CS 6323
Complex Networks and Systems:
Modeling and Inference

Integrated System Modeling

Prof. Gregory Provan
Department of Computer Science
University College Cork

Basic Concepts

- Concepts:
  - *computation capacity*, the *mean computation before failure*, and the *computation availability*.

  - The "**computation capacity**" is the amount of useful computation per unit time (a performance measure)
  - the **computation availability** is defined as the expected value of the *computation capacity* of the system at time *t* or in steady-state operation.

- While these metrics are declared "performance-related reliability measures", they actually have more "performance" flavor.

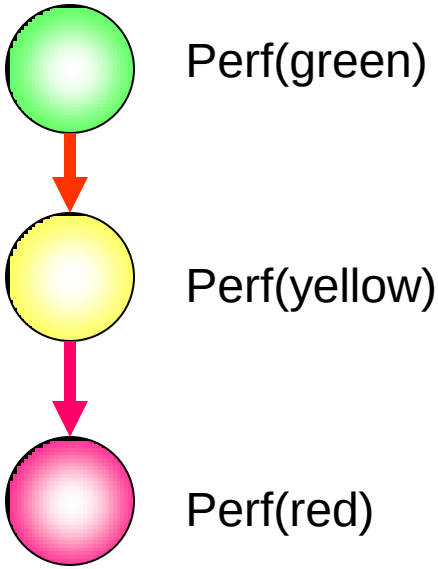
The first method of modeling performability

Computing the expected value of certain performance attributes, with the consideration of failures, was the first method introduced in performability estimation.

$$E[\text{performance}] = \sum_{\forall i} Perf(i) \times Prob(i)$$

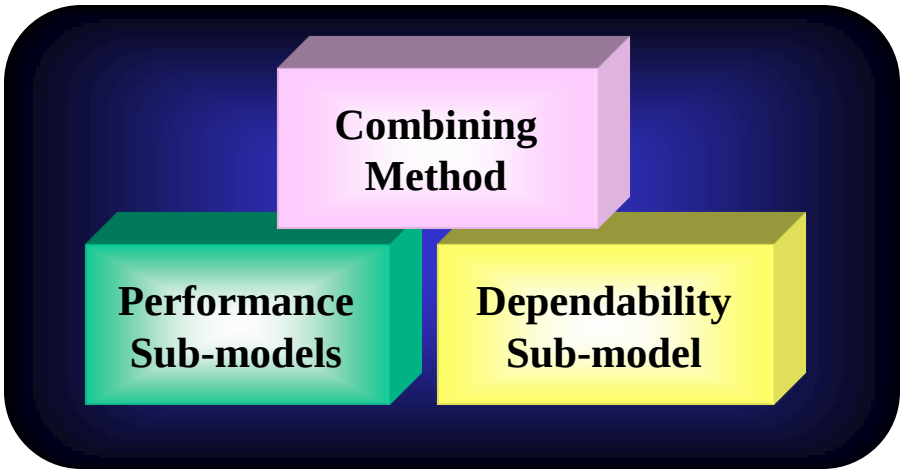
Performance analysis

Dependability analysis

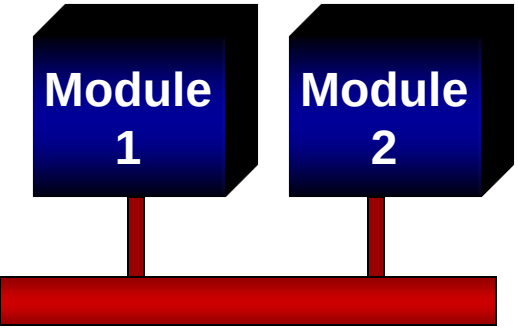


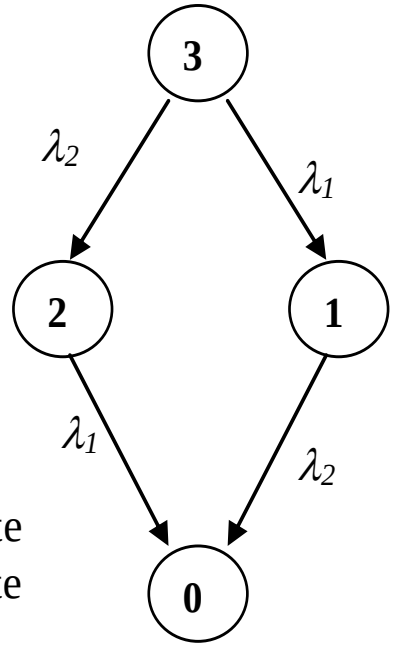
Structure of a Performability Model

- A dependability sub-model
  - A state space model describing the failure/repair behavior
- A performance sub-model
  - Model(s) describing the **performance** of a system
- A **method** of combing the results from the two sub-models



A 2-Parallel Processor System with Simple "Reward" Performance Model





λ_1: module 1 failure rate
λ_2: module 2 failure rate

The Baseline Model

States Description

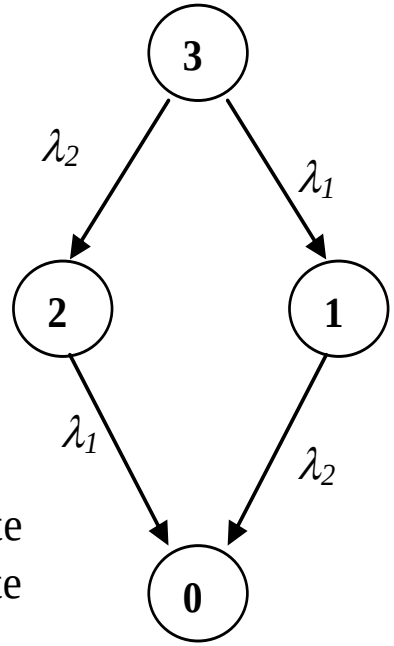
State 3: Both modules 1 and 2 are working
State 2: Module 2 is failed, module 1 is working
State 1: Module 1 is failed, module 2 is working
State 0: Both modules are failed

Reward values

State 3: Performance level 3, l_3
State 2: Performance level 2, l_2
State 1: Performance level 1, l_1
State 0: Performance level 0, l_0

Reward for Simple Model

- Expected reward= $\sum_i Pr(S_i) * l_i$



λ_1: module 1 failure rate
λ_2: module 2 failure rate

The Baseline Model

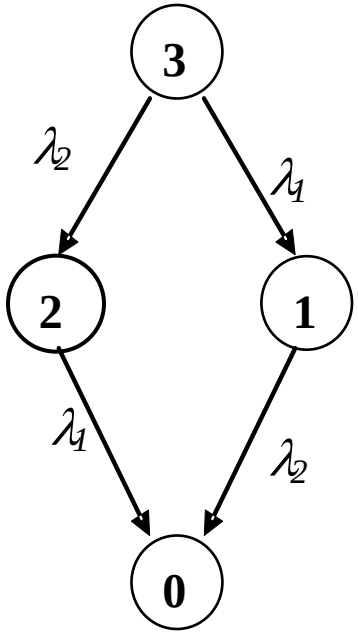
States Description

State 3: Both modules 1 and 2 are working
State 2: Module 2 is failed, module 1 is working
State 1: Module 1 is failed, module 2 is working
State 0: Both modules are failed

