CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 18)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 17

Periodicity Properties of Markov Chain

Example Problems

Counting Processes (Poisson Process as a Counting Process)

Merging and Splitting of Poisson Processes

Chapter # 6

FUNDAMENTALS OF QUEUING MODELS

Prepared by: Ms. Fakhra Aftab (Lecturer, CISD, NEDUET)

What is Queuing Theory?

- *Queuing theory* is the study of waiting in various guises.

- It deals with quantifying the phenomenon of waiting in lines using representative measures of performance, such as
  - *average queue length*,
  - *average waiting time* in queue, and
  - *average service time*.

- It uses ***queuing models*** to represent the various types of ***queuing systems*** that arise in practice.

What is Queuing Theory?

- **Advantages of queuing models**: very helpful for determining how to operate a queuing system in the most effective way.

- Providing too much service capacity to operate the system involves excessive costs.

- But not providing enough service capacity results in excessive waiting and all its unfortunate consequences.

- The operating characteristics of queuing systems are determined largely by two statistical properties, namely,
  - the probability distribution of *inter-arrival times* and
  - the probability distribution of *service times*.

The Basic Queuing Process

- *Customers* requiring service are generated over time by an ***input source***.
- These customers enter the ***queuing system*** and join a ***queue***.
- At certain times, a member of the queue is selected for service by some rule known as the ***queue discipline***.
- The required service is then performed for the customer by the ***service mechanism*** after which the customer leaves the queuing system.

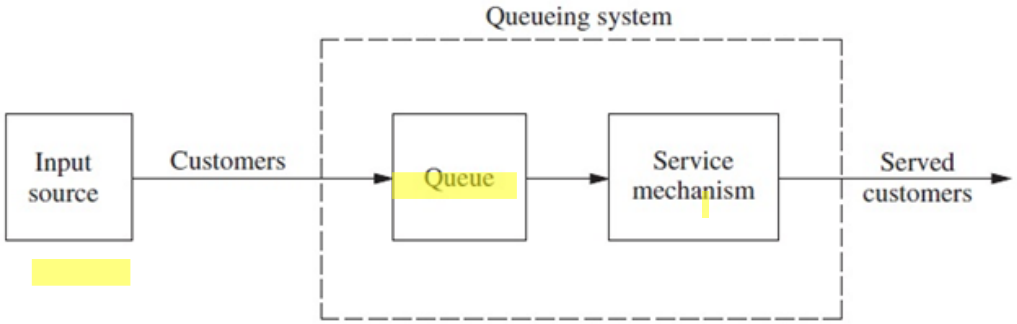


Fig 1: The basic queuing process

Input Source (Calling Population)

- **Calling population**: population from which arrivals come.

- One characteristic of the input source is its size.
  - total number of distinct potential customers that might require service from time to time.

- The size may be assumed to be either ***infinite*** or ***finite***.

- An infinite source is forever abundant (e.g., calls arriving at a telephone exchange).

- Because the calculations are ***far easier*** for the infinite case, this assumption often is made even when the actual size is some relatively large finite number.

Input Source (Calling Population)

- The finite case is ***more difficult analytically.***

- The statistical pattern by which customers are generated over time must also be specified.

- **The common assumption**: ***Poisson process;*** i.e.,
  - the number of customers generated until any specific time has a Poisson distribution.
  - arrivals to the queuing system occur randomly but at a certain fixed mean rate, regardless of how many customers already are there
  - so the *size* of the input source is *infinite*.

- **An equivalent assumption**: the probability distribution of the time between consecutive arrivals is an ***exponential distribution***.

Queuing Behavior

- The queuing behavior of customers plays a role in waiting-line analysis.
- Human customers may *jockey* from one queue to another in the hope of reducing waiting time.
- They may also *balk* from joining a queue altogether because of anticipated long delay, or
- They may *renege* from a queue because they have been waiting too long.

Queue

- The queue is where customers wait *before* being served.
- A queue is characterized by the maximum permissible number of customers that it can contain.
- Queues are called *infinite* or *finite,* according to whether this number is infinite or finite.
- The assumption of an *infinite queue* is the standard one for most queuing models.
- However, for queuing systems where this upper bound is small enough that it actually would be reached with some frequency, it becomes necessary to assume a *finite queue*.

Queue Discipline

The order in which members are selected from a queue

- An important factor in the analysis of queuing models.
- First-come-first-served (FCFS) usually is assumed by queuing models, unless stated otherwise.
- Other disciplines include Last Come, First Served (LCFS) and Service In Random Order (SIRO).
- Customers may also be selected from the queue based on some order of priority.
- For example, rush jobs at a shop are processed ahead of regular jobs.

Service Mechanism

- The service mechanism consists of
  - one or more ***service facilities***,
  - each of which contains one or more ***parallel service channels***, called **servers**.
- If there is more than one service facility, the customer may receive service from a sequence of these (*service channels in series*).
- At a given facility, the customer enters one of the parallel service channels and is completely serviced by that server.
- A queuing model must specify the arrangement of the facilities and the number of servers (parallel channels) at each one.
- Most elementary models assume one service facility with either one server or a finite number of servers.

Service Mechanism

- **Service time (or *holding time*)**: The time elapsed from the commencement of service to its completion for a customer at a service facility.

- The service-time distribution that is most frequently assumed in practice
  - (largely because it is far more tractable than any other)
  - is the *exponential* distribution, and most of our models will be of this type.

- Other important service-time distributions are
  - the ***degenerate*** distribution (constant service time) and
  - the ***Erlang (gamma)*** distribution.

Cost-based Queuing Decision Model

➢ **Cost optimization model**: we seek the minimization of the sum of the two costs:-

- the cost of offering the service and the
- cost of waiting.

- Fig 2 depicts a typical cost model (in Rs per unit time).

- **The main obstacle**: difficulty of obtaining reliable estimates of the cost of waiting, particularly when human behavior is involved.

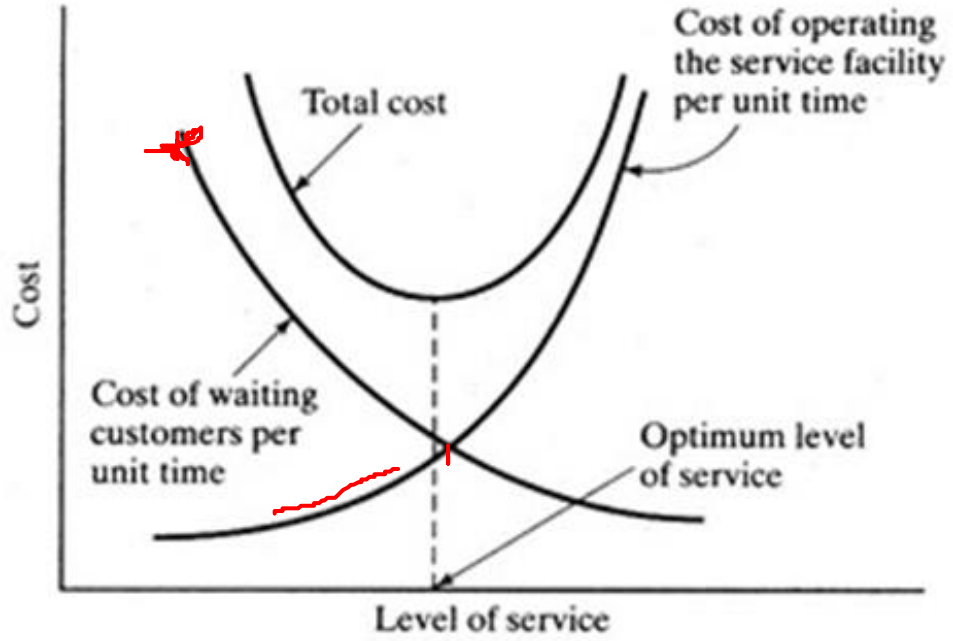


Fig 2

An Elementary Queuing Process

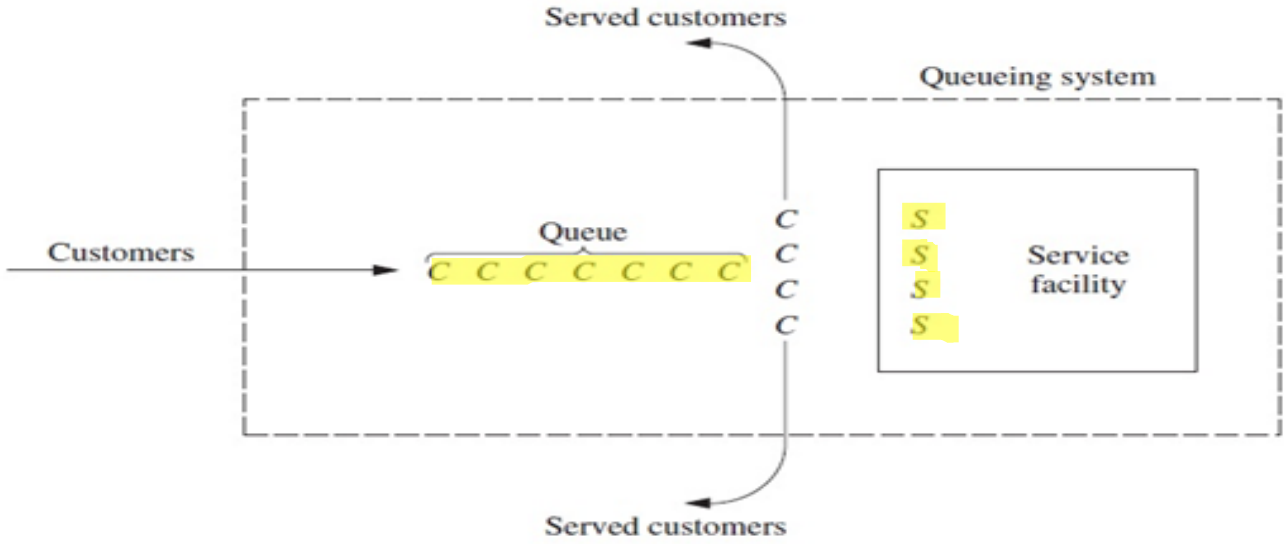
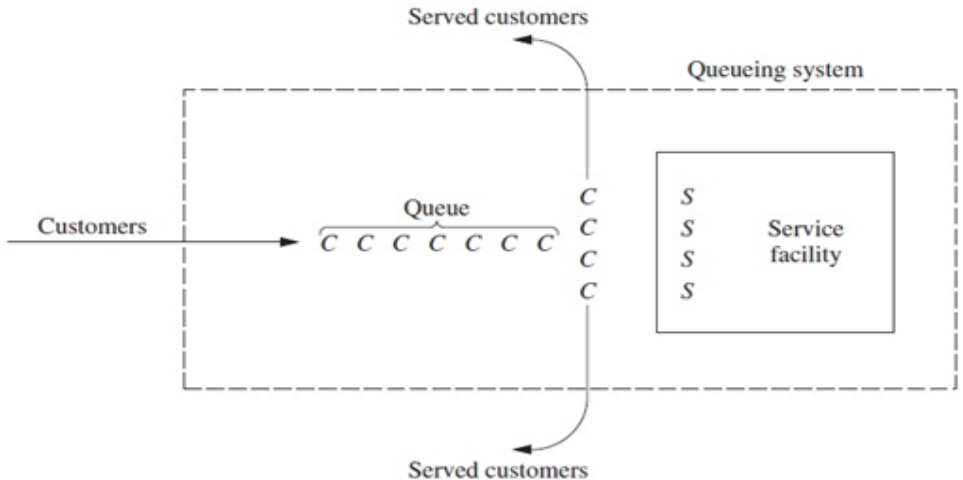


Fig 3: An elementary Queuing System

- Queuing theory has been applied to many different types of waiting-line situations.
- The most prevalent type of situation is the following:
  - A single waiting line (which may be empty at times) forms in the front of a single service facility, within which are stationed one or more servers.
  - Each customer generated by an input source is serviced by one of the servers, perhaps after some waiting in the queue (waiting line).

An Elementary Queuing Process

- Furthermore, servers need not even be people.
- In many cases, a server can instead be a machine, a vehicle, an electronic device, etc.
- By the same token, the customers in the waiting line need not be people.
- For example, they may be items waiting for a certain operation by a given type of machine, or they may be cars waiting in front of a tollbooth.



