Operating Systems

Lecture 4b

Dr Ronald Grau

School of Engineering and Informatics

Spring term 2018

Previously

Scheduling

- Scheduling policies & performance
- Multi-level queue scheduling
- Feedback scheduling
- Real-time scheduling
- Java thread scheduling

Today

Evaluating Scheduling Algorithms

- Deterministic Evaluation
- Probabilistic Evaluation
- Stochastic Evaluation
- Simulation
- Implementation & Testing

- Coursework 1 introduction

Recap: Scheduling

Questions:

- What happens if the chosen time quantum for RR is too large?

- What is the purpose of multi-level queue scheduling?

- What is the effect of using feedback in multi-level queue scheduling?
- In what kind of system would we commonly find periodic processes?

- Under which conditions can a set of periodic processes be scheduled to not miss their deadlines?
- Can a Java thread be preempted by the OS?

How to select a CPU scheduling algorithm?

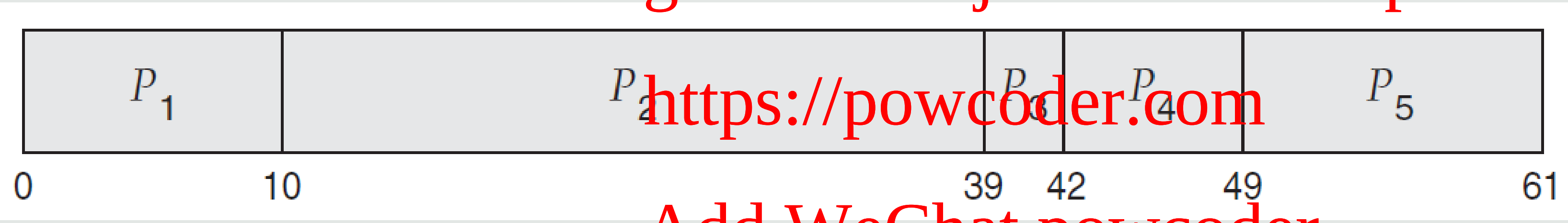
Performance Evaluation

- Factors to consider:
  - Type of system
  - Expected process characteristics
  - Expected workload
  - Scheduling criteria/goals ! metrics
- Evaluation methods:
  - Deterministic
  - Probabilistic
  - Stochastic
  - Testing the implementation (ultimately)

Deterministic Evaluation

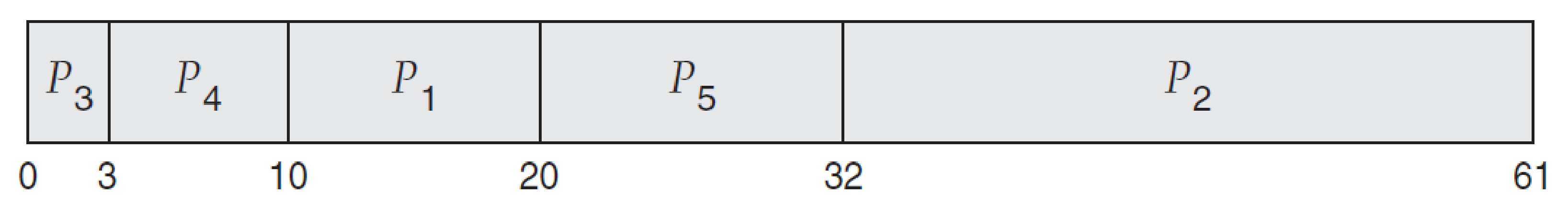
Given a predefined work load

- First-come first-served (FCFS)



Average waiting time: 28ms

- Shortest Job First (SJF)