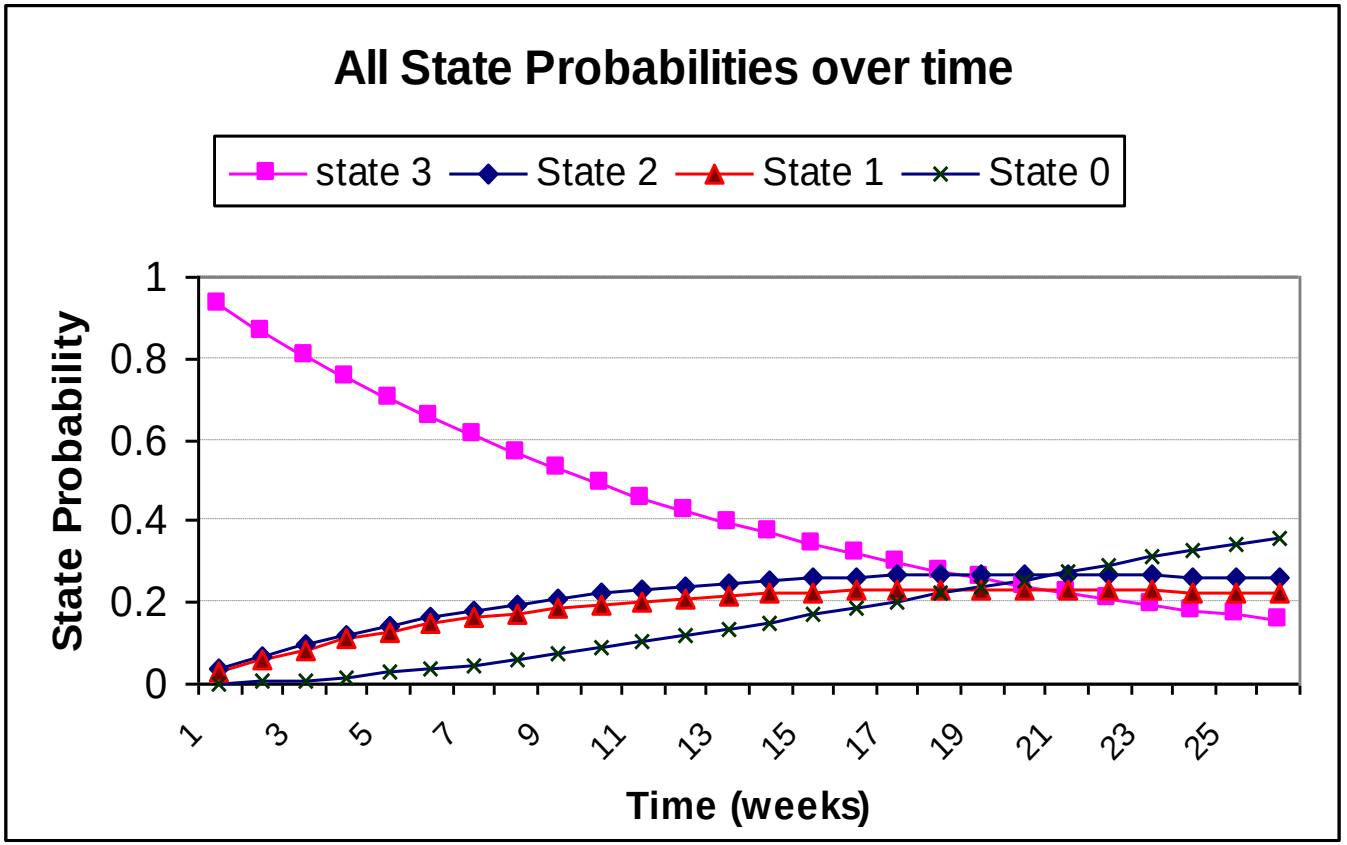
Reward values

State 3: Performance level 3, l_3
State 2: Performance level 2, l_2
State 1: Performance level 1, l_1
State 0: Performance level 0, l_0

Example 1: State Probabilities

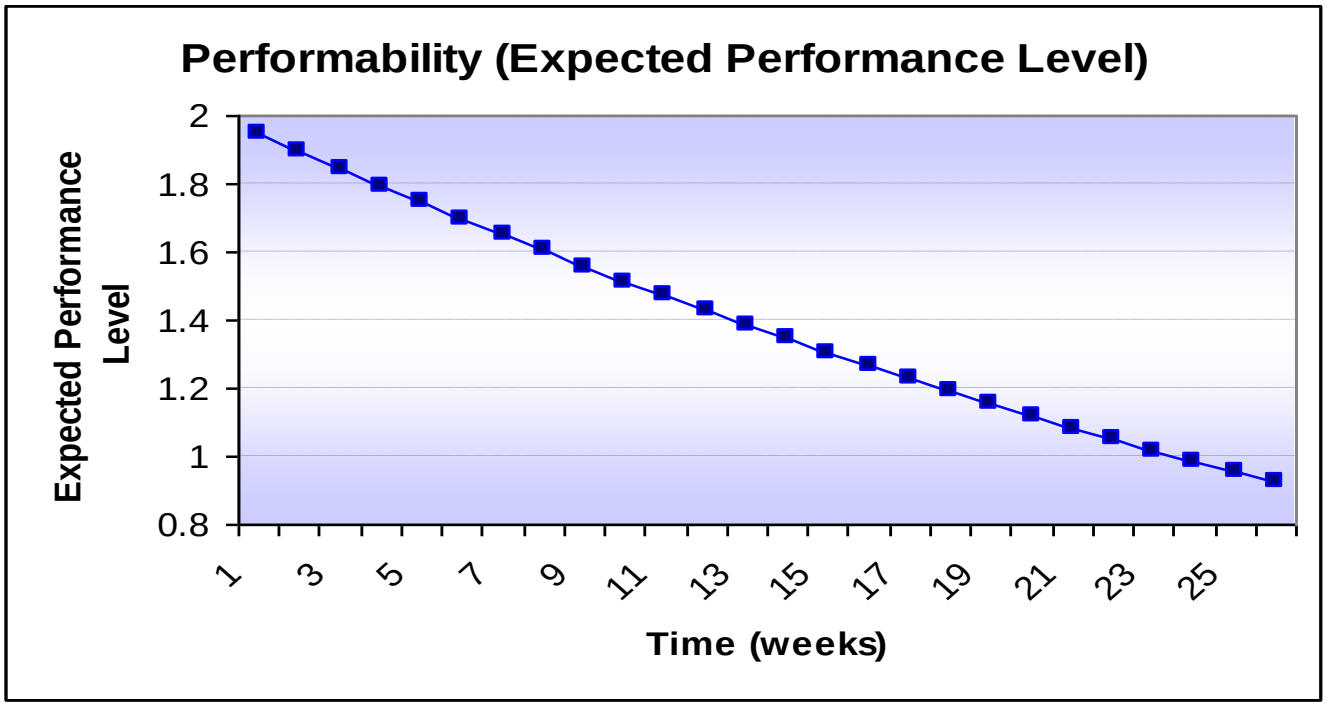


State Probabilities over 26-week Period of Time



Example 1: Performability as a Function of Time

A simple performability measure: $\sum_{i=0}^{3} li \times pi$

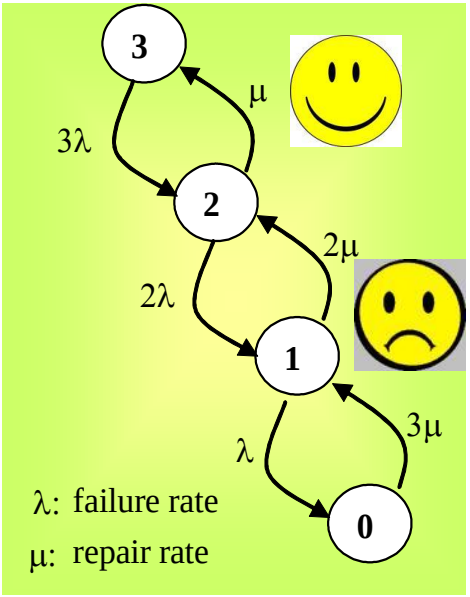


Performability Assessment over a 26-week Period of Time

Markov Reward Model

- Reward rate: a **constant** for each state in Ex. 1
- When the reward rates take only two values
  - "0" means failed
  - "1" means working

  performability is the system's availability or reliability.
- In general, performance need to be assessed under
  - The fully operational condition, and
  - all degraded conditions

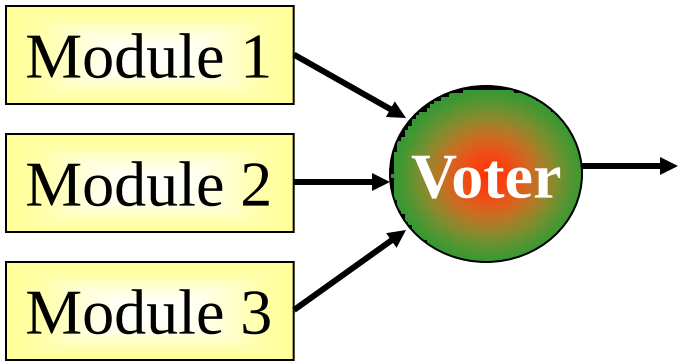


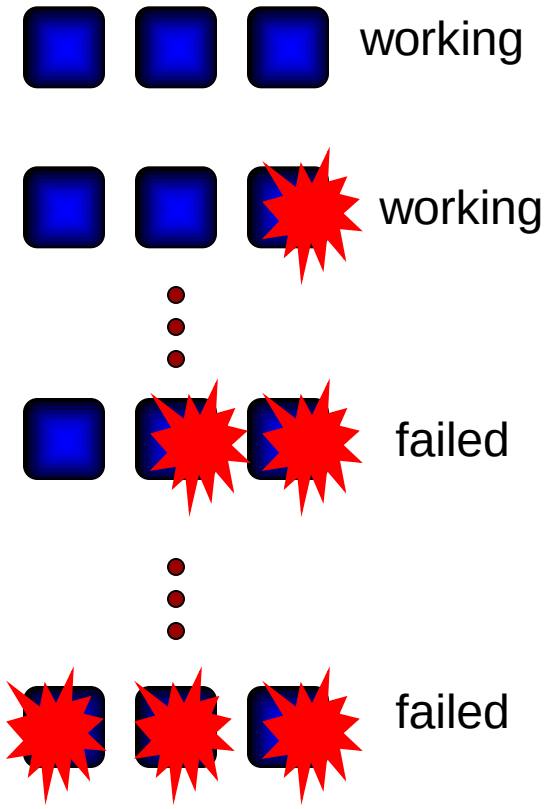
Some Measures of Interest

- The *expected performance* of the system at time ***t*** with the consideration of the effects of *failure, repair, contention for resources and so on*
- The *time-averaged performance* of the system over *an interval (0,t),* with the consideration of component failures and repairs
- The *probability* that *x* tasks are completed by time *t,* with the consideration of component failures and repairs