Examples

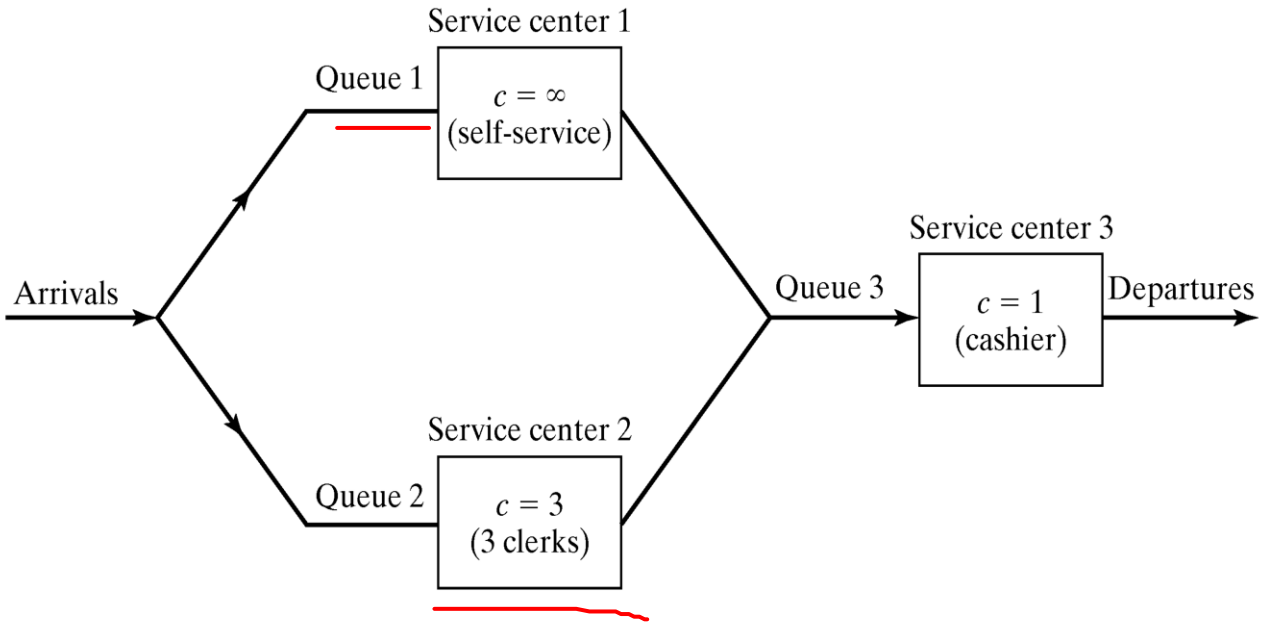
System	Customers	Server
Reception desk	People	Receptionist
Hospital	Patients	Nurses
Airport	Airplanes	Runway
Production line	Cases	Case-packer
Road network	Cars	Traffic light
Grocery	Shoppers	Checkout station
Computer	Jobs	CPU, disk, CD
Network	Packets	Router

Example Problem

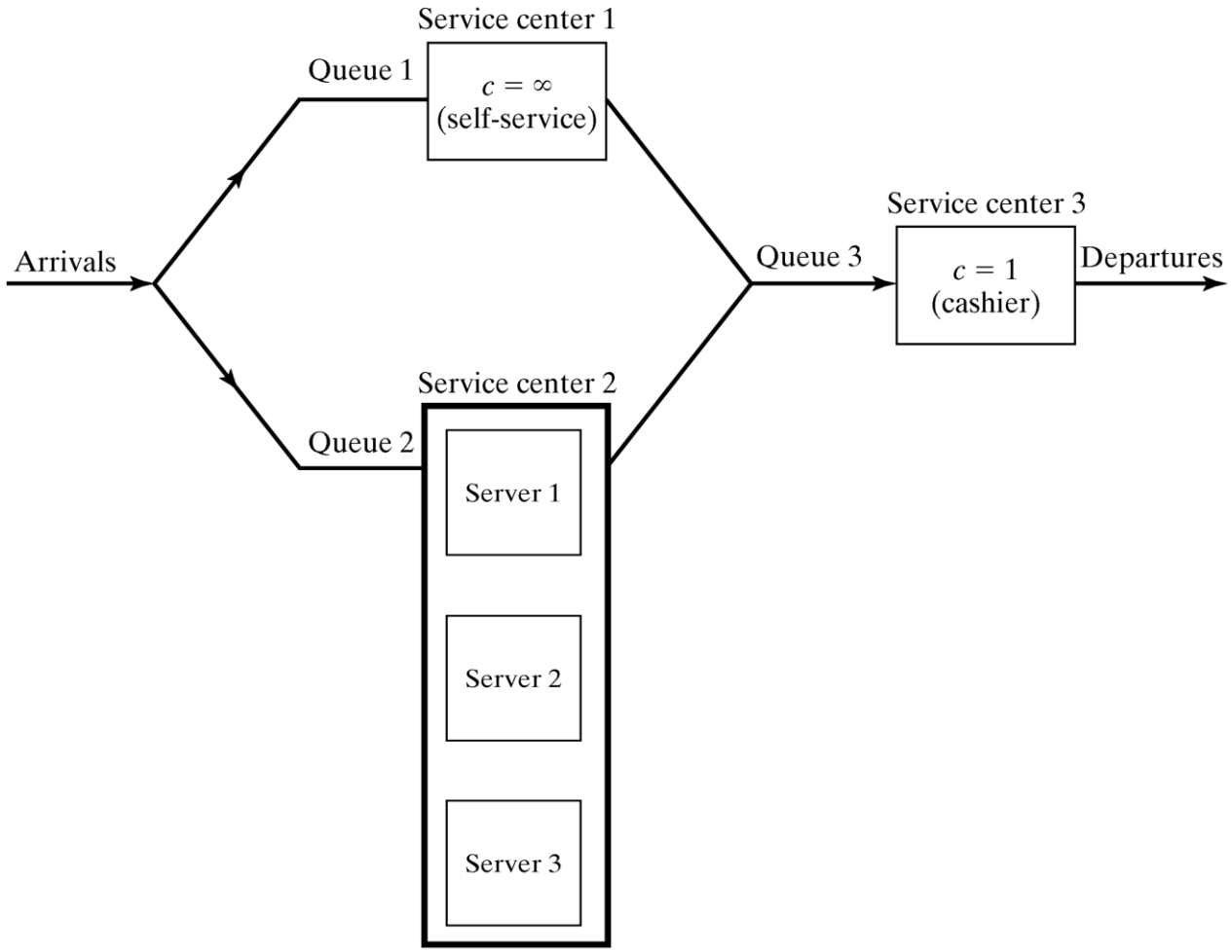
Consider a discount warehouse where customers may:

- serve themselves before paying at the cashier (service center 1) or
- served by a clerk (service center 2)

Solution:



Solution:


Service center 1
Queue 1
$c = \infty$
(self-service)
Arrivals
Service center 3
Queue 3
$c = 1$
(cashier)
Departures
Service center 2
Queue 2
Server 1
Server 2
Server 3

CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 19)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 18

Queuing Theory - Definition

Basic Queuing Process

Input Source, Queue Behavior, Queue Discipline and Service Mechanism

An Elementary Queuing Process

Chapter # 6 (Cont'd)

FUNDAMENTALS OF QUEUING MODELS

Kendall Notation

Notation: A/B/C/X/Y/Z

A	Inter-arrival Time Distribution
B	Service Time Distribution
C	No. of Servers
X	System Capacity (In queue + in server)
Y	Population Size
Z	Service Discipline

A and/or B could be:

M	Markovian/Exponential
E_k	Erlang with parameter k
G	General (Sometimes GI is used rather than G)
D	Deterministic
H_k	Hyper-exponential with parameter k

- A, B and C are always provided. Default values of X, Y and Z are ∞, ∞ and FIFO.
- Scheduling Policies could be pre-emptive or non-preemptive. Examples are FCFS, SIRO, RR, IS (Infinite Server) and Priority-based.

Kendall Notation Examples

1) The notation **M/D/2/∞/∞/FCFS** indicates a queuing process

- with exponential interarrival times, deterministic service times,
- two parallel servers, no restriction on the maximum number allowed in the system and FCFS queue discipline.

- In many situation, only the first three symbols are used.

2) Thus, **M/D/2** would be a queuing system with exponential input, deterministic service, two servers, no limit on system capacity and FIFO discipline.

3) The term $M^{[x]}/M/1$ denotes a single server queue with bulk Poisson arrivals and exponential service times.

Practice Questions

Explain the meanings of given Kendal's Notations:

- M/M/3/20/1500/FCFS
- $E_k/G^{[x]}/5/300/5000/LCFS$
- $M/G/m$

Basic Assumptions

- ***Job flow balance*** - The number of jobs arriving at a server within a sufficiently large observation interval must be equal to the number of jobs that depart from the server.
- ***One-step behaviour*** - At any instant of time, only a single job can enter or leave a server so that the system's state changes only incrementally.
- ***Homogeneity*** - Homogeneity means that the average job-arrival rate and the average service rate are independent of the system's state.
- ***Exclusivity*** - A job can be present in only a single server, either waiting in the queue or receiving service. Thus, a single job cannot make a request of two servers simultaneously.
- ***Non-blocking*** - The service provided by a server cannot be controlled by any other device in the system.
- ***Independence*** - Jobs are not allowed to interact in any way, except for sharing space in a queue.

Terminology and Notation

- Unless otherwise noted, the following standard terminology and notation will be used:

1. **State of system** = number of customers in queuing system.

2. **Queue length** = number of customers waiting for service to begin = state of system *minus* number of customers being served.

3. $N(t)$ = number of customers in queuing system at time t ($t \geq 0$).

4. $P_n(t)$ = probability of exactly n customers in queuing system at time t, given number at time 0.

5. s = number of servers (parallel service channels) in queuing system.

Terminology and Notation

6. λ = mean arrival rate (expected number of arrivals per unit time)
7. μ = mean service rate for overall system (expected number of customers completing service per unit time).

- Under these circumstances, $1/\lambda$ and $1/\mu$ are the *expected inter-arrival time* and the *expected service time,* respectively.

Little's Law

- Most widely used formula in queuing theory is:

$$L = \lambda W$$

- (Because John D. C. Little provided the first rigorous proof, this equation sometimes is referred to as **Little's formula**)
- Furthermore, the same proof also shows that

$$L_q = \lambda W_q$$

where

L = Avg. no of customers in the system; L = E[N]

W = Avg. time spent in the system; W = E[W]

Let,

a (t): no. of arrivals by time t, therefore

$$\lambda_t = \frac{a\,(t)}{t}$$

Little's Law (Cont'd)

Let,

g (t): total time spent by all the customers by time t, therefore

$$W_t = \frac{g(t)}{a(t)}$$

$$L_t = \frac{g(t)}{t} = \frac{g(t)}{a(t)} \times \frac{a(t)}{t}$$

$$L_t = \lambda_t W_t$$

- Can be applied to any block of queuing system.

$$L_q = \lambda W_q$$

$$L_s = \lambda W_s$$

Utilization Law

- $\rho = \lambda/(s\mu)$ is the **utilization factor** for the service facility, i.e.,
  - the expected fraction of time the individual servers are busy,
  - because $1/(s\mu)$ represents the fraction of the system's service capacity ($s\mu$) that is being *utilized* on the average by arriving customers (λ).
  - With a single server, s = 1 and $\rho = \lambda/\mu$
  - **For a stable queuing system, we must have $\lambda < \mu$ i.e. $\rho < 1$**

Types of Queuing Analysis

Operational Analysis

- Operational analysis views the system being studied as a black box in which jobs arrive at one end, are processed for some period.
- Queueing models are studied using simple equations while making no assumptions about the probability distribution of the times between arrivals of jobs and the times required to service these jobs.
- Apply some simple laws to determine the system's overall, or average behaviour.
- Examples: Little's or Utilization law

Stochastic Analysis

- If the times between the arrivals of new jobs, and the times required to service these jobs, follow certain probabilistic stochastic distributions, then detailed insight of the given system is possible.
- Jobs enter the system at times determined by the arrival process. If a server is available, the job can be serviced immediately. Otherwise, it must wait in the queue until one of the jobs currently being serviced completes. The time required to service each job also follows some assumed stochastic distribution.

Thank You!!

CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020
Batch: 2016-17
(LECTURE # 20)

FAKHRA AFTAB
LECTURER
DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 19

Kendall's Notation

Little's Law and Utilization Law

Operational and Stochastic Analysis

Chapter # 6 (Cont'd)

FUNDAMENTALS OF QUEUING MODELS

Transient State & Steady State

- When a queuing system has recently begun operation,
  - the state of the system (number of customers in the system) will be greatly affected by the initial state and by the time that has since elapsed.
- The system is said to be in a **transient condition.**
- However, after sufficient time has elapsed, the state of the system becomes essentially independent of the initial state and the elapsed time (except under unusual circumstances).
- The system has now essentially reached a **steady-state condition,** where the probability distribution of the state of the system remains the same (the *steady-state* or *stationary* distribution) over time.
- Queuing theory has tended to focus largely on the steady-state condition.