Average waiting time: 13ms

Process	Burst time
P1	10
P2	29
P3	3
P4	7
P5	12

Deterministic Evaluation

Given a predefined work load

- Round Robin (RR) : 10ms time quantum

Average waiting time: 23ms

Process	Burst time
P1	10
P2	29
P3	3
P4	7
P5	12

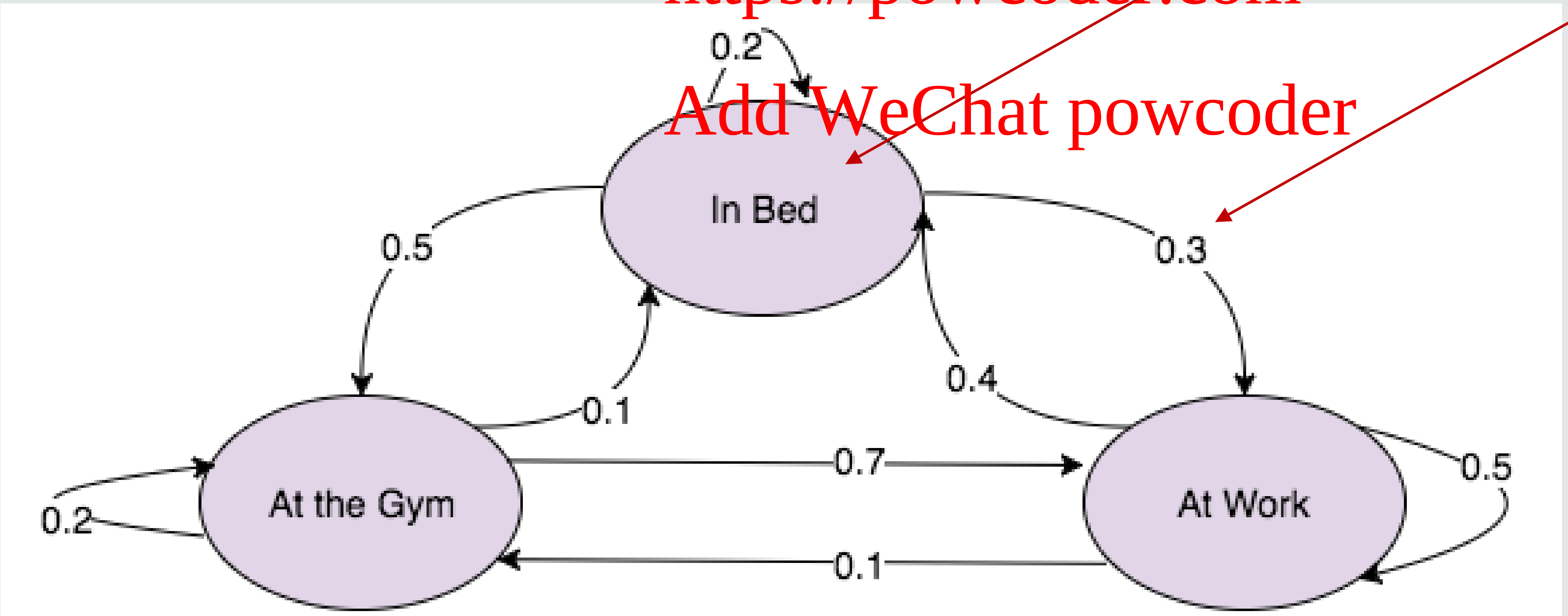
Probabilistic Evaluation

Analytical models

- Mathematical representations of computer systems