Example: TMR unit with simple Function-Based Performance Metric

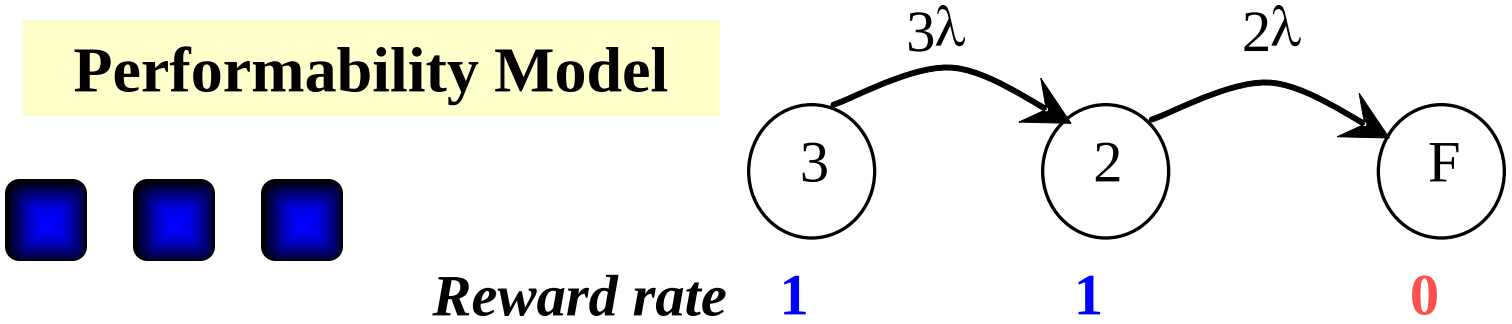
- Derive the Performability as a function of time
- The performance metric is considered only “working” or “failed”. (Reliability)





Example: Model Description

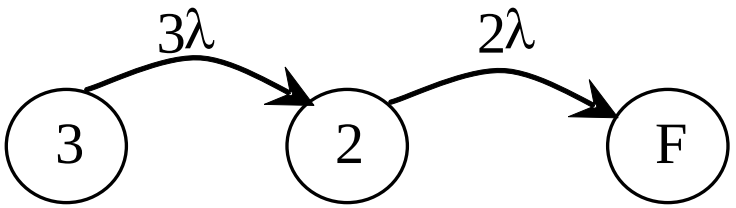
Performability Model



• At time t, a random variable X_t is used to describe the system's performance.
• Let X_t be "1" if at time t the unit is in state 3 or state 2; otherwise X_t is "0".

•Assume all three modules have the same failure rate λ, and their failure behavior can be described using the exponential distribution.

Example 2: Solutions



Reward rate **1** **1** **0**

State 2: 3 ways that 1 processor can fail
Pr(2 processors Ok, 1 failed) = $e^{-2\lambda t}(1- e^{-\lambda t})$

- At time *t*
  - probability being in state 3 is $e^{-3\lambda t}$,
  - probability of being in state 2 is $3[e^{-2\lambda t}(1- e^{-\lambda t})]$
- Performability : $1 \times [e^{-3\lambda t} + 3e^{-2\lambda t}(1- e^{-\lambda t})] = 3e^{-2\lambda t} - 2e^{-3\lambda t}$. (13)

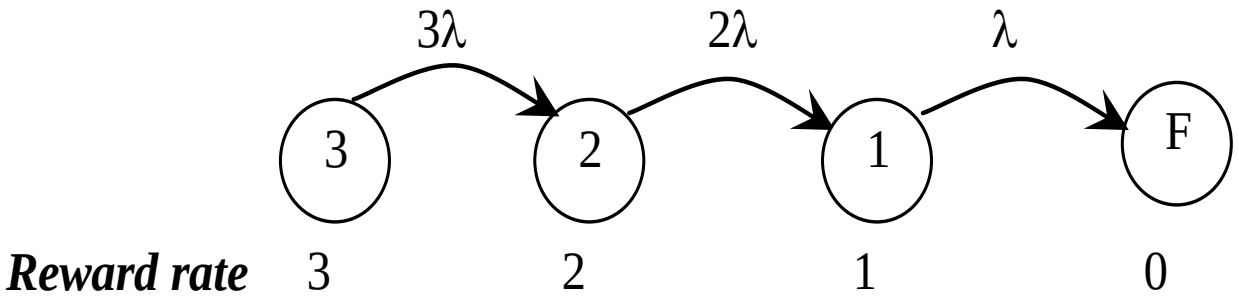
- This is equivalent to the reliability at time *t*

Example 2: TMR unit used as a 3-module Parallel Processing Unit

- 3 parallel processing units
  - conducting parallel tasks
  - no redundancy
  - if one of the processing units fails, the system degrades its performance
- Define the performance metric (reward rate) as the number of working processing units
- Random variable X_t represents the performance of this system.
  - X_t takes four discrete values: {3, 2, 1, 0}