Steady State Notations

The following notation assumes that the system is in a *steady-state condition:*

P_n = probability of exactly n customers in queueing system.

L = expected number of customers in queueing system = $\sum_{n=0}^{\infty} nP_n$.

L_q = expected queue length (excludes customers being served) = $\sum_{n=s} (n - s)P_n$.

$\mathcal{W}$ = waiting time in system (includes service time) for each individual customer.

$W = E(\mathcal{W})$.

$\mathcal{W}_q$ = waiting time in queue (excludes service time) for each individual customer.

$W_q = E(\mathcal{W}_q)$.

M/M/1 QUEUE ANALYSIS

THE BIRTH-AND-DEATH PROCESS

- Most elementary queuing models assume that the inputs (arriving customers) and outputs (leaving customers) of the queuing system occur according to the *birth-and-death process*.

- The *state* of the system at time t ($t \geq 0$), denoted by $N(t)$,
  - number of customers in the queuing system at time t.

- The birth-and-death process describes *probabilistically* how $N(t)$ changes as t increases.

- Broadly speaking, it says that *individual* births and deaths occur *randomly*, where their mean occurrence rates depend only upon the current state of the system.

THE BIRTH-AND-DEATH PROCESS (Cont'd)

- The next transition in the state of the process is either
  - $n \rightarrow n + 1$ (a single birth)

    or
  - $n \rightarrow n - 1$ (a single death),
  - depending on whether the former or latter random variable is smaller.

- Because of these assumptions, the birth-and-death process is a special type of *continuous time Markov chain*.

- Queuing models that can be represented by a continuous time Markov chain are far more tractable analytically than any other.

THE BIRTH-AND-DEATH PROCESS (Cont'd)

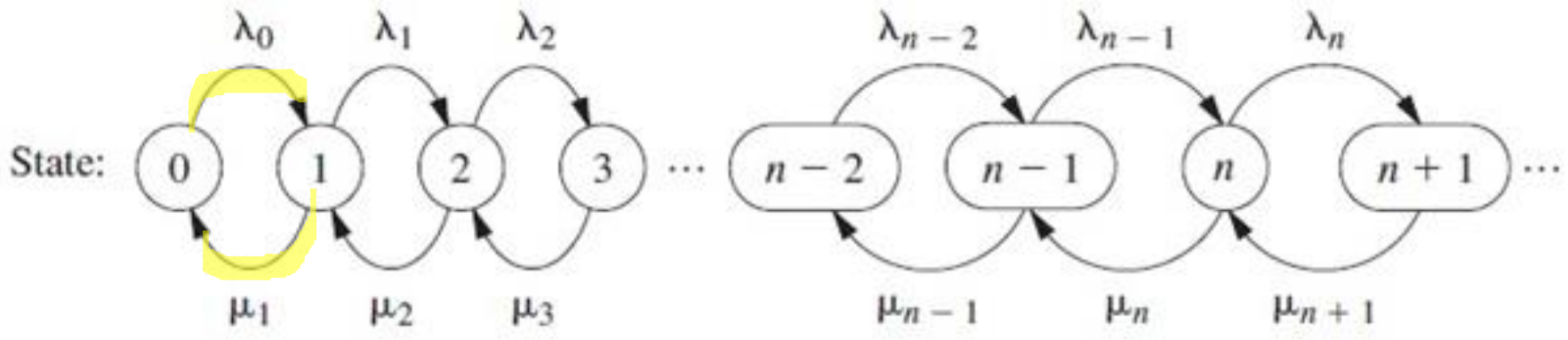


Fig 1: Rate diagram for the birth-and-death process

- The arrows in this diagram show the only possible *transitions* in the state of the system.
- The entry for each arrow gives the mean rate for that transition.
- These results yield the following key principle:

Rate In = Rate Out Principle

THE BIRTH-AND-DEATH PROCESS (Cont'd)

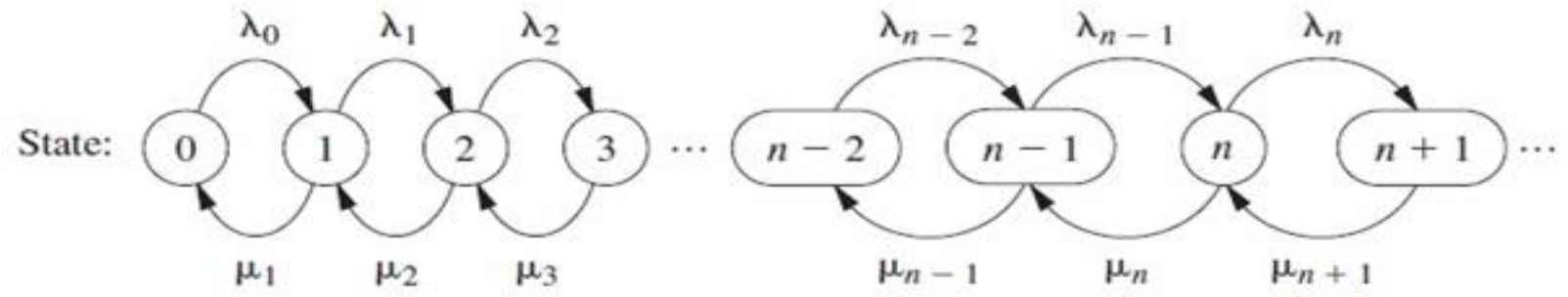


Fig 1: Rate diagram for the birth-and-death process

- For any state of the system n ($n = 0, 1, 2, \ldots$),
  - mean entering rate = mean leaving rate.
- The equation expressing this principle is called the **balance equation** for state n.
- After constructing the balance equations for all the states in terms of the *unknown* P_n probabilities,
  - we can solve this system of equations (plus an equation stating that the probabilities must sum to 1) to find these probabilities.

THE BIRTH-AND-DEATH PROCESS (Cont'd)

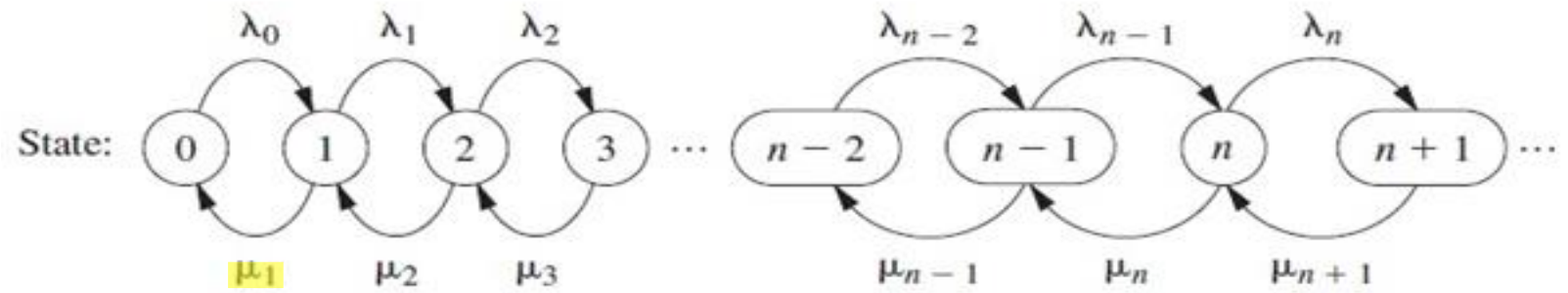


Fig 1: Rate diagram for the birth-and-death process

- To illustrate a balance equation, consider state 0. The process enters this state *only* from state 1.
- Given that the process is in state 1, the mean rate of entering state 0 is μ_1.
- From any *other* state, this mean rate is 0.
- Therefore, the overall mean rate at which the process leaves its current state to enter state 0 (the *mean entering rate*) is

$$\mu_1 P_1 + 0(1 - P_1) = \mu_1 P_1$$

THE BIRTH-AND-DEATH PROCESS (Cont'd)

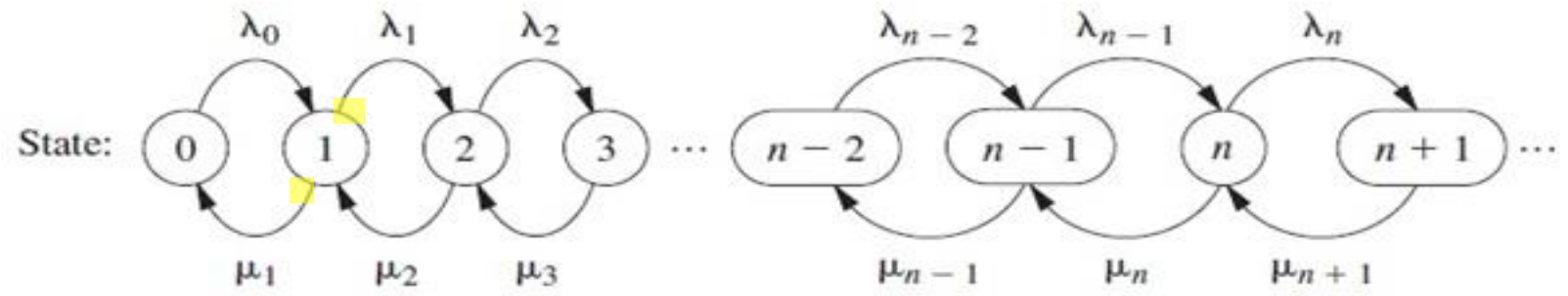


Fig 1: Rate diagram for the birth-and-death process

- By the same reasoning, the *mean leaving rate* must be $\lambda_0 P_0$, so the balance equation for state 0 is

$$\mu_1 P_1 = \lambda_0 P_0$$

- For every other state there are two possible transitions both into and out of the state.
- Therefore, each side of the balance equations for these states represents the *sum* of the mean rates for the two transitions involved.
- Otherwise, the reasoning is just the same as for state 0.

Writing Flow Equations

$\lambda_0 = \lambda_1 = \lambda_2 = \cdots = \lambda_n = \lambda$

$\mu_0 = \mu_1 = \cdots = \mu_n = \mu$

State 0:

$\lambda P_0 = \mu P_1$

$\Rightarrow \boldsymbol{P_1 = \rho P_0}$

State 1:

$\lambda P_0 + \mu P_2 = \lambda P_1 + \mu P_1$

$\Rightarrow \mu P_2 = (\lambda + \mu)\frac{\lambda}{\mu} P_0 - \lambda P_0$

$= [\frac{\lambda}{\mu} + 1 - 1]\, \lambda P_0$

$\therefore \boldsymbol{P_2 = \rho^2 P_0}$

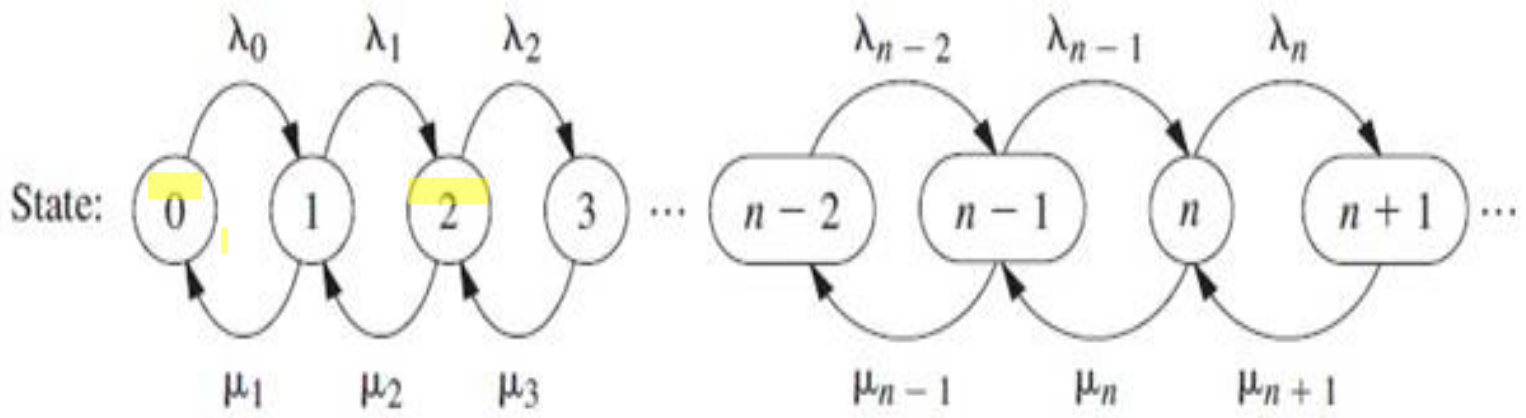


Fig 1: Rate diagram for the birth-and-death process

Writing Flow Equations

So, from these two equations, we can generalize the relation:

$$\boldsymbol{P_n} = \boldsymbol{\rho^n P_0}$$

$\because \sum_{n=0}^{\infty} P_n = 1$

$\Rightarrow \boldsymbol{P_0} + \boldsymbol{\rho P_0} + \boldsymbol{\rho^2 P_0} + \ldots\ldots + \infty = 1$

$\Rightarrow \boldsymbol{P_0}\left(\frac{1}{1-\boldsymbol{\rho}}\right) = 1$

$\therefore \boldsymbol{P_0} = 1 - \boldsymbol{\rho}$

Q1: Explain what does $\rho = 1 - P_0$ indicate ?