- E.g. Markov chains

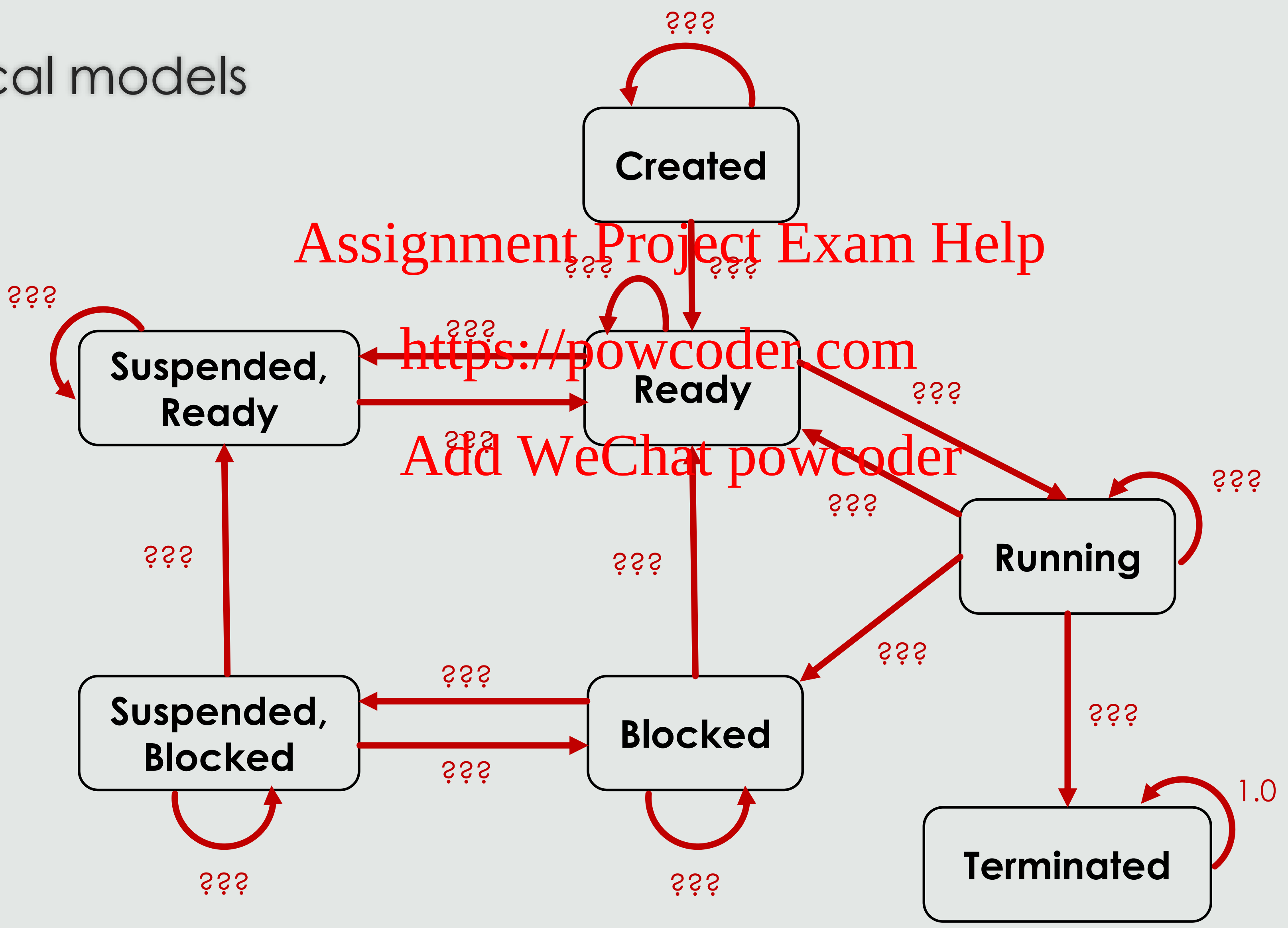
State

Probability of state change



Probabilistic Evaluation

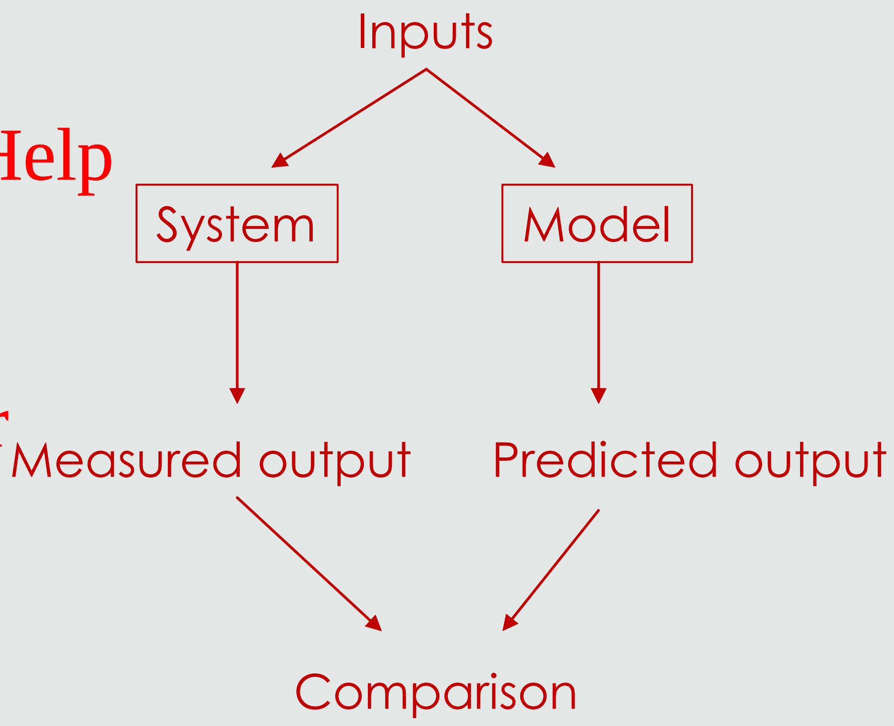
Analytical models



Probabilistic Evaluation

Analytical models

- Pros
  - Large body of existing work
  - Can be applied to new problems
  - Relatively fast and accurate
- Cons
  - Probabilities are only approximations of real behaviour