Example 3: Model Description

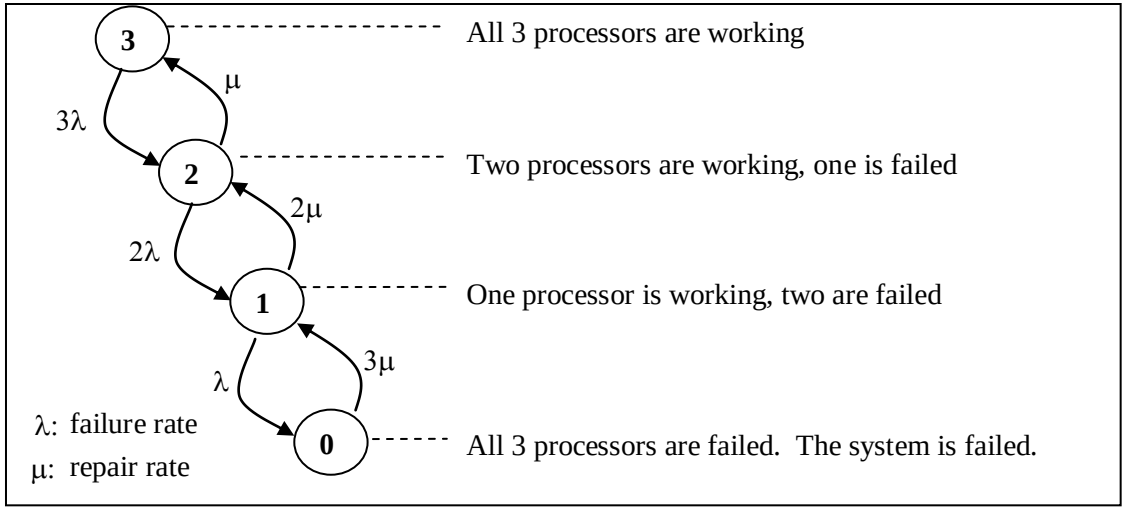
Performability Model for Example 3



Without repair

Dependability Sub-Model

With repair



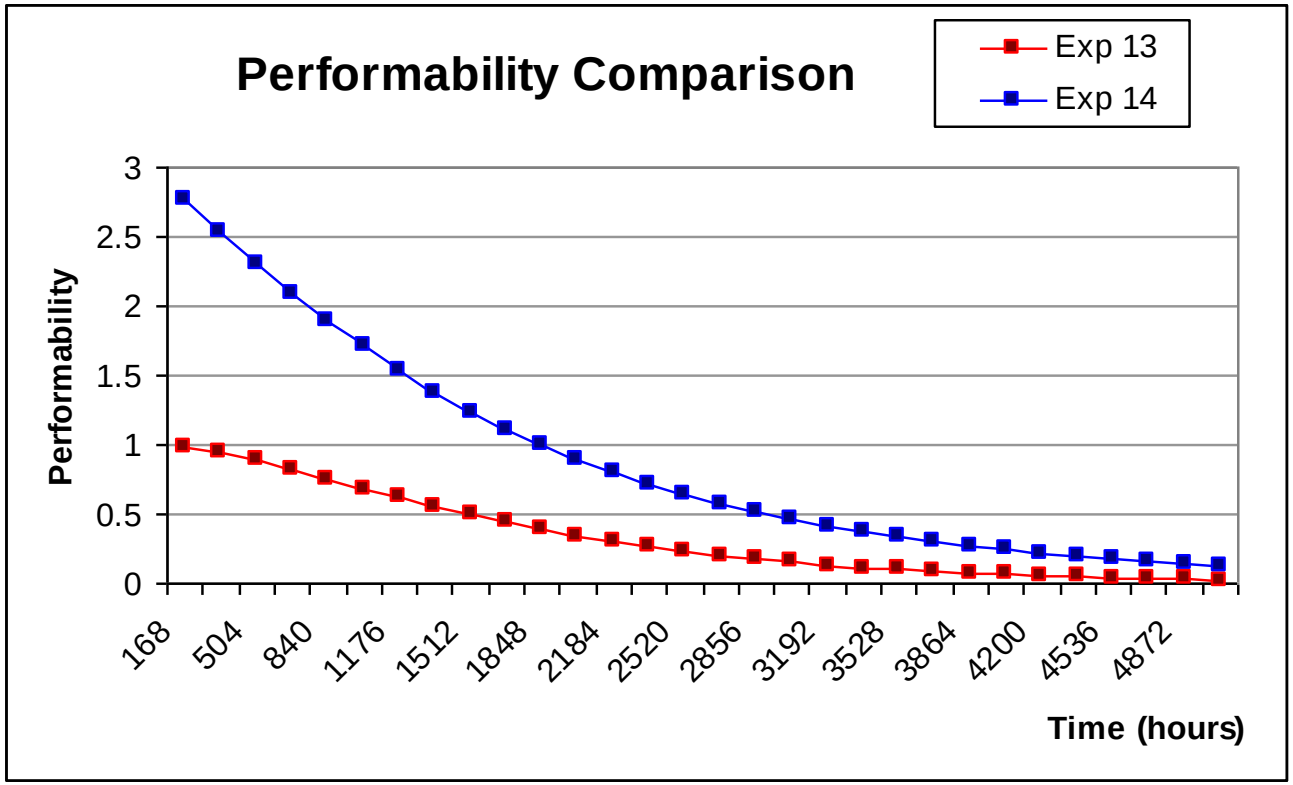
Example 3: Solution

The performability metric used here is the expected number of working processors at time *t*.

Performability is calculated as

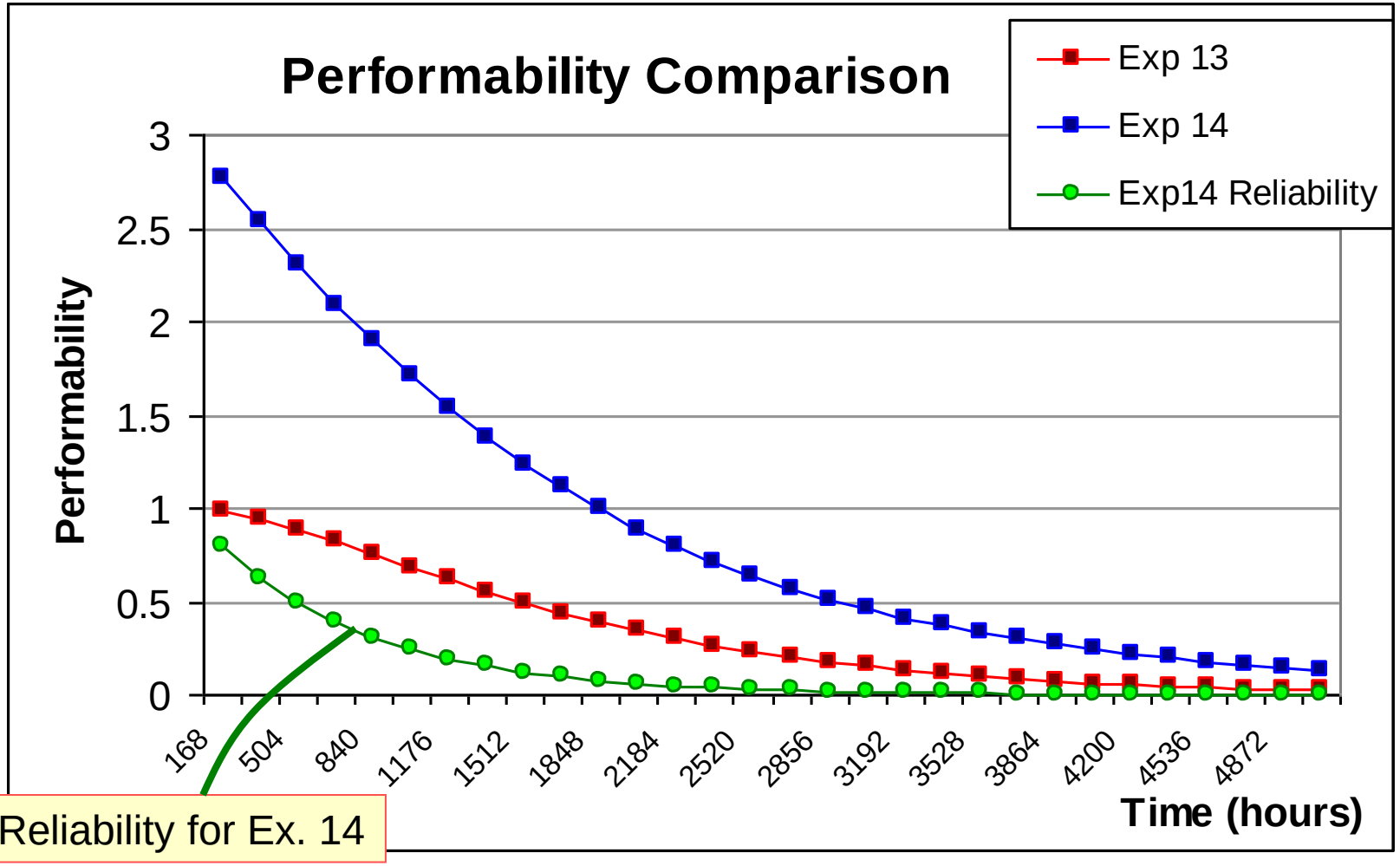
$$= 3 \times e^{-3\lambda t} + 2 \times 3e^{-2\lambda t}(1-e^{-\lambda t}) + 1 \times 3e^{-\lambda t}(1- e^{-\lambda t})^2 \quad (14)$$