ρ is the probability that server is busy and $1 - P_0$ suggests the probability of at least one customer.

Generalizing, we will get:

$$\therefore \quad \boldsymbol{P_n} \quad = \boldsymbol{\rho^n(1 - \rho)}$$

CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 21)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 20

Transient State and Steady State

M/M/1 Queue Analysis (Birth-Death Process)

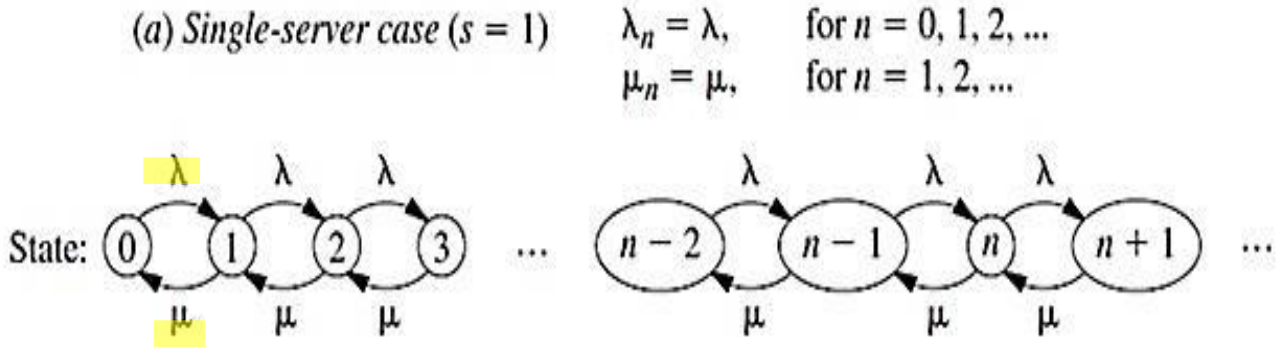
Balance/Flow Equations for BD-Process

Chapter # 6 (Cont'd)

FUNDAMENTALS OF QUEUING MODELS

M/M/1 Analysis (Cont'd)

- Assumptions
  - all *inter-arrival times* are independently and identically distributed according to an *exponential distribution*, all *service times* are independent and identically distributed according to *another exponential distribution and* the *number of servers* is *1*.
- Just the special case of the birth-and-death process where
  - the queuing system's *mean arrival rate* and *mean service rate per busy server* are constant (λ and μ, respectively) regardless of the state of the system.
- When the system has just a *single server* ($s = 1$), the implication is that the parameters for the birth-and-death process are $\lambda_n = \lambda$ (n = 0, 1, 2, …) and $\mu_n = \mu$ (n = 1, 2, …).



M/M/1 Analysis (Cont'd) - BD Process

Q2: *What is the probability of at least k customers in the system?*

$$P[N \geq k] = \sum_{n=k}^{\infty} P_n$$

$$= \rho^k P_0 + \rho^{k+1} P_0 + \cdots \cdot \infty$$

$$= \boldsymbol{P_0} \left(\frac{\rho^k}{1 - \boldsymbol{\rho}}\right) = (1 - \rho) \left(\frac{\rho^k}{1 - \boldsymbol{\rho}}\right)$$

$$P[N \geq k] = \rho^k$$

M/M/1 Analysis (Cont'd) - BD Process

Q3: *Determine the expected number of customers in the system.*

$$L = E[N] = \sum_{n=0}^{\infty} n\, P_n$$

$$= \sum_{n=1}^{\infty} n\rho^n(1-\rho)$$

$$= (1-\rho)\left(\sum_{n=1}^{\infty} n\rho^n\right)$$

$$= (1-\rho)\,\rho\,\frac{d}{d\rho}\left(\sum_{n=1}^{\infty} \rho^n\right)$$

$$= \rho(1-\rho)\,\frac{d}{d\rho}\left(\frac{\rho}{1-\rho}\right)$$

$$= \rho(1-\rho)\left(\frac{1-\rho+\rho}{(1-\rho)^2}\right)$$

$$L = E[N] = \frac{\rho}{1-\rho}$$

M/M/1 Analysis (Cont'd) - BD Process

Q4: *Determine the expected time spent in the system.*

Starting with Little's Law, $L = \lambda W$

$$\Longrightarrow W = \frac{L}{\lambda} = \frac{\rho}{1-\rho} \cdot \frac{1}{\lambda}$$

$$\Longrightarrow W = \frac{\lambda/\mu}{1-\lambda/\mu} \cdot \frac{1}{\lambda}$$

$$\Longrightarrow W = \frac{1}{\mu-\lambda}$$

$$\Longrightarrow W = \frac{1}{\mu\,(1-\rho)} = \frac{W_s}{(1-\rho)}$$

M/M/1 Analysis (Cont'd) - BD Process

Q5: *Determine the mean waiting time.*

$$\Longrightarrow W_q = W - Ws$$

$$\Longrightarrow W_q = \frac{1}{\mu-\lambda} - \frac{1}{\mu}$$

$$\Longrightarrow W_q = \frac{\lambda}{\mu(\mu-\lambda)}$$

$$\Longrightarrow W_q = \rho W = \frac{\rho}{1-\rho} \cdot \frac{1}{\mu}$$

$$\Longrightarrow W_q = \frac{L}{\mu}$$

M/M/1 Analysis (Cont'd) - BD Process

Q6: *Determine the expected number of customers in the queue.*

$$\Longrightarrow L_q = \lambda W_q$$

$$\Longrightarrow L_q = \frac{\lambda}{\mu} . L$$

$$\Longrightarrow L_q = \frac{\rho^2}{1-\rho}$$

M/M/1 Analysis (Cont'd) - BD Process

Q7: *Determine the mean number of customers in the service facility.*

$$\Longrightarrow L_s = L - L_q$$

$$\Longrightarrow L_s = \frac{\rho}{1-\rho} - \frac{\rho^2}{1-\rho}$$

$$\Longrightarrow L_s = \rho$$

M/M/1 Analysis (Cont'd) - BD Process

Q8: *Give the formulae for the distribution of total time, service time & waiting time.*

By hypothesis;

$$W_s(t) = P\{s \leq t\} = 1 - e^{-\mu t} = 1 - e^{-t/Ws}$$

Similarly,

$$W(t) = P\{W \leq t\} = 1 - e^{-t/W}$$

Also,

$$W_q(t) = P\{q \leq t\} = 1 - \rho e^{-t/W}$$ (When queue discipline is FCFS)

M/M/1 Analysis (Cont'd) - BD Process

Q8: *Give the formulae for the distribution of total time, service time & waiting time.*

By hypothesis;

$$W_s(t) = P\{s \leq t\} = 1 - e^{-\mu t} = 1 - e^{-t/Ws}$$

Similarly,

$$W(t) = P\{W \leq t\} = 1 - e^{-t/W}$$

Also,

$$W_q(t) = P\{q \leq t\} = 1 - \rho e^{-t/W}$$ (When queue discipline is FCFS)

PASTA (Poisson Arrivals See Time Averages) Theorem

- The state of an M/M/1 queue is the number of customers in the system.
- More general queueing systems have a more general state that may include how much service each customer has already received.
- For Poisson arrivals, the arrivals in any future increment of time is independent of those in past increments and for many systems of interest, independent of the present state S(t) (true for M/M/1, M/M/m, and M/G/1).
- In steady state, arrivals see steady state probabilities.

M/M/1 Analysis – Example Problem

A computer system has tasks arriving on average every 0.4 sec and requires service of 0.3 sec CPU time to process each job.

Assume an exponential distribution to process each job and their random arrivals. Find out:

a) *Utilization of CPU*

b) *Avg. System Time*

c) *Avg. Waiting Time*

d) *Avg. number of jobs in the queue*

e) *Avg. number of jobs inside the system*

f) *Probability that there are 5 or more tasks waiting for service.*

g) *If inter-arrival time is reduced to 0.25 sec, what will happen to the queuing system ?*

Answers:

a) 0.75

b) 1.2 sec

c) 0.9 sec

d) 2.25

e) 3 tasks

f) 0.1779

g) Unstable system

Task

The time between requests to a Web server is found to approximately follow an exponential distribution with a mean time between requests of 8 msec. The time required by the server to process each request is also found to roughly follow an exponential distribution with an average service time of approximately 5 msec.

(Use M/M/1 queueing analysis)

a) What is the average response time observed by the users making requests on this server?

b) How much faster must the server process requests to halve this average response time?

Answers:

a) 13.33 msec

b) 275 / sec

CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 22)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 21

M/M/1 Queue Analysis (Birth-Death Process)

PASTA Theorem

Example Problems

Chapter # 6 (Cont'd)

FUNDAMENTALS OF QUEUING MODELS

M/M/1/K System (Finite Buffer Capacity)

The **M/M/1/k** queue is a **short hand** notation for the **M/M/1/k/∞/FIFO** queue.

- **M = Arrival process** is **Poisson**
- **M = Service (departure) process** is **Exponential**
- **1** = There is **1 server** in system
- **k = Queue capacity**; **the $(k+1)^{th}$ arriving client** will be ***rejected***
- ∞ = **Infinite population size** (the **arrival process** will be **unaffected** by the number of clients already in the system)
- **FIFO** = First In First Out Service

