is moved to a lower priority queue.
    - I/O bound processes and interactive processes are those left in the higher priority queue

Multi-Level Feedback Queue Scheduling (MLFQ)

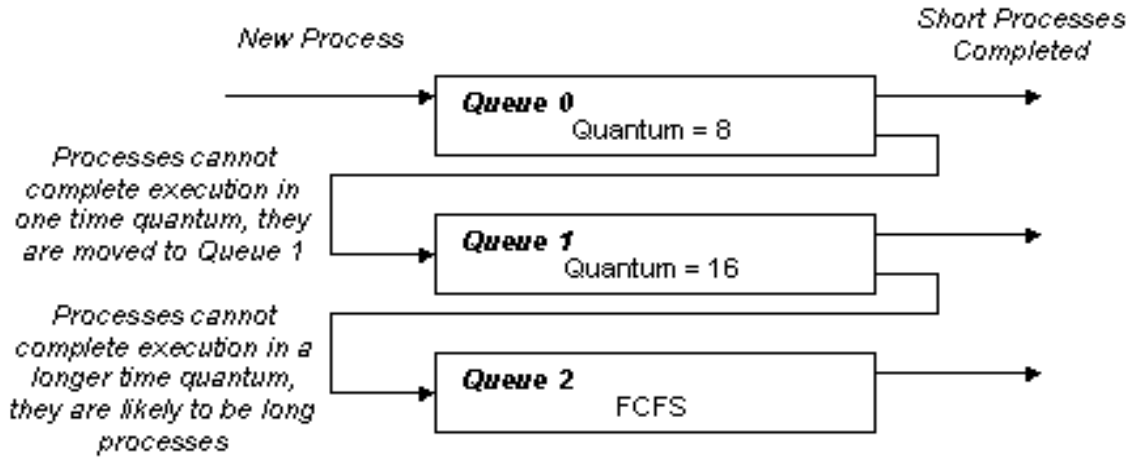
Example 8: Design a MLFQ

MLFQ has not specified a rigid scheduler. Rather it allows flexibility to have a particular of design of MLFQ to achieve a specific objective.

Design a MLFQ based scheduler that favours short process and makes sure long processes are not executed until there is no waiting short process.

Answer:

The design of the MLFQ is shown below.



New processes will come in queue 0. Queue 0 has time quantum of 8. Once the process used up the time quantum, it is pre-empted and moved to queue 1. Queue 1 has time quantum of 16. A process used up the time quantum is moved to queue 2. Queue 2 uses FCFS. Processes that ended up in Queue 2 are likely to be long processes.

process scheduler evaluation

Scientific Evaluation by Experiments

- Determining causal relationships between variables
  - Independent variables: the algorithm and the characteristics of the workload (i.e. process sets)
    - Workload characteristics: long, short, mixed, etc
  - Dependent variables: the performance metrics

Deterministic Modelling

- This method employs pre-determined workload to test the performance of each scheduling algorithm
- Advantages
  - Simple, fast, and quantitative
- Disadvantages
  - Too specific and the chosen pre-determined workload needs to be representative for any generalization

Simulations

- Simulations is the use of a program to model after a computer system.
  - Parameters of the computer system are modelled in the simulation
  - The data used in driving the simulations is most important.
  - Suitable data must reflect the characteristics of the real world cases, which the program is simulating.

Scientific Evaluation by Analysis

- Analyze the scheduling algorithm and use existing models to explain operations and processes

Queuing Model

- The real world processes vary significantly.
  - Each process may have its own characteristics of CPU burst and I/O burst
  - A useful abstraction of the process is the proportion of CPU burst and I/O burst
  - Arrival rates and service rates can also be modelled mathematically
- Little's formula, an useful estimation between average queue length, average waiting time, and average arrival rate.
  - Assume that the number of processes leaving the queue equals the number of processes arriving.

$$n = \lambda \times W$$

Summary

- How to choose a CPU scheduler?
  - Depends on the requirement or situation
- Step-by-step approach
  - Specify the performance criteria
  - Weights the criteria
  - An example is maximizing CPU utilization under the constraint that maximum response time is 1 second.

case study: windows os scheduling

Windows Family of OS

- Supported PC users since 1990s.
  - Most popular OS for desktop users
- The dispatcher of Windows OS is priority-based
  - A running thread can continue until:
    - Termination
    - Time quantum
    - Another higher priority process
    - Making I/O blocking system call
  - A 32-level priority scheme
    - Level 1 to 15: variable class priority
    - Level 16 to 31: real-time class priority
    - Level 0: system thread for memory management

Windows Family of OS

- The priority can change
  - The dispatcher can adjust based on situation
  - The user can change it in the Task Manager
- Windows OS boost performance of interactive programs.
  - Distinguish foreground (higher priority) and background (hidden from users)