  - Systems can be too complex to model everything
  - Must use other techniques to validate results

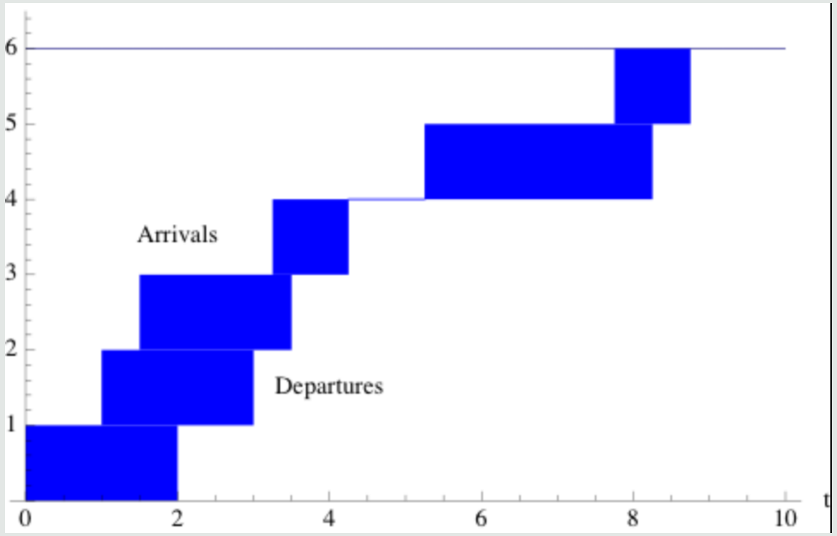


Queueing Models

Computer system:
A network of servers, each with a queue of waiting processes

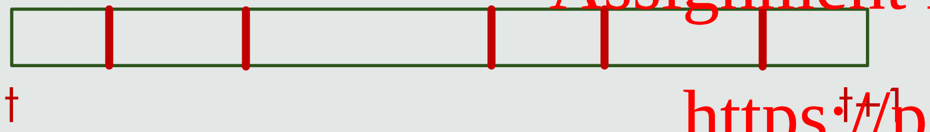
- Knowing arrival rates and service rates
- Computes throughput, average queue length, average wait time, etc