M/M/1/K System Analysis

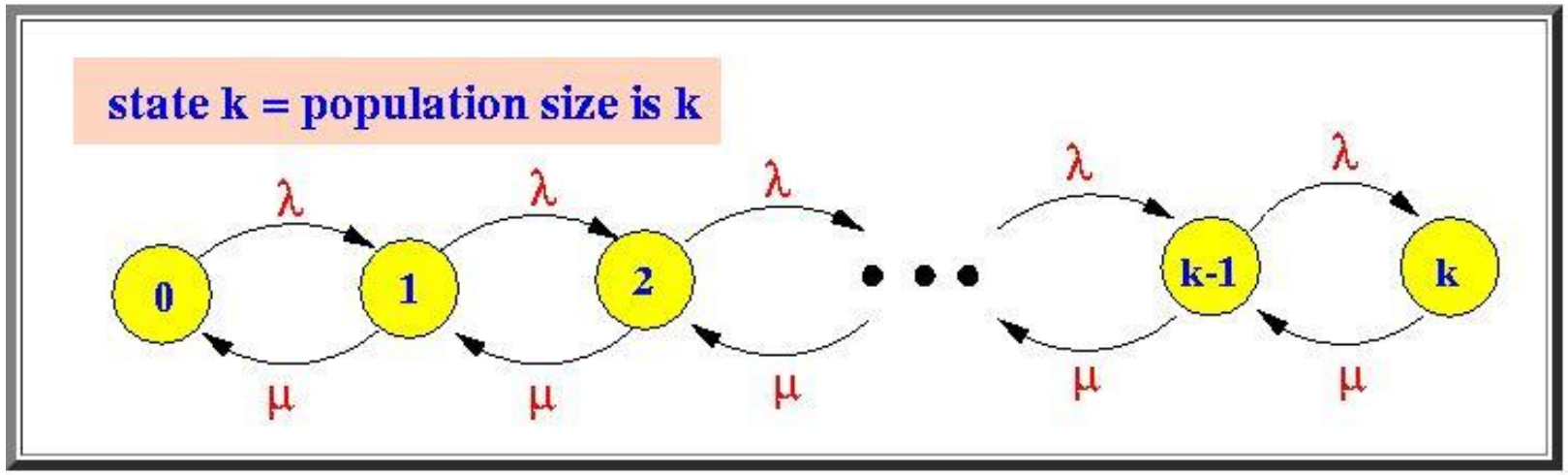


Fig 1: Rate diagram for the M/M/1/K System

Starting with: $P_n = \rho^n P_0$; n =0 , 1, 2, ……., k

$$\sum_{n=0}^{k} P_n = 1$$

M/M/1/K System Analysis

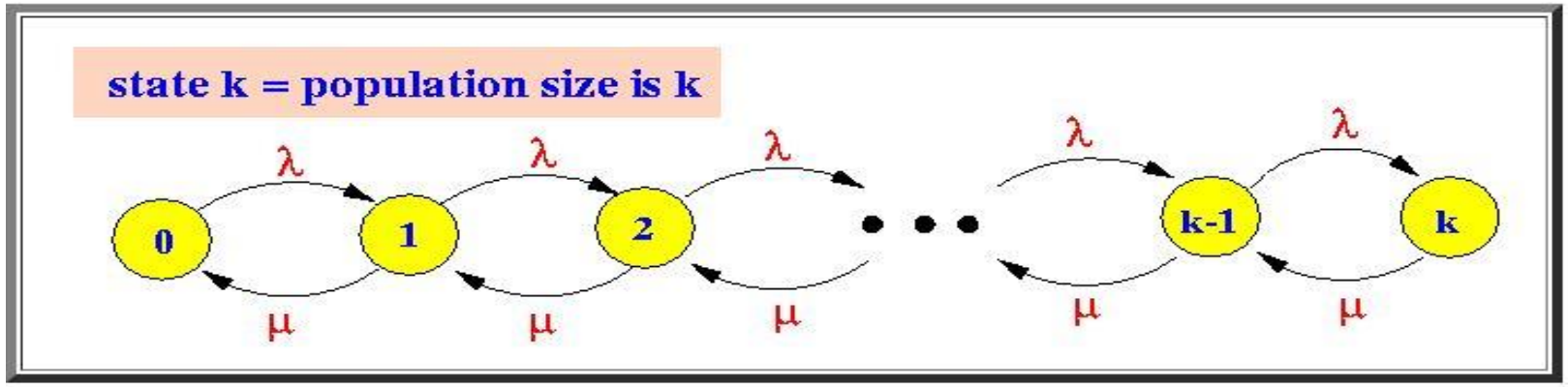


Fig 1: Rate diagram for the M/M/1/K System

Starting with: $P_n = \rho^n P_0$; n =0 , 1, 2, ……, k

$$P_0 + \rho P_0 + \rho^2 P_0 + \ldots\ldots + \rho^k P_0 = 1$$

$$\Rightarrow \frac{1-\rho^{k+1}}{1-\rho} P_0 = 1$$

$$\Rightarrow P_0 = \frac{1-\rho}{1-\rho^{k+1}}$$

$$\therefore P_n = \rho^n \frac{1-\rho}{1-\rho^{k+1}}$$

$$S = a\left(\frac{1-r^n}{1-r}\right)$$

M/M/1/K System Analysis

Because of the limited system capacity, we do not require $\lambda < \mu$.

If $\lambda = \mu$ $(i.e. \boldsymbol{\rho = 1})$:

$$\sum_{n=0}^{k} P_n = 1$$

$$\sum_{n=0}^{k} \boldsymbol{\rho^n P_0} = \mathbf{1}$$

$$\Rightarrow (1+1+1+ \ldots\ldots\ldots + 1)\ \boldsymbol{P_0} = 1$$

$$\Rightarrow\ \boldsymbol{P_0} = \frac{1}{1+k} \Rightarrow\ \boldsymbol{P_n} = \frac{1}{1+k}$$

$$S = \frac{n}{2}\left(2a + (n-1)d\right)$$

$$= \frac{k+1}{2}\left(2 + (k)0\right)$$

M/M/1/K Analysis (Task)

Q) Derive the equation to determine the total number of customers in the system.

Hint: Start with $L = E[N] = \sum_{n=0}^{k} n P_n$

Result for verification:

$$L = \frac{\rho}{1-\rho} - (k+1)\frac{\rho^{k+1}}{1-\rho^{k+1}}$$

M/M/1/K Analysis (Task)

Total Number of Customers when $\rho = 1$,

$$L = \sum_{n=0}^{k} n\left(\frac{1}{1+k}\right)$$

$$L = \frac{1}{1+k} \sum_{n=0}^{k} n$$

$$L = \frac{1}{1+k} . \frac{k}{2} (k+1)$$

$$L = \frac{k}{2}$$

M/M/1/K Analysis

Average Number of customers in the Service & in the Queue:

$$L_s = E[N_s] = P[N_s \mid N = 0]\, P[N = 0] + P[N_s \mid N > 0]\, P[N > 0]$$

$$= 0 \times P_0 + 1 \times (1 - P_0)$$

$$L_s = 1 - P_0$$

$$L_q = L - L_s$$

$$= L - (1 - P_0)$$

M/M/1/K Analysis

Effective Arrival Rate (λ_a)

Customers are turned away when there are k customers in the system:

$$\lambda_a = \lambda (1 - P_k)$$

Using Little's Law,

$$W = \frac{L}{\lambda_a} = \frac{L}{\lambda (1 - Pk)}$$

$$W_q = \frac{L_q}{\lambda (1 - Pk)}$$

M/M/1/K Analysis

Server Utilization (U):

For balancing M/M/1/K system:

$$\lambda (1 - \boldsymbol{P_k}) = \mu(1 - \boldsymbol{P_0})$$

Probability that the server is busy is given by:

$$\Rightarrow \mathbf{U} = 1 - \boldsymbol{P_0} = \frac{\lambda}{\mu}(1 - \boldsymbol{P_k})$$

$$\Rightarrow \mathbf{U} = \boldsymbol{\rho}(1 - \boldsymbol{P_k})$$

Example Problem 1

Packets arrive at a router according to Poisson distribution at an average rate of 6 per second. The router is serving packets at an average rate of 8 packets per second with exponential distribution of service time. However, the buffer has capacity for only 8 packets. Calculate:

a) average response time of packets

b) average number of packets dropped if a total of 5000 packets approach for the service

c) average number of packets in the router

d) average number of packets in the buffer

Answers:

a) *0.4081 sec*
b) *100 packets*
c) *2.4*
d) *1.665*

Example Problem 2

Consider the following single-server queue: the inter-arrival time is exponentially distributed with a mean of 10 minutes and the service time is also exponentially distributed with a mean of 8 minutes, find out:

(i) mean wait in the queue,

(ii) mean number in the queue,

(iii) the mean wait in the system,

(iv) mean number in the system and

(v) proportion of time the server is idle.

Answers:

i) 3.2

ii) 32 mins

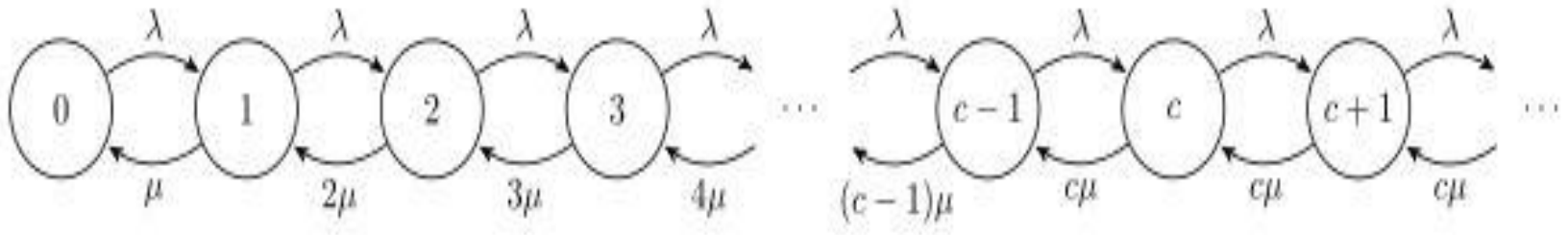
iii) 40 mins

iv) 4

v) 0.2

Task Assignment *(To be submitted)*

Perform the analysis of M/M/C Queuing System where C indicates the number of identical servers. The rate diagram for this system is given by:



Server Utilization, $\rho = \frac{\lambda}{c\mu} < 1$ for stability.

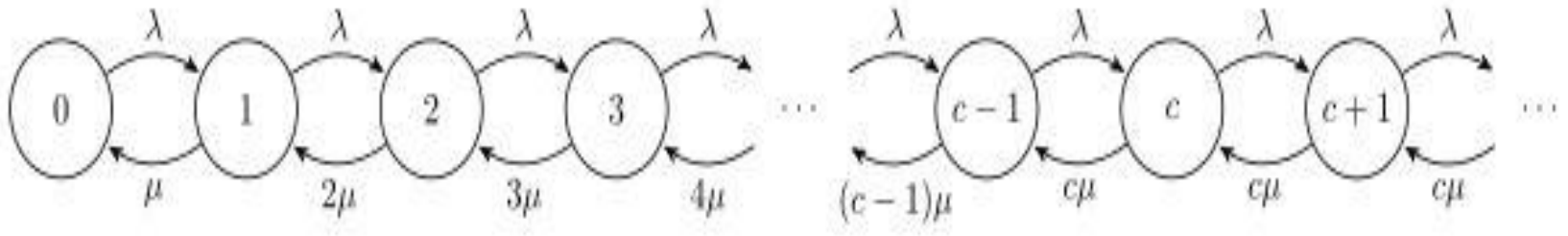
Consider the value of c = 3.

Provide the flow equations and determine the generalized results of the following:

- *Average number of customers in the system.* ($L = E[N] = \sum_{n=1}^{\infty} nP_n$)
- *Average number of customers in the queue.* ($L_q = \sum_{n=0}^{\infty} nP_{n+c}$)
- *Average number of customers in the server.*
- *Probability of Queuing.* ($P[N \geq C] = \sum_{n=c}^{\infty} P_n$)
- *Average number of busy servers.*

Task Assignment *(To be submitted)*

Perform the analysis of M/M/C Queuing System where C indicates the number of identical servers. The rate diagram for this system is given by:



Server Utilization, $\rho = \frac{\lambda}{c\mu} < 1$ for stability.

Consider the value of c = 3.

Provide the flow equations and determine the generalized results of the following:

- *Average number of customers in the system.* ($L = E[N] = \sum_{n=1}^{\infty} nP_n$)
- *Average number of customers in the queue.* ($L_q = \sum_{n=0}^{\infty} nP_{n+c}$)
- *Average number of customers in the server.*
- *Probability of Queuing.* ($P[N \geq C] = \sum_{n=c}^{\infty} P_n$)
- *Average number of busy servers.*

CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 23)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 22

M/M/1/K Queue Analysis

Example Problems

Chapter # 6 (Cont'd)

FUNDAMENTALS OF QUEUING MODELS

QUEUING MODELS INVOLVING NONEXPONENTIAL DISTRIBUTIONS

- The assumption of exponential inter-arrival times implies that arrivals occur randomly (a Poisson input process).

- Furthermore, the actual service-time distribution frequently deviates greatly from the exponential form, particularly when the service requirements of the customers are quite similar.

- Therefore, it is important to have available other queuing models that use alternative distributions.

- Unfortunately, the mathematical analysis of queuing models with non-exponential distributions is much more difficult.

QUEUING MODELS INVOLVING NONEXPONENTIAL DISTRIBUTIONS

➢The *M*/*G*/1 Model

Assumptions

1. The queuing system has a *single server* and

2. A *Poisson input process* (exponential inter-arrival times) with a *fixed* mean arrival rate λ.

3. The customers have *independent* service times with the *same* probability distribution.

- In fact, it is only necessary to know (or estimate) the mean $1/\mu$ and variance σ^2 of this distribution.

QUEUING MODELS INVOLVING NONEXPONENTIAL DISTRIBUTIONS

- The readily available steady-state results for this general model are the following:

$$P_0 = 1 - \rho,$$

$$L_q = \frac{\lambda^2\sigma^2 + \rho^2}{2(1-\rho)},$$

$$L = \rho + L_q,$$

$$W_q = \frac{L_q}{\lambda},$$

$$W = W_q + \frac{1}{\mu}.$$

MODELS INVOLVING NONEXPONENTIAL DISTRIBUTIONS

- The formula for L_q is one of the most important results in queuing theory because of
  - its ease of use and
  - the prevalence of $M/G/1$ queuing systems in practice.

- This equation for L_q (or its counterpart for W_q)
  - commonly referred to as **Pollaczek-Khintchine formula**,
  - named after two pioneers in the development of queuing theory
  - who derived the formula independently in the early 1930s.

- The model does not provide a closed-form expression for P_n because of analytical intractability.

MODELS INVOLVING NONEXPONENTIAL DISTRIBUTIONS

- For any fixed expected service time $1/\mu$, notice that L_q, L, W_q, and W all increase as σ^2 is increased.