Performability Comparison for Examples



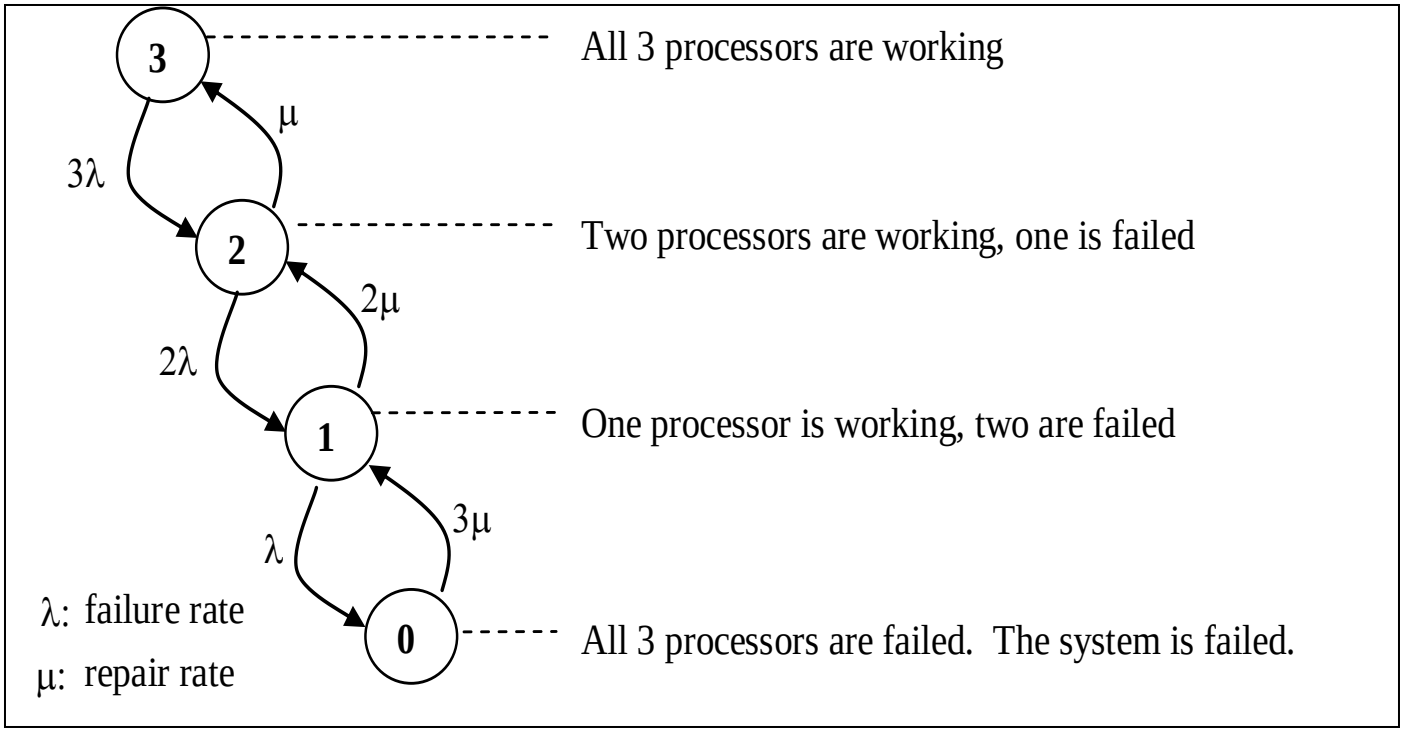
Recall that "performability has more performance flavor"

Reliability for Ex. 3

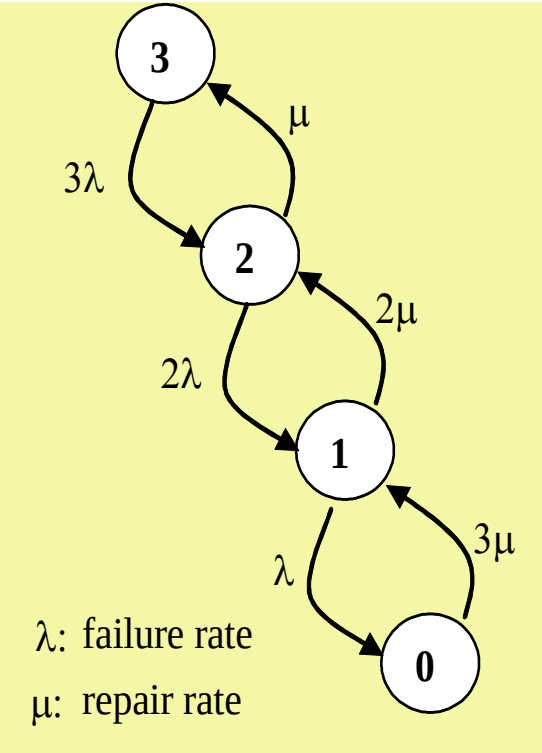


Example 4: A 3-processor system with repair and Queue-based reward

Dependability Sub-Model



Baseline Model Discussion



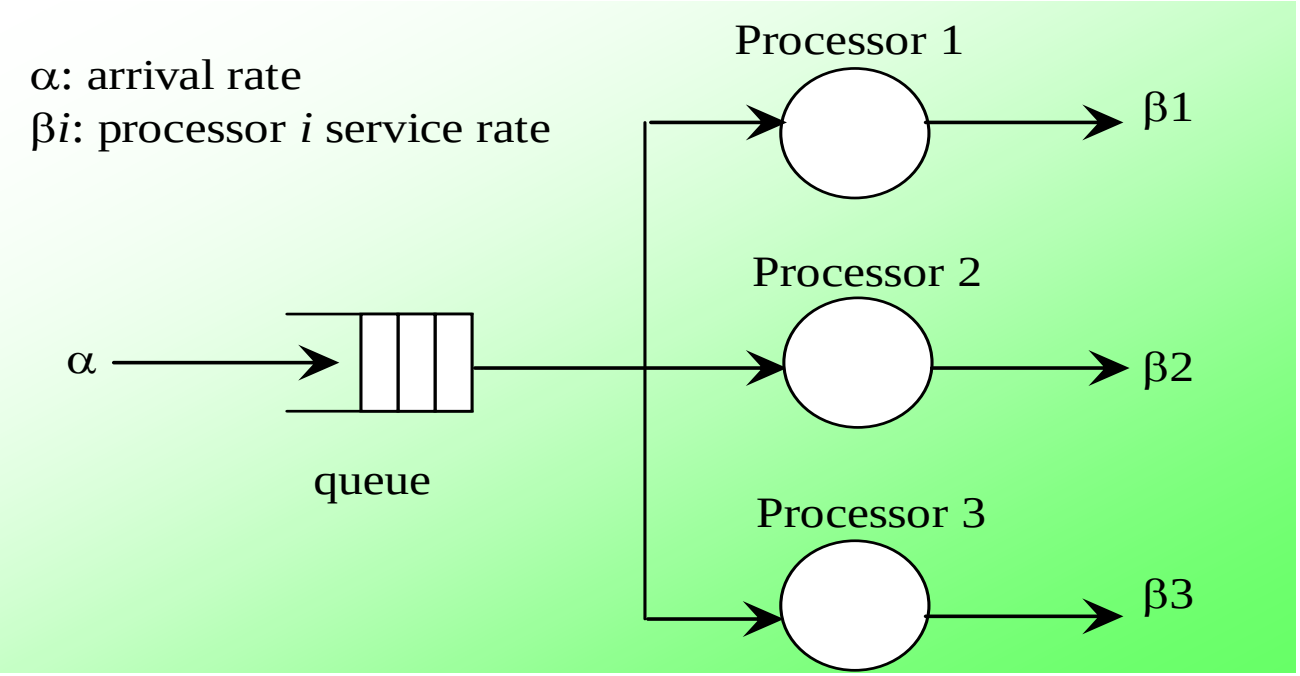
- At any time *t*, the system may be in one of the (*n+1*) states.
- For each state, there is a probability that the system will be in that state at time *t*.
- Markov chain techniques can be used to compute the state probabilities

Petri-net based models are often used to reduce the complexity in model construction. For instance, the *Stochastic Activity Networks* (SAN) model and the *Stochastic Reward Nets* (SRNs) model which are based on *Generalized Stochastic Petri Nets* (GSPN).