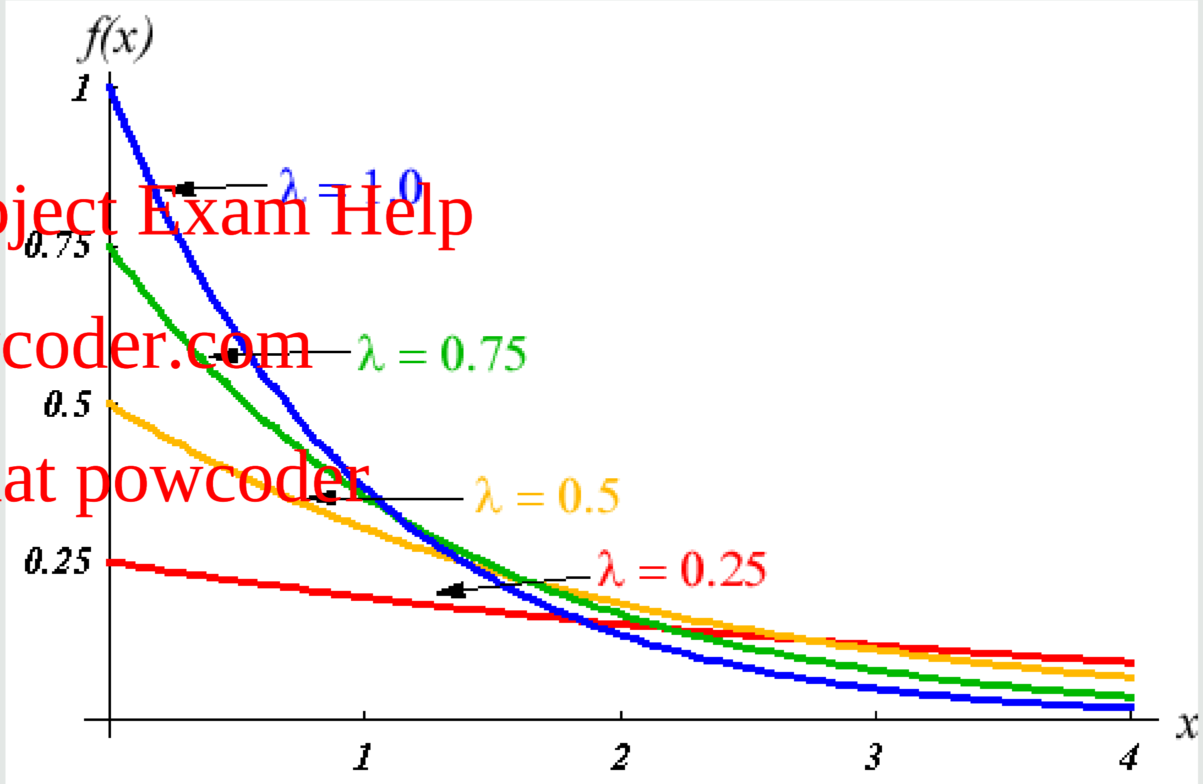
- Arrival of processes & CPU and I/O bursts can be determined probabilistically
- Typically, these are exponentially distributed and described by their mean value



Exponential Distribution

- Rate $\lambda > 0$
- E.g. arrival times





- Probability density:

$$f(x) = \lambda * e^{-\lambda * x} \quad \text{(for x} \geq 0\text{)}$$

- Mean: $^1/_\lambda$

- Probability of arrival interval for one process: $P(l < x < u) = e^{-\lambda * l} - e^{-\lambda * u}$