- This result is important because it indicates that
  - the consistency of the server has a major bearing on the performance of the service facility—
  - not just the server's average speed.

- When the service-time distribution is exponential, $\sigma^2 = 1/\mu^2$, and the preceding results will reduce to the corresponding results for the $M/M/1$ model.

UNIFORM DISTRIBUTION

- X is a uniform random variable if the PDF of X is

$$f_X(x) = \begin{cases} \dfrac{1}{(b-a)} & a \le x < b \\ 0 & otherwise \end{cases}$$

where the two parameters are **b > a**.

Theorem

- If X is a uniform random variable with parameters **a** and **b > a**.
- The CDF of X is

$$F_X(x) = \begin{cases} 0 & x \le a \\ \dfrac{(x-a)}{(b-a)} & a < x \le b \\ 1 & x > b \end{cases}$$

- The expected value of X is E[X] = **(b+a)/2**
- The variance of X is Var[X] = **$(b-a)^2/12$**

Example Problem 1

M/G/1

Consider the following single-server queue: the inter-arrival time is exponentially distributed with a mean of 10 minutes and the service time has the uniform distribution with a maximum of 9 minutes and a minimum of 7 minutes, find out:

(i) mean wait in the queue,

(ii) mean number in the queue,

(iii) the mean wait in the system,

(iv) mean number in the system and

(v) proportion of time the server is idle.

Answers

i) *1.602*

ii) *16.02 mins*

iii) *24.02 mins*

iv) *2.402*

v) *0.2*

MODELS INVOLVING NONEXPONENTIAL DISTRIBUTIONS

The *M/D/s* Model

- The $M/D/s$ model often provides a reasonable representation for this kind of situation,
  - because it assumes that all service times actually equal some fixed *constant* (the *degenerate* service- time distribution) and
  - that we have a *Poisson* input process with a fixed mean arrival rate λ.

- When there is just a single server, the $M/D/1$ model is just the special case of the $M/G/1$ model where $\sigma^2 = 0$, so that the *Pollaczek-Khintchine formula* reduces to

$$L_q = \frac{\rho^2}{2(1-\rho)},$$

- where L, W_q, and W are obtained from L_q as just shown.

MODELS INVOLVING NONEXPONENTIAL DISTRIBUTIONS

- Notice that these L_q and W_q are exactly *half* as large as those for the exponential service-time case of the $M/M/1$ model, where $\sigma^2 = 1/\mu^2$, so decreasing σ^2 can *greatly* improve the measures of performance of a queuing system.

- For the multiple-server version of this model ($M/D/s$), a complicated method is available for deriving the steady-state probability distribution of the number of customers in the system and its mean [assuming $\rho = \lambda/(s\mu) < 1$].

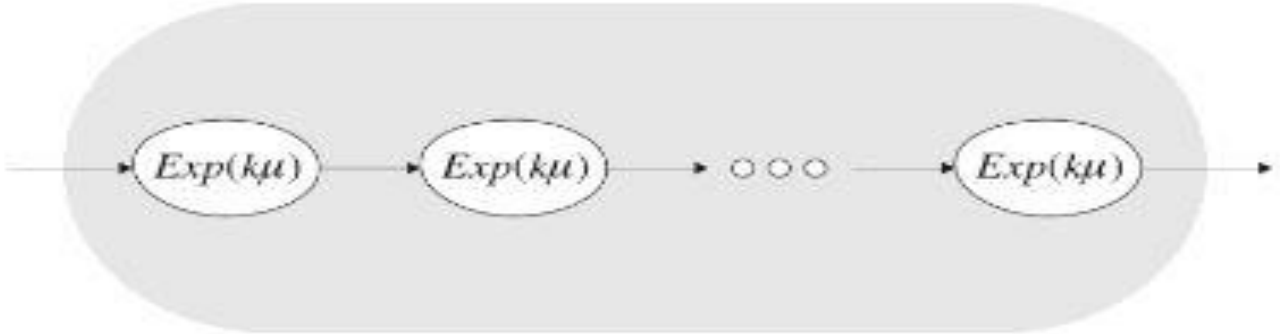
MODELS INVOLVING NONEXPONENTIAL DISTRIBUTIONS

- Notice that these L_q and W_q are exactly *half* as large as those for the exponential service-time case of the $M/M/1$ model, where $\sigma^2 = 1/\mu^2$, so decreasing σ^2 can *greatly* improve the measures of performance of a queuing system.

- For the multiple-server version of this model ($M/D/s$), a complicated method is available for deriving the steady-state probability distribution of the number of customers in the system and its mean [assuming $\rho = \lambda/(s\mu) < 1$].

Erlang-n Distribution

- Think of putting exponential distribution in series.
- If a random variable X is the sum of n-identical exponential random variables of service times n/μ, then X is said to have an Erlang-n distribution. Service time of a single server is given by $1/\mu$.
- The customer has to visit each stage in the facility to complete the service.



- A generalized Erlang Distribution is the sum of exponential random variables with different rates (also called a hypo-exponential distribution)

Erlang-n Distribution

- An Erlang random variable X with scale parameter α and n stages has probability density function:

$$f(x) = \frac{x^{n-1}e^{-x/\alpha}}{\alpha^n(n-1)!} \qquad x > 0.$$

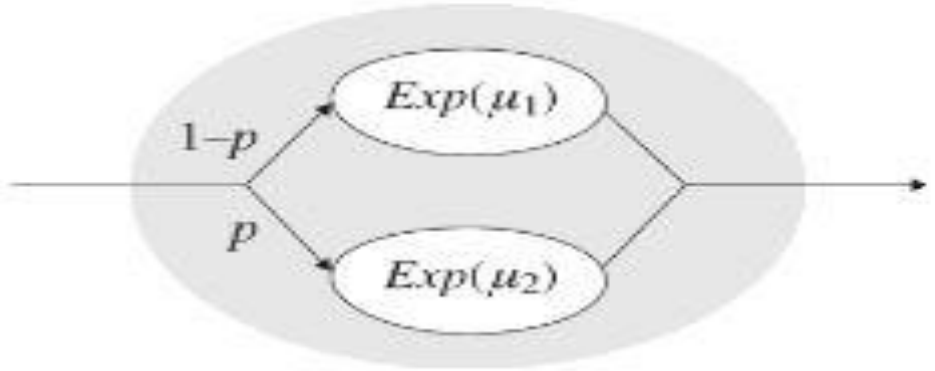
- The cumulative distribution function on the support of X is:

$$F(x) = P(X \leq x) = 1 - \sum_{i=0}^{n-1} \frac{e^{-x/\alpha}x^n}{\alpha^n n!} \qquad x > 0.$$

- The population mean and variance are given by E[X] = nα V [X] = nα^2 respectively. α is given by 1/μ.

Hyper-Exponential Distribution

- Think of putting exponential distribution in parallel.



- A random variable X is hyper-exponentially distributed if X is with probability p_i , i = 1, . . . , k an exponential random variable X_i with mean $1/\mu_i$. For this random variable we use the notation $H_k(p_1, \ldots, p_k; \mu_1, \ldots, \mu_k)$, or simply H_k. The density is given by:

$$f(t) = \sum_{i=1}^{k} p_i \mu_i e^{-\mu_i t}, \qquad t > 0,$$

Hyper-Exponential Distribution

- The cumulative distribution function on the support of X is:

$$F(x) = P(X \leq x) = 1 - \sum_{i=1}^{n} p_i e^{-x/\alpha_i} \qquad x > 0.$$

α is given by 1/μ.

CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 24)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 23

Queuing Models involving non-exponential distribution

Example Problem

Erlang-n and Hyper-exponential Distribution

Chapter # 6 (Cont'd)

FUNDAMENTALS OF QUEUING MODELS

QUEUING NETWORKS

- A job may receive service at one or more queues before exiting from the system. Such systems are modelled by queueing networks.

- In general, a model in which jobs departing from one queue arrive at another queue (or possibly the same queue) is called a **queuing network.**

- There are two types of queuing networks:
  - Open queuing networks
  - Closed queuing networks

OPEN QUEUING NETWORKS

- An **open queueing network** has external arrivals and departures. The jobs enter the system at "In" and exit at "Out".

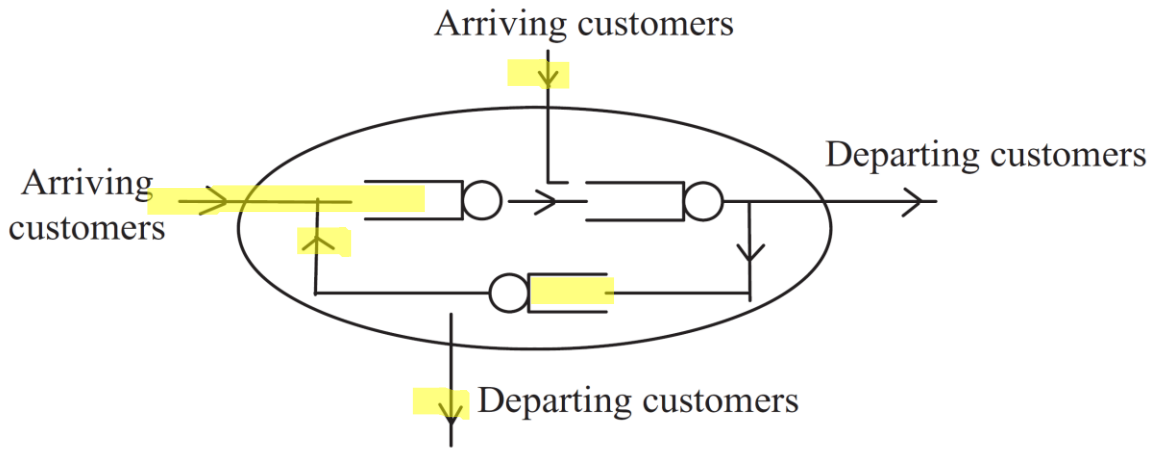


Fig 1: Open Queuing Network

- The number of jobs in the system varies with time.
- In analyzing an open system, we assume that the throughput is known (to be equal to the arrival rate), and the goal is to characterize the distribution of number of jobs in the system.

CLOSED QUEUING NETWORKS

- A **closed queueing network** has no external arrivals or departures.
- The jobs in the system keep circulating from one queue to the next. The total number of jobs in the system is constant.

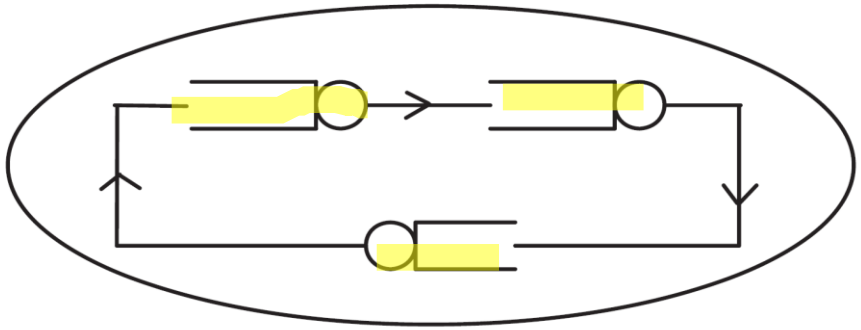


Fig 2: Closed Queuing Network

- It is possible to view a closed system as a system where the Out is connected back to the In.
- The jobs exiting the system immediately re-enter the system.

Jackson's Network

- A Jackson network consists of a number of nodes, where each node represents a queue in which the service rate can be both node-dependent (different nodes have different service rates) and state-dependent (service rates change depending on queue lengths).

- There is no notion of priority in serving the jobs: all jobs at each node are served on a first-come, first-served basis.

Necessary conditions for a Jackson Network

- If the network is open, any external arrivals to node i form a Poisson process,
- All service times are exponentially distributed and the service discipline at all queues is first-come, first-served,
- A customer completing service at queue i will either move to some new queue j with probability P_{ij} or leave the system with the probability $1 - \sum_{j=1}^{m} P_{ij}$,
- The utilization of all of the queues is less than one.

Jackson's Theorem

- Let n = (n_1, n_2, ….., n_M) represent the global state of an open network of M nodes.

- Let n_i = state of node i

(i.e. the total number of customers present in node i)

- Jackson's Theorem states that:

$$p(n) = p(n_1)\, p(n_2) \ldots\ldots p(n_M)$$

Burke's Theorem

Consider an M/M/1, M/M/c or M/M/∞ system with arrival rate λ. Suppose that the system is in steady state. Then:

a) The departure process is also Poisson with rate λ.

b) At each time t, the number of customers in the system is independent of the departures prior to t.