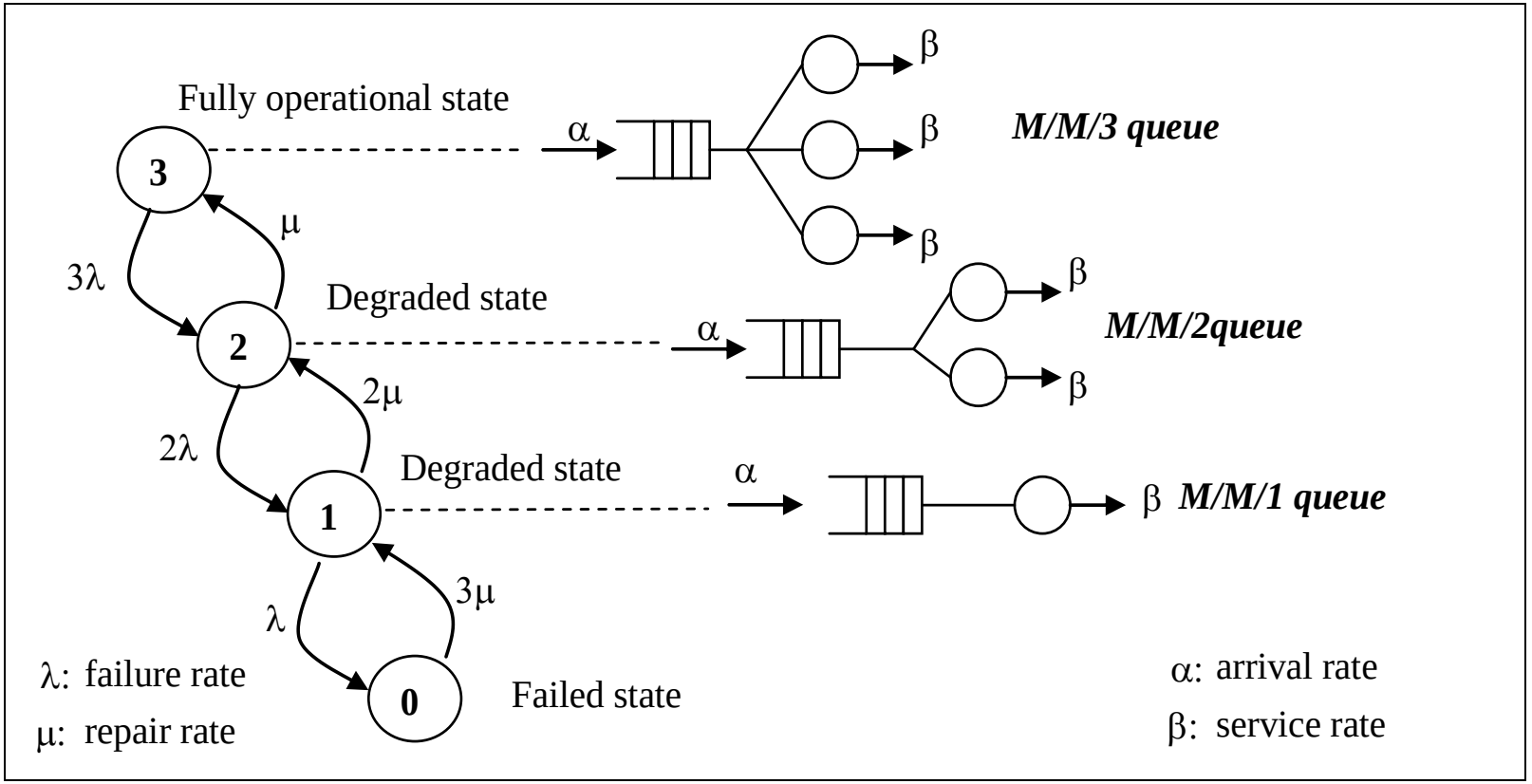
Performance Sub-Model

- At each state of the baseline model, a performance sub-model can be constructed to describe the system's performance behavior under that configuration.

Exp: the Performance Sub-Model for **state 3** in the baseline model



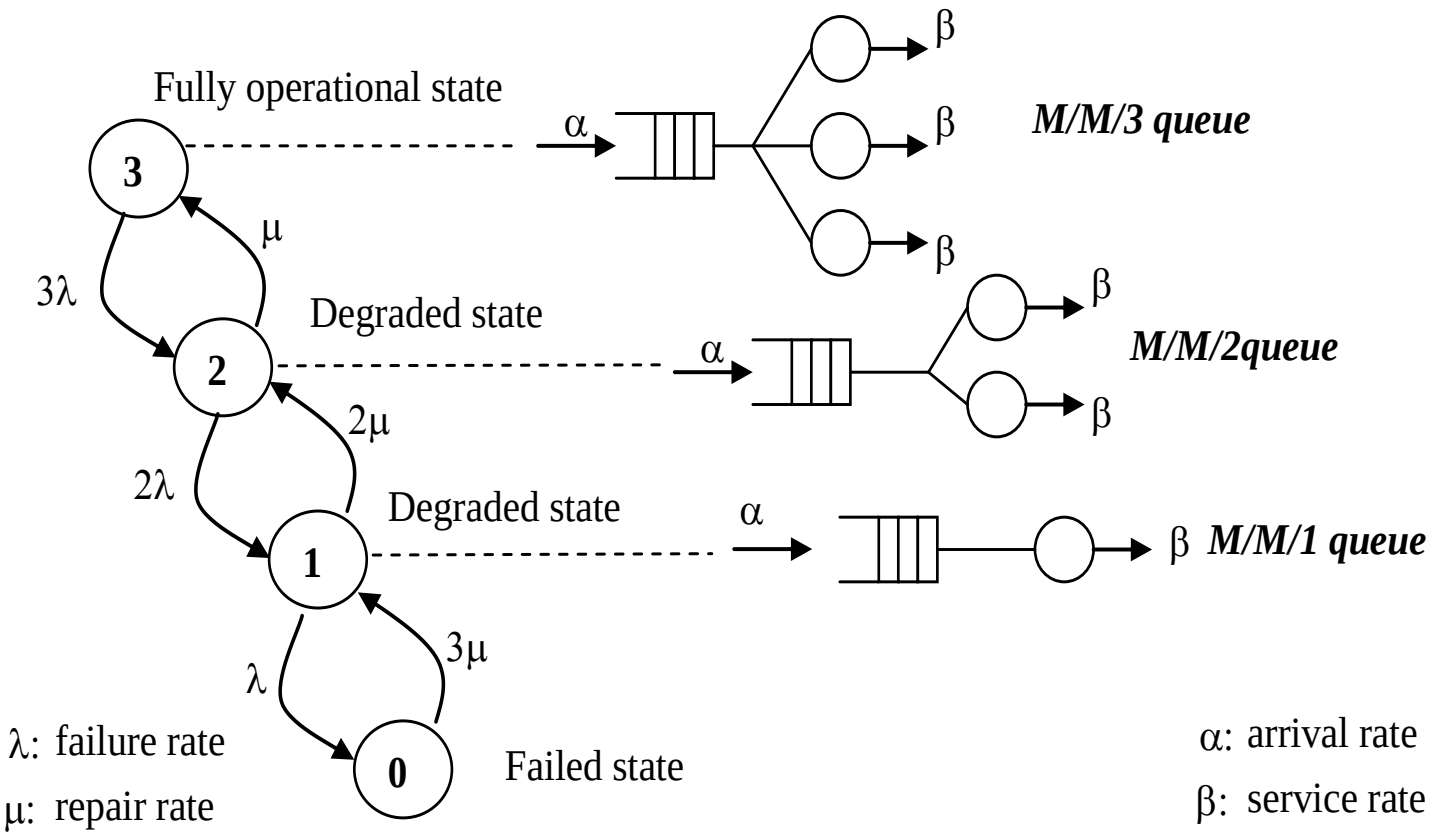
Example 4: The Sub-Models for the 3-Processor System



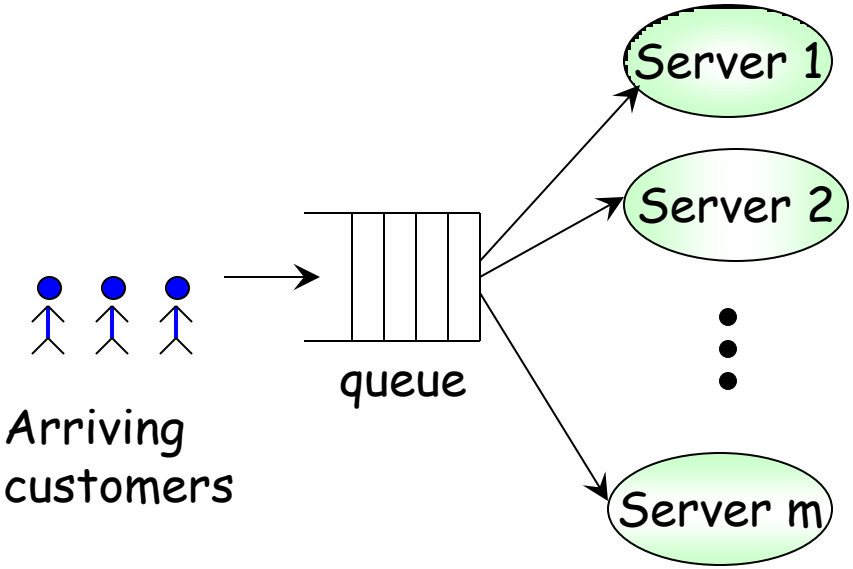
Example 5

- System Description:
  - same as the unit used in Example 4
  - performance degradable.
  - Repairable (independent repairs)
- What's different from Ex. 4?
  - Performance metric considered: probability that an incoming job will be waiting for service
- Approach:
  - Each state in the dependability sub-model is modeled with a unique model
    - state 3 : *M/M/3* queue
    - state 2 : *M/M/2* queue
    - state 1 : *M/M/1* queue
    - state 0 : the probability of waiting is always **1.**
  - The results have to be combined in order to obtain an overall performability assessment