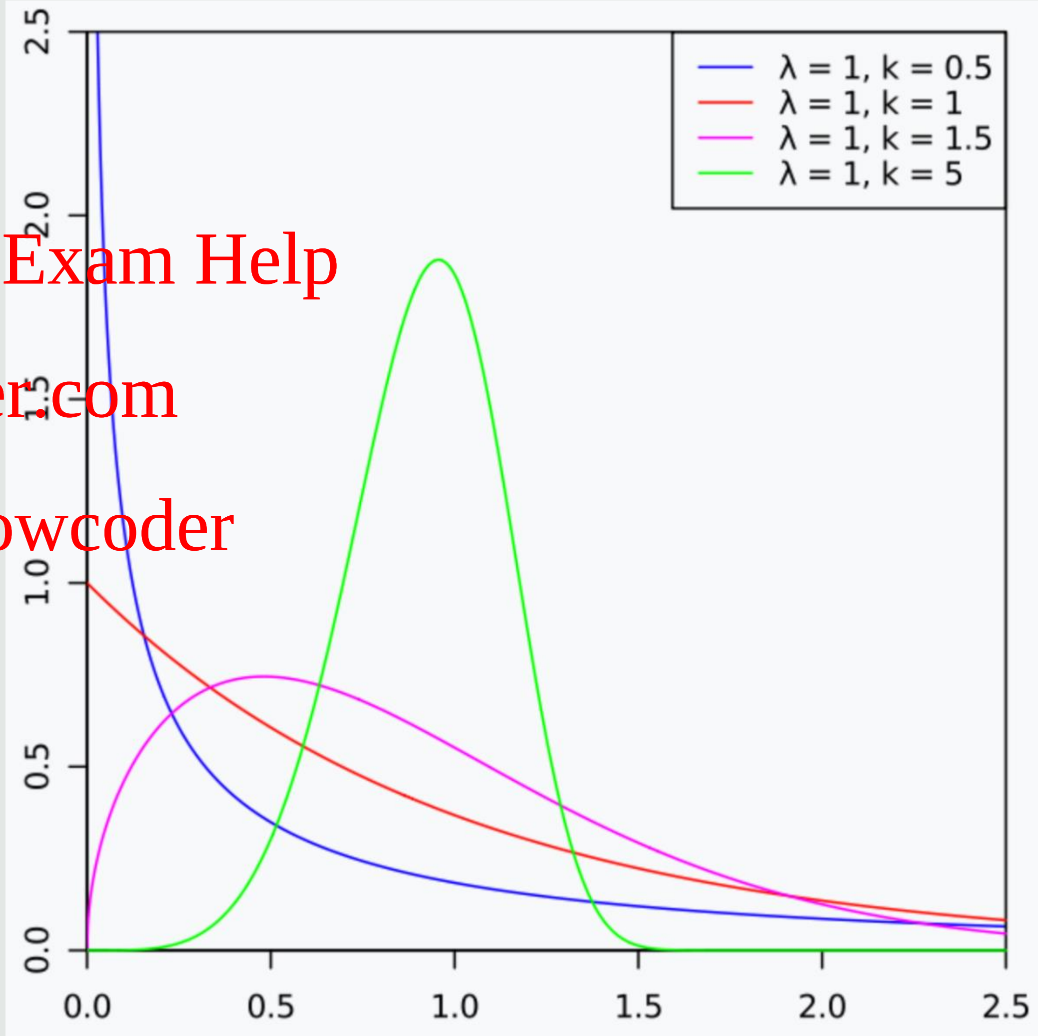
Weibull Distribution

- Rate: λ
- Shape: k (β)

  $k = 1$ (exponential)

  $k < 1$

  $k > 1$



- E.g. decreasing/increasing failure rates
- Other distributions:
  Uniform, Gaussian, Rayleigh, Fisher, etc.

Little's Law

Assuming a system in steady state, processes that are leaving the queue must equal processes arriving at the queue

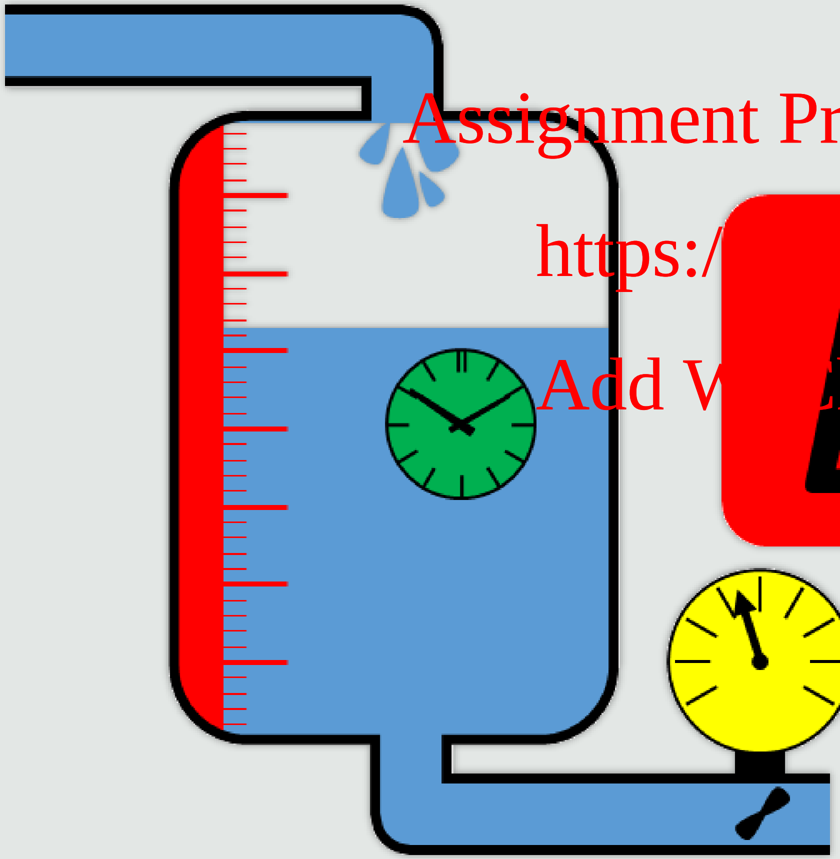
$$L = \lambda * W$$

where

- L : average queue length
- W : average waiting time in queue
- λ : average arrival rate into queue

- Valid for any scheduling algorithm and arrival distribution
- Given two parameters we can determine the third:
  - For example, if 7 processes arrive per second on average, and there are 14 processes in queue on average, then the average wait time per process = 2 seconds.