Two Stages Tandem Network

Consider the given tandem network consisting of two M/M/1 queues:

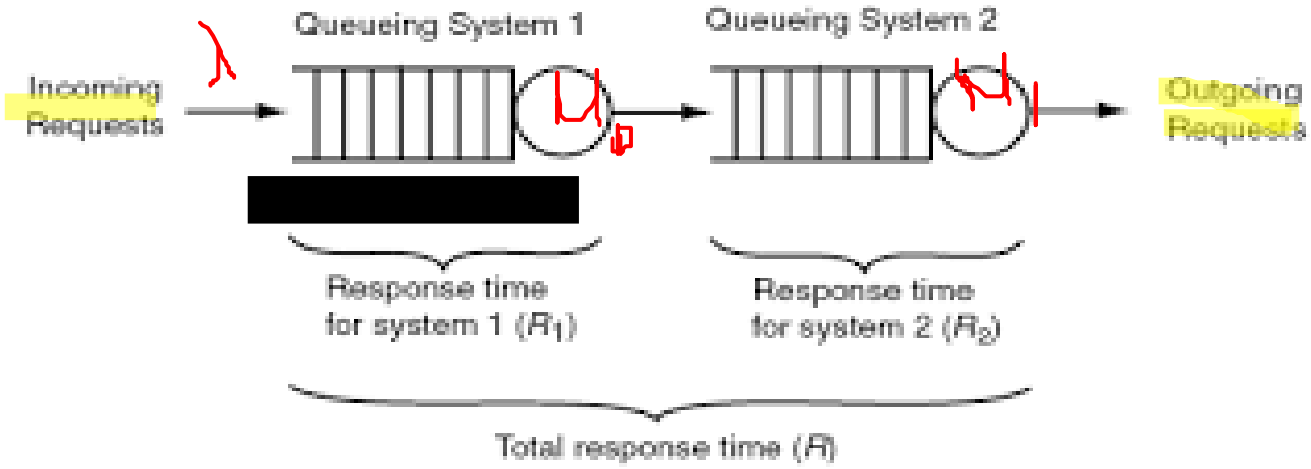


Fig 3: Two Stages Tandem Network

Two stages TN can be modelled as a continuous time Markov Chain.

Two Stages Tandem Network (Cont'd)

Two stages TN can be modelled as a continuous time Markov Chain.

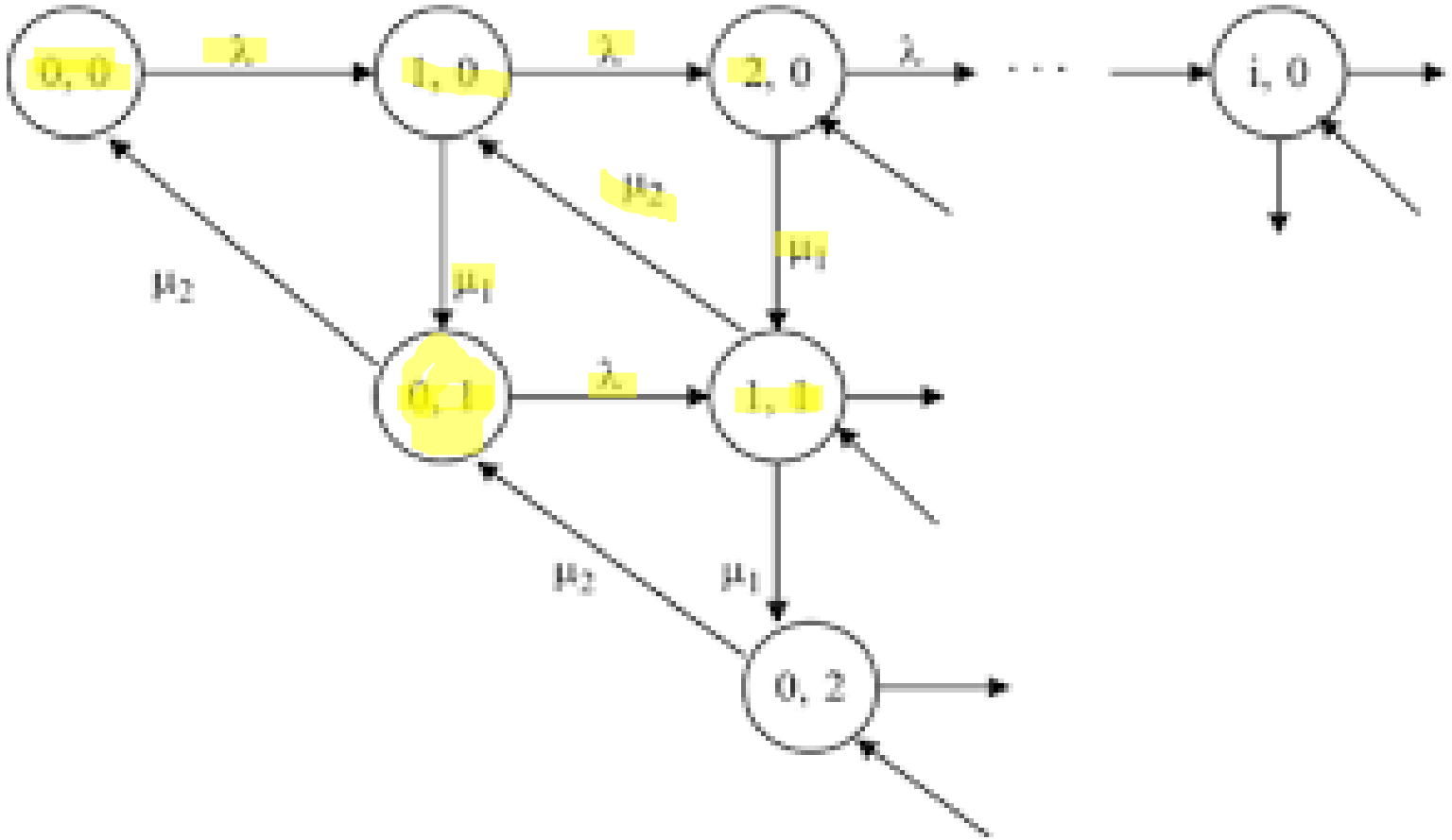


Fig 4: Rate Diagram for 2-Stages Tandem Network

Two Stages Tandem Network (Cont'd)

Applying Jackson's Theorem,

$$P(n_0,n_1) = P(n_0)\,P(n_1)$$
$$= \rho_0^{n_0}(1 - \rho_0)\,\rho_1^{n_1}(1 - \rho_1)$$

– This equation can be extended to any number of stages.

$$\text{For stability, } \rho_0 = \frac{\lambda}{\mu_0} < 1 \;\&\; \rho_1 = \frac{\lambda}{\mu_1} < 1$$

Example Problem 1

A repair facility shared by a large number of machines has two sequential stations (with service rates 1/hr & 2/hr). The cumulative failure rate of all the machines is 0.5/h. Assume a 2-stage tandem network with failure rates following Poisson distribution.

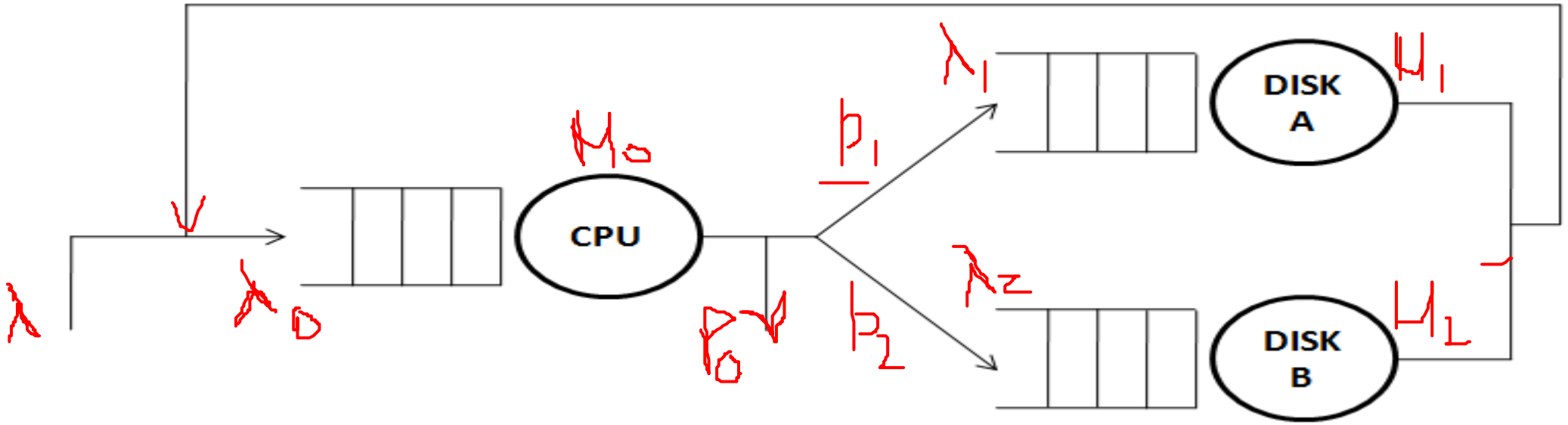
Calculate the average repair time and probability that both the stations are idle.

Answers:

i) 8/3 hours

ii) 0.375

Open Central Server Networks



$$\lambda_j = \begin{cases} \dfrac{\lambda}{p_0}, & j = 0 \\ \dfrac{p_i}{p_0}\lambda, & j = 1, 2, 3, \ldots\ldots, m \end{cases}$$

Example Problem 2

Consider an open central server queuing model with a CPU (service rate: 2 programs/sec) and two I/O channels (each with service rate: 1.2/sec). The external job arrival rate is 1/7 programs/sec according to Poisson Process. The branching probabilities are $p_0 = 0.1$, $p_1 = 0.3$ and $p_2 = 0.6$. Assume all the service times are independent exponentially distributed RVs.

Calculate:

a) The probability that there is just a single program in the CPU and Disk B and none in the Disk A.
b) Mean number of programs in the network.
c) Average response time of the overall network.

Answers:

a) 0.02677

b) 5.5554

c) 38.88 sec

CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 25)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 24

Open and Close Queuing Networks

Jackson’s and Burke’s Theorem

Two Stages Tandem Network

Open Central Server Network

Chapter # 7

SIMULATION MODELING

Prepared by: Ms. Fakhra Aftab (Lecturer, CISD, NEDUET)

Simulation

The technique of using a computer to imitate the operation of an entire process or system

- **Purpose:** provides a way to predict the performance of a computer system or compare several alternatives.

- **When to use simulation:** the stochastic system involved is too complex to be analyzed satisfactorily by the kinds of mathematical models (e.g., queuing models).

Advantages

- Simulation is much *less expensive* than actually building a machine.
- It is much *more flexible* than measuring the performance of a real machine.
- Simulation allows a system to be studied in *more detail* than analytical modeling.
- Simulation enables to *study dynamic systems* in real, compressed or expanded time.

Terminology

- **State Variables:** The variables whose values define the state of the system. In the CPU scheduling simulation, the state variable is the *length of the job queue*.

- **Event:** A change in the system state. In the CPU scheduling simulation, there are two events: *arrival of a job* and *departure of a job*.

- **Continuous-Time and Discrete-Time Models**
  - CT models: A model in which the system state is defined at all times. e.g. the CPU scheduling model.
  - DT models: The system state is defined only at particular instants in time. e.g. a process that activates every 5-msec and records the number of blocked processes in the system.

- **Continuous-State and Discrete-State Models:**
  - CS models: State variables are continuous e.g. pressure of gas in thermodynamic systems. Also called a continuous-event model.
  - DS models: state variables are discrete e.g. length of job queue. Also called a discrete-event model.

Thus, there are four possible combinations:

- discrete state/discrete time,
- discrete state/continuous time,
- continuous state/discrete time, and
- continuous state/continuous time models.

- **Open and Closed Models**

  - Open model: The input is external to the model and is independent of it. e.g. new jobs enter the model from outside.

  - Closed model: There is no external input. e.g. the same jobs keep circulating in the model. A job departing the second queue reenters the first queue. This is therefore a closed model.

Simulation Efficiency Considerations

1) An important limitation of simulation: the simplifying assumptions restrict its ability to exactly duplicate the behavior of a real system.

2) Determining the level of detail necessary when writing a new simulator depends on the level of detail necessary to solve the problem, and the consequences of being wrong.

- E.g., the consequences of being wrong when using a simulator to determine the best cache size for a new system are relatively small.
- The incorrect operation of a microprocessor used to control a heart pacemaker, though, can have much more serious consequences. Thus, in this case, a very detailed simulation may be warranted.