Example 5: Sub-Models

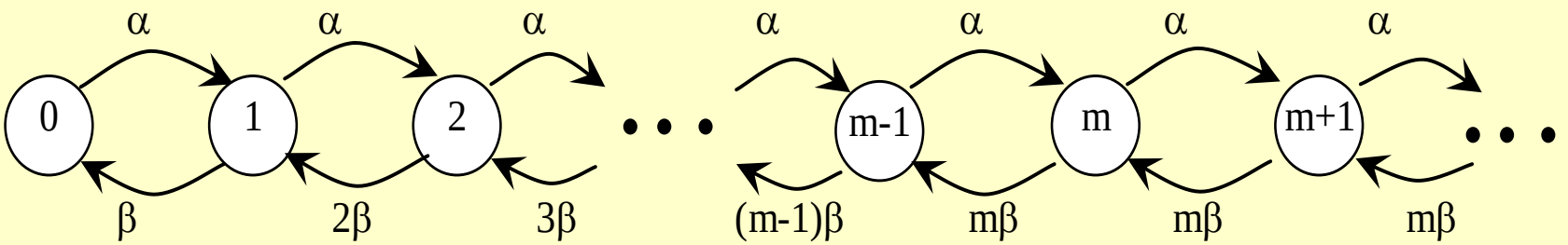


Solving M/M/m Queue

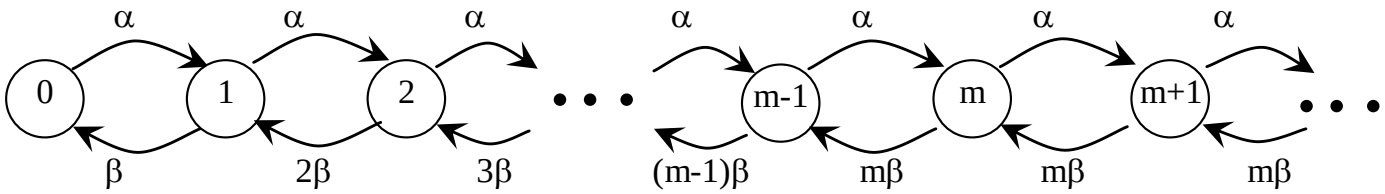


α is the job arrival rate
β is the service rate

The State Diagram Model (Markov chain) for an M/M/m queue



Solving M/M/m Queue (continued)



The probability that an arriving job will be waiting in the queue is the sum of the probabilities of being in states *m+1* and beyond in the above state diagram. This sum can be calculated as

$$\frac{(m\rho)^m}{m!} \bullet \frac{\pi}{1-\rho}$$

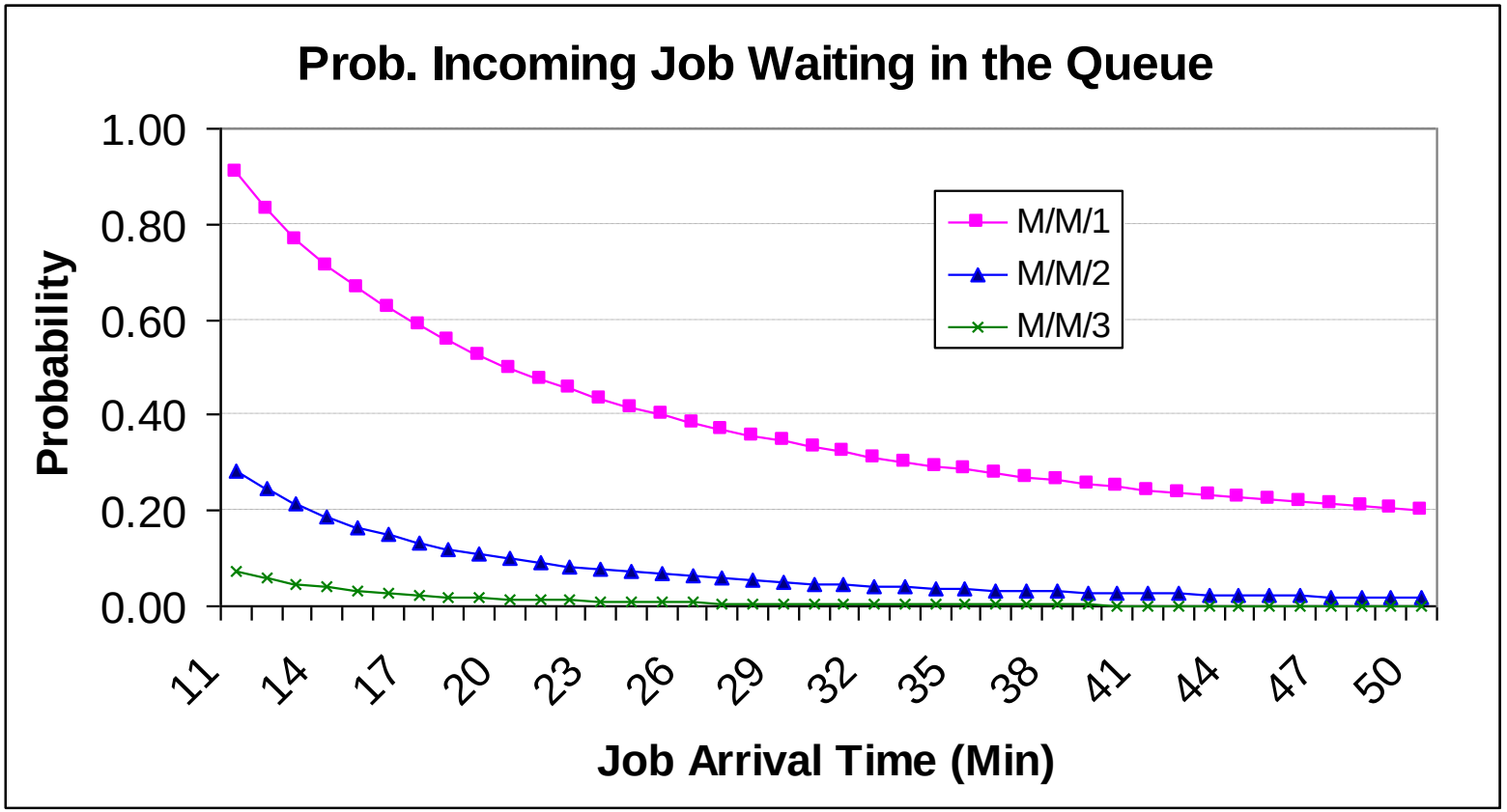
m: the number of working processors that are conducting different tasks

ρ : the utilization of any individual server, calculated as $\frac{\alpha}{m\beta}$

π : the probability of being in state *0*, calculated as $\left[\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!} \bullet \frac{1}{1-\rho}\right]^{-1}$

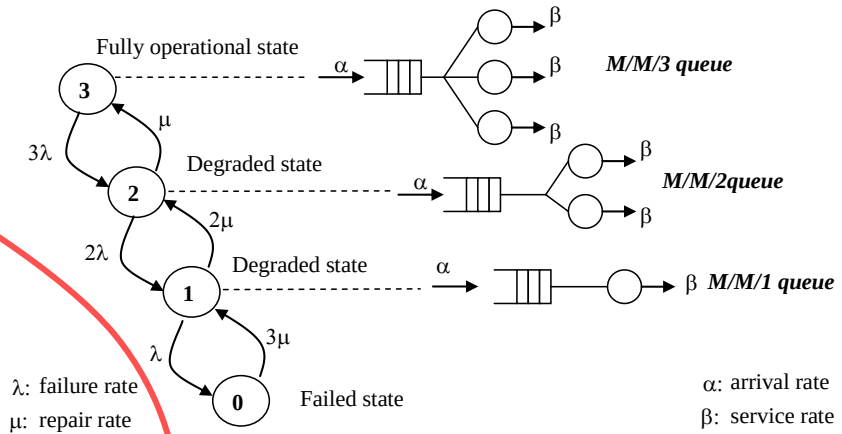
Performance for M/M/1, M/M/2 and M/M/3 Queues in Ex. 5

The mean service time is assumed to be 10 minutes in all cases, and the mean job arrival time varies from 11 minutes to 50 minutes.