Little's Law – Conservation of Flow



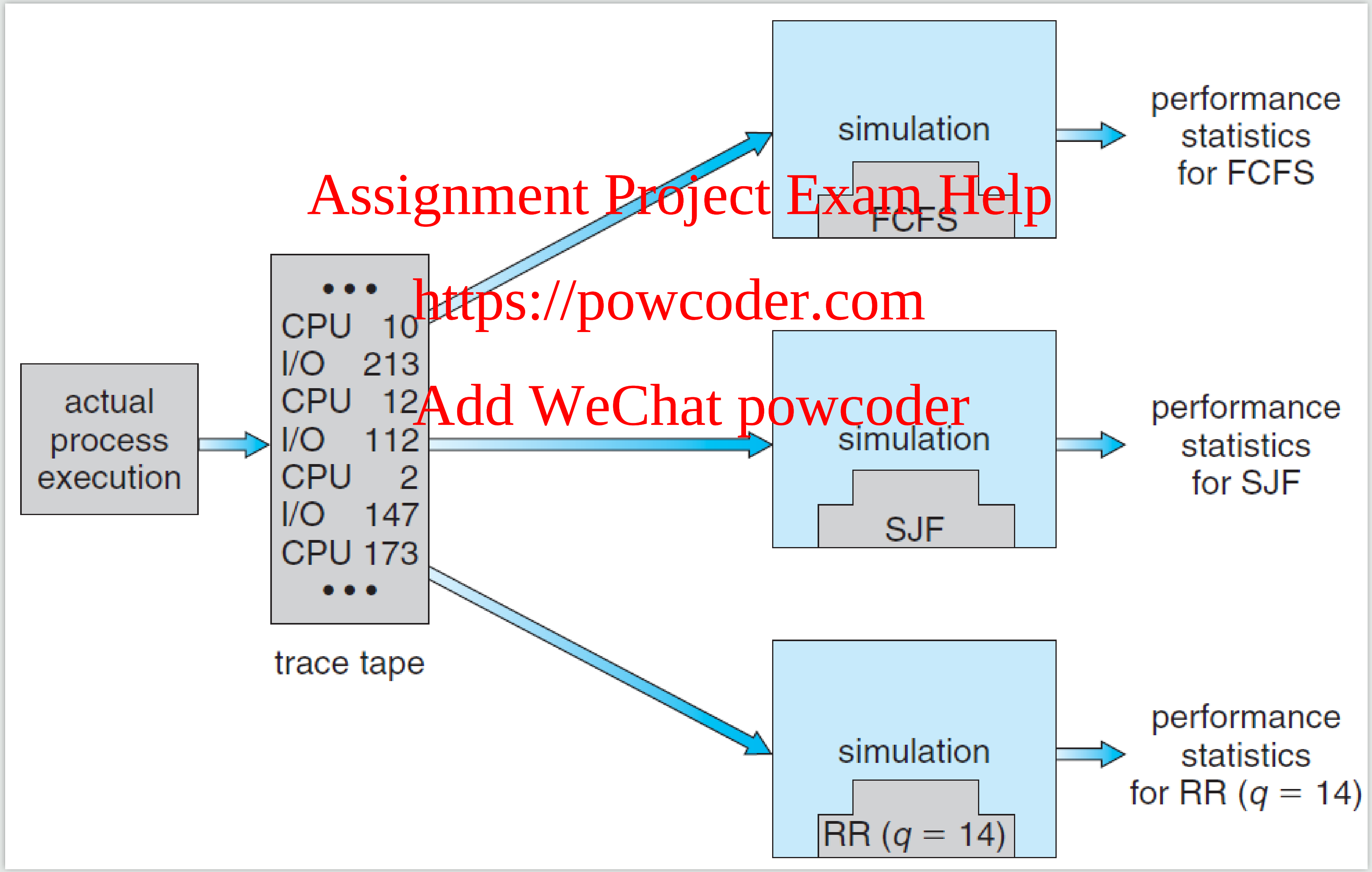
$$L = \lambda \cdot W$$

Stochastic Evaluation

Simulation

- Programmed model of computer system
- Gather statistics indicating algorithm performance
- Data to drive simulation gathered via
  - Random generation informed by probability distributions (mathematical/empirical)
  - Data recorded from real systems (trace tape of sequences of events)
- Pros & Cons
  - More accurate
  - Bugs in the simulation
  - Imprecise modelling, deliberate omissions (due to complexity)
  - Results of a simulation must be validated

Simulation


actual process execution
• • •
CPU 10
I/O 213
CPU 12
I/O 112
CPU 2
I/O 147
CPU 173
• • •
trace tape
simulation
FCFS
performance statistics for FCFS
simulation
SJF
performance statistics for SJF
simulation
RR (q = 14)
performance statistics for RR (q = 14)

Implementation and Testing

The real thing

- Even the best simulations have limited accuracy
- We could also implement our own scheduler and
  - Test it with hand-crafted / random-generated data
  - Test it in real systems
- Parameterisation of scheduler important for compatibility & fine-tuning
- High cost, high risk

Summary

Evaluation of scheduling algorithms

- Deterministic evaluation
- Probabilistic evaluation
  - Queueing models
  - Little's Law
- Stochastic evaluation
  - Simulation models

Read

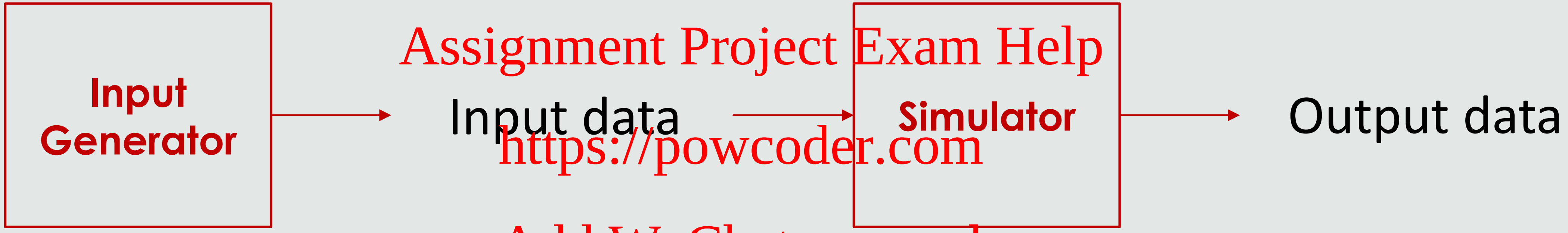