Combination Method

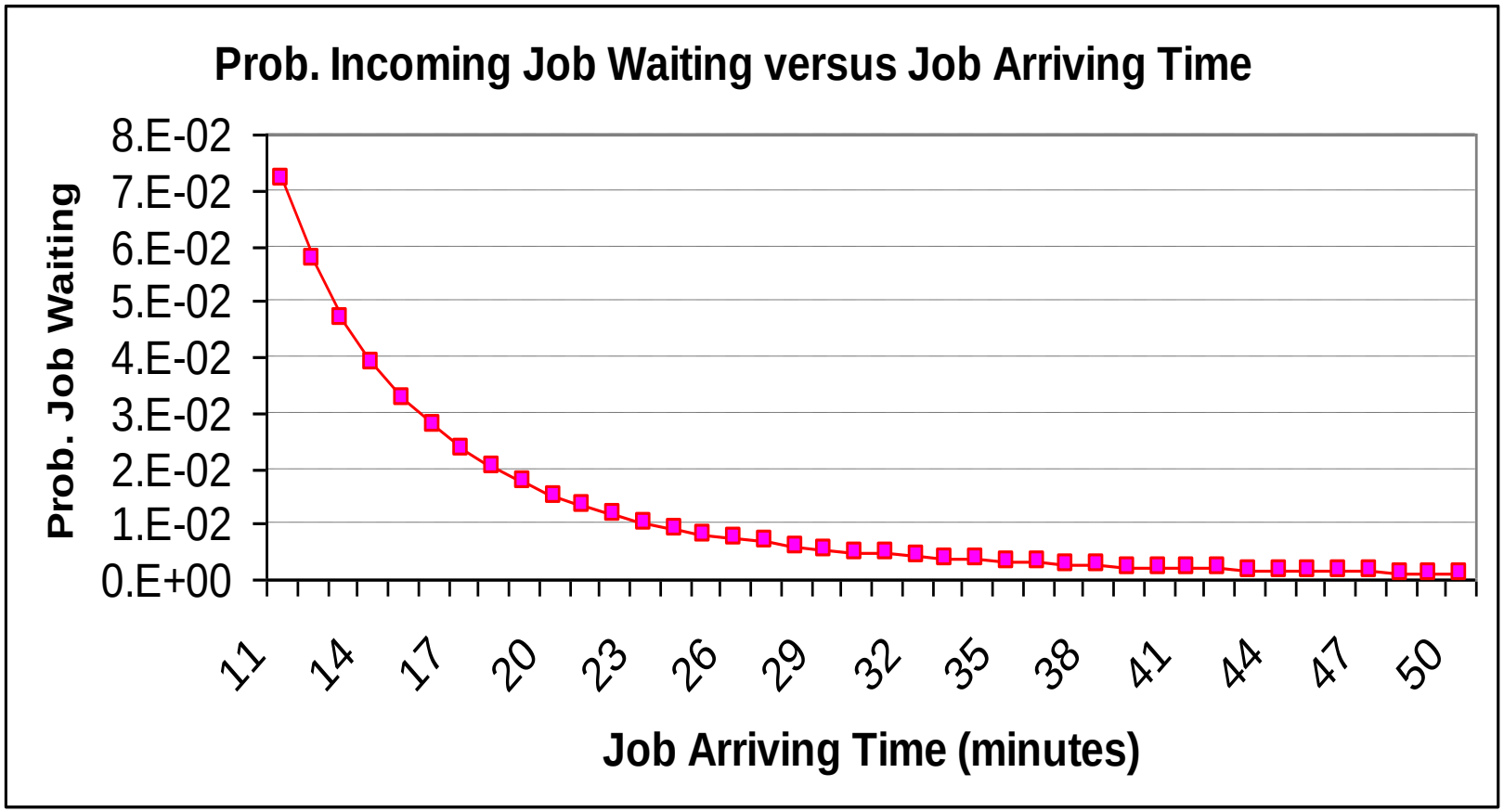
- We have:
  - the dependability sub-model
  - the performance sub-models (*M/M/1, M/M/2 and M/M/3*)
- We know how:
  - to solve the dependability model
  - to solve the performance sub-models



Performability metric: the expected value of the probability that an incoming job will be waiting for service.

$$\sum_{i=0}^{3} P_{queuing i} \times pi$$

Performability for Ex.5



Discussion on the Modeling Approach

Why do we use the "three steps" approach?
(i.e., a dependability model, a performance model and a combing method,)

You might have been wondering, why don't we **construct a single model** with both dependability and performance events in it?

Stiffness Problem

One difficulty of conducting a single model is the ***stiffness*** problem

The Problem

The order of magnitude **differences** between the transition rates in dependability models and the rates in performance models:

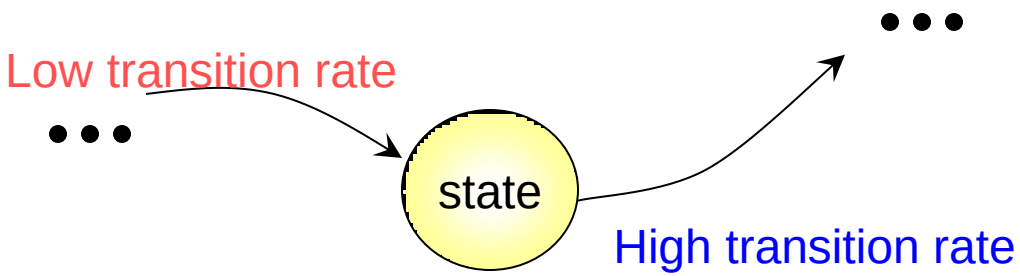
– Transition rates in the dependability models are much smaller than the rates in the performance models

– Usually the mean time to failure is at least in the order of 10^3 hours (failure rate $\propto 10^{-5}$ s^{-1})

– The job arrival time or response time considered in the performance models is normally in minutes or seconds.

The Consequences of the Stiffness Problem

- It can cause serious problems in obtaining solutions.
- Even if solutions can be obtained, the probability of being in a state where the outgoing transition rates are much higher than that of the incoming transition rates, would be too small to make any impact.



It is this stiffness problem that makes the separation of the two sub-models necessary.

Problem-Growth Issue

Tradeoff between Fidelity and Complexity

- Modeling systems with great detail will end up with huge state space
- Complexity in model construction and solutions

Although the computer processing power has been increased greatly due to Moore's law, the complexity of a modern system has also increased exponentially, which **ensures the occurrence of the model-growth problem.**

Dealing with the Complexity

- The decomposition approach (introduced before)

- There are ways to deal with the largeness problem in model construction, using well-developed modeling techniques, for example, SRN (Stochastic Reward Nets) and SAN (Stochastic Activity Networks).

Outline

Concepts

The meaning of performability

- Basics
- Structure of a Performability Model
- A Case Study

Mechanisms

How to combine performance and dependability

- Examples
- Stiffness & Largeness Problems
- **Formal Definitions**

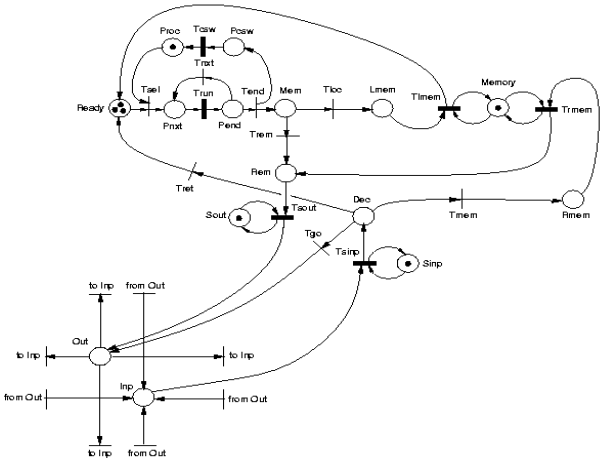
Applying Performability

Considering performability in system design & development

- Process
- Real Issues

Formalizing Performability

- If performability is treated as a stochastic process with time, then at each instance of time t, a random variable P_t is used to represent the performability of the system at time t.
- That is, a stochastic process defined as $P = \{P_t \mid t \in T\}$ is used to represent the performability of a system
- Depending on the application, T can be either discrete or continuous.



Settings

Dependability Part

- *S*: the set of all possible configurations that the target system can operate.
- The dependability model:
  - defines a stochastic process on **S**, describing the configuration of the system at time *t*.
  - the probability of the system being in a particular state (configuration) at time *t* is determinable.

Performance Part

- *R:* the overall performance of the considered system
  - discrete random variable
  - continuous random variable.

Conclusions

Summary

- introduced performability: a useful metric for performance-degradable systems
- showed many trivial examples to demonstrate performability analysis, including dependability models and performance models

In the future, more and more performance degradable systems will appear