- Silberschatz et al., Operating System Concepts
  - Chapter 5.8

Coursework 1

Basics

- Demonstrate that you understand how CPU Scheduling works
- Write simple classes to simulate scheduling algorithms
- Create, run and save experimental data from simulations
- Write a report to present your results
  - Create hypotheses about expected behaviour and discuss your evidence / results
  - Basic statistics (mean, variance, etc.)
  - Visualise data (tables, charts, etc.)

Coursework 1



- Which scheduling algorithm is most suitable for different kinds of workloads?

Coursework 1


```
public abstract class AbstractScheduler {

  // Initializes with given parameters
  public void initialize(Properties parameters) {
  }

  // Adds a process to the ready queue.
  public abstract void ready(Process process);

  // Returns the next process to be run
  // and removes it from the ready queue
  public abstract Process schedule();
}
```

Coursework 1

- Find the assignment description and detailed guidelines on Study Direct (Module Assessment)
- Read the guidelines carefully!
- E-submission via Study Direct
- Deadline 22 March 2018 , 4PM

Next Lecture

- Introduction
- Operating System Architectures
- Processes
- Threads - Programming
- Process Scheduling - Evaluation
- **Process Synchronisation**
- Deadlocks
- Memory Management
- File Systems
- Input / Output
- Security and Virtualisation