Types of Simulations

- Emulation
- Static (or Monte-Carlo) simulation
- Discrete-event simulation
- Continuous-event simulation

1) Emulation

- An **emulator** is hardware or software that enables one computer system (called the *host*) to behave like another computer system (called the *guest*). An emulator typically enables the host system to run software or use peripheral devices designed for the guest system.

- Emulation allows program to run on the platform other than the one for which they were originally developed.

- For Example
  - Android Emulator
  - DOSBox emulates the command-line interface of DOS

2) **Static (Monte Carlo) Simulation**

- No time parameter.
- The simulation is run until further refinement of the state of the simulated system is no longer useful or possible.
- Static simulations are often used to evaluate some physical phenomenon (probabilistically), or to numerically estimate the solution of some mathematical expression, such as a complex integral.

Example

- Consider the problem of numerically determining the value of π.
- We begin with a geometric description as shown in Figure 1.
- Since the area of a circle with a radius of 1 is $\pi(1)^2 = \pi$, the area of the quarter-circle within the first quadrant is $\pi/4$.
- The area contained within the unit square in this quadrant is simply 1.
- Thus, the ratio of the area of the quarter-circle to the area of the square, which we denote R, is $R=\pi/4$.

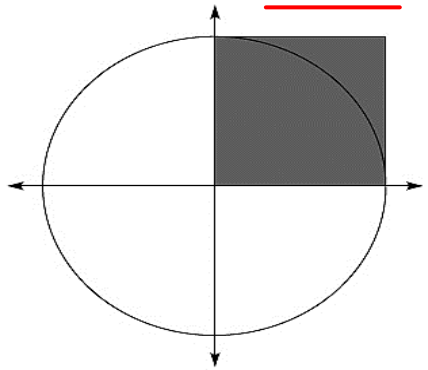


Figure 1: A simple Monte Carlo simulation, to estimate the numerical value of π.

- The numerical value of π then can be found from $\pi=4R$.
- The problem of computing the value of π has been transformed into the equivalent geometric problem of determining the ratio of the two areas, R.
- A Monte Carlo simulation can be used to find R by modeling an equivalent physical system.
- Imagine throwing a large number of darts randomly at figure 1 such that every dart hits within the unit square.
- Then we count the number of times a dart hit within the quarter-circle, n_{circ}, and the total number of darts thrown, n_{total}.
- Then the desired ratio of the two areas is $R = n_{circ} / n_{total}$.

- We can simulate this dart-throwing experiment by generating two random numbers, u_1 and u_2, for each dart thrown, such that u_1 and u_2 are both uniformly distributed between 0 and 1.

$\pi = 4R$

- If the distance from the origin of the point defined by (x,y) = (u_1,u_2) is smaller than the radius of the circle, that is, $u_1^2 + u_2^2 \leq 1$, then the simulated dart has hit within the quarter-circle.

- By repeating this process a large number of times, we can theoretically compute the value of π to any level of precision desired.

- The key to this type of Monte Carlo simulation is identifying an appropriate physical model for the system being studied.

Example Problem

Monte Carlo Simulation can be used in several application areas including forecasting models, financial & cost analysis, project management etc .

Choose any application area & provide a detailed example of Monte Carlo Simulation.

CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 26)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 25

Simulation & its advantages

Simulation Efficiency Consideration

Emulation

Monte-Carlo Simulation

Chapter # 7 (Cont'd)

SIMULATION MODELING

3) Discrete-Event Simulations

- A simulation using a discrete-state model of the system that changes as a function of time.

- Components of a discrete-event simulator:
  - An event scheduler
  - Simulation Clock and a Time-advancing Mechanism
  - Event processing routines
  - Initialization Routines
  - Event-generation
  - Recording and summarization of data

a) Event Scheduler

- It maintains a linked list of all pending events in their global time order.
- Processes the next event by removing it from the list and dispatching it to the appropriate event-processing routine.
- May be called several times during one event to schedule other new events.
- The scheduler also inserts new events into the appropriate point in the list based on their execution time.
- Events can be manipulated in various ways as follows:
  - Schedule event *X* at time *T*.
  - Cancel a previously scheduled event *X*.
  - Hold event *X* indefinitely (until it is scheduled by another event).

b) Simulation Clock and a Time-advancing Mechanism

- A global time variable records current simulation time.
- It can be updated by the scheduler using one of two approaches:
  - In the ***fixed-increment approach,*** the scheduler increments the global clock by some fixed amount.
  - It then checks the pending events.
  - If the scheduled time for any of the pending events matches the current time, all of these events are dispatched for execution.
  - After all these events have been processed, the scheduler again increments the global time variable.
  - The alternative ***event-driven approach*** allows the global time to jump to the value of the next event at the head of the pending-event list.
  - Here the value of the clock will change non-uniformly.
  - The event-driven approach is probably the most common in simulations of computer systems.

c) Event Processing Routines

- Each event in the system is simulated by its own event-processing routine. For example, routines to handle the two events of job arrivals and job departure.

- These routines may update the global state and they may generate additional events that must be inserted into the pending-event list.

- For example, a memory-access event in a simulation of a processor may result in two possible outcomes.
  - Cache-hit: the event may simply return the stored value.
  - Cache-miss: the memory-access event routine may generate a new event that returns the corresponding data value t_{miss} time units in the future, where t_{miss} is the time required to service a cache miss.

d) Initialization Routines

These set the initial state of the system state variables and initialize various random-number generation streams.

e) Event-Generation

Three possible techniques of Event-generation are as follows:

- Execution Driven
- Trace Driven
- Distribution Driven

Execution Driven

- The simulator executes a benchmark program.
- It models the necessary details of the system being tested.
- To simulate a program that does floating-point-arithmetic operations or any input/output operations, for instance,
    - the simulator must provide mechanisms for performing the necessary arithmetic and input/output operations.

Trace Driven

- A trace is a time-ordered record of events on a real system.
- Trace-driven simulations are generally used in analyzing or tuning resource management algorithms like Paging algorithms, CPU scheduling algorithms, deadlock prevention algorithms etc.
- In this technique, trace of resource demand patterns of key programs are obtained on a system.
- This trace can then be used as input to the simulation which models different algorithms.

Distribution Driven

- A distribution driven simulation is like a trace-driven simulation, except that input events are generated by the simulator itself to follow some predefined probabilistic function.

- For example, sending messages over a communication network could be modeled by using an exponential distribution to determine the amount of time that elapses between each message.

- The simulator then produces an output that would occur if the real system were driven by an application program that produced the same sequence of inputs.

f) Recording and Summarization of Data

- The simulator maintains appropriate event counts and time measurements.

- These values will be used at the end of simulation to calculate appropriate statistics.

- For example, if a memory system is being modeled, the simulator would probably count
  - the total number of memory references and the number of those references that result in cache misses.
  - These values can be used to estimate the cache-miss ratio.

4) Continuous-Event Simulations

- The state of the system varies continuously with time.

- Used in chemical simulations where the state of the system is described by the concentration of a chemical substance.

The Simulation Algorithm

```
Initialize global state variables
      Initialize the global time to 0
      Obtain the first input event
      while ((no more events) AND (time < maximum simulation
time limit))
      {     Advance the global time
            Remove the next event from the pending-event list
            Process the event
            {     Perform event-specific operations
                  Update global variables
                  Update simulation statistics
                  Generate new events triggered by this event
            }
      }
      Print the simulation results
```


GENERATION OF RANDOM NUMBERS

- Implementing a distribution-driven discrete-event simulation and Monte-Carlo simulation requires a supply of random numbers from probability distributions.
- A **random number generator** is an algorithm that produces sequences of random numbers that follow a specified probability distribution.

METHODS FOR GENERATION

Three methods for generating successive random samples ($t = t_1, t_2, \ldots$) from a probability distribution $f(t)$:

1. Inverse Transformation method.
2. Convolution method.
3. Acceptance-rejection method.

The Inverse Transformation Method

- Suppose it is desired to obtain a random sample t from the (continuous or discrete) PDF $f(t)$.
- The inverse method first determines a closed-form expression of the Cumulative Density Function, CDF, $F(t) = P\{y \leq t\}$, where $0 \leq F(t) \leq 1$, for all defined values of y.
- Given that R is a random value obtained from a uniform (0, 1) distribution, and assuming that F^{-1} is the inverse of F, the steps of the method are as follows:

- **Step 1**. Generate a (0, 1) uniform random number, R.
- **Step 2**. Set $F(x) = R$ and solve for x *as* $x = F^{-1}(R)$.

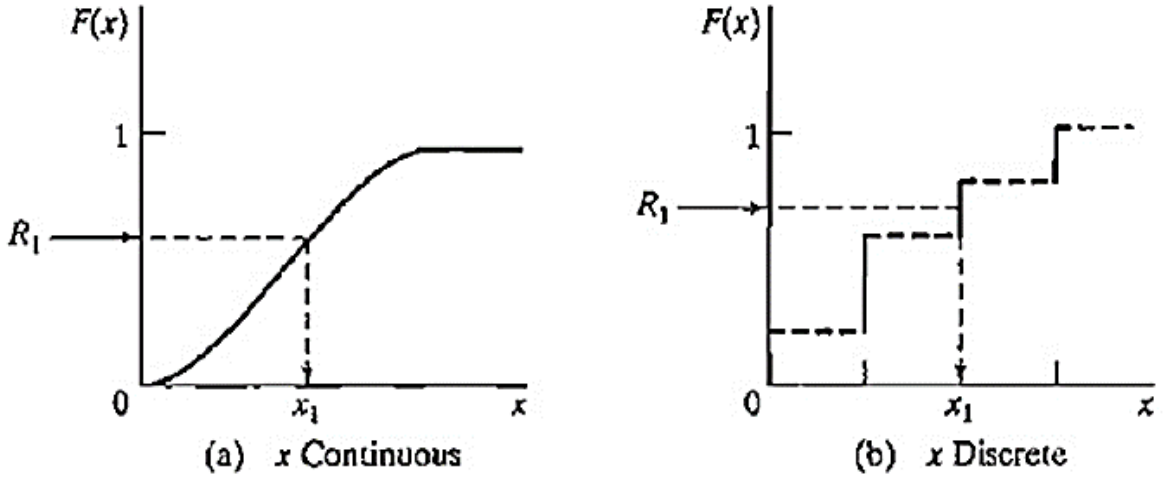


Fig 1: Sampling from a probability distribution by the inverse method

- For certain *continuous* distributions, the inverse transformation method can be implemented on a computer by first solving the equation $F(x) = R$ *analytically* for x.
- The CDF for the **exponential distribution** is

$$F(t) = 1 - e^{-\lambda t}, \text{ for } x \geq 0,$$

- Setting F(t) = R thereby yields

$$1 - e^{-\lambda t} = R, \Rightarrow e^{-\lambda t} = 1 - R,$$

- Therefore, taking the natural logarithm of both sides gives

$$\ln e^{-\lambda t} = \ln (1 - R), \quad \Rightarrow \quad -\lambda t = \ln (1 - R),$$

which yields $$t = \frac{\ln(1-R)}{-\lambda}$$

- Note that 1 – R is itself a uniform random number.
- In terms of simulation, the result means that arrivals are spaced t time units apart.
- For example, for $\lambda = 4$ customers per hour and $R = .9$, the time period until the next arrival occurs is computed as:

$$t = \frac{\ln(1 - 0.9)}{-4} = 0.577\ hour = 34.5\ minutes$$

- The values of R used to obtain successive samples must be selected *randomly* from a uniform (0, 1) distribution.

CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 27)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 26

Discrete & Continuous Events Simulation

Simulation Algorithm

Generation of Random Numbers

Chapter # 8

PETRI NET-BASED PERFORMANCE MODELING

Prepared by: Ms. Fakhra Aftab (Lecturer, CISD, NEDUET)

INTRODUCTION

- Petri nets (PNs) provide a *graphical tool* as well as a *notational method* for the formal specification of systems.

- The systems PNs model tend to include more than simply an arrival rate and a service rate.

- Two important aspects in modeling the performance of computer systems (1) contention for resources and (2) synchronization between various concurrent activities.

- The former aspect can be adequately represented by Queuing Network Modeling, but the later cannot.

- Petri nets have long been used for modeling synchronization behavior of concurrent systems but not as completely as simulations.
- They act more like simulations in that they allow the modeler to examine single entities within the system, as well as their movement and effect on the state of entire system.
- Petri nets were first introduced in 1966 to describe concurrent systems.
- This initial introduction was followed by continual improvements for example,
  - the addition of timing to transitions,
  - priority to transitions,
  - types to tokens,
  - and colors depicting conditions on places and tokens.

GRAPHICAL REPRESENTATION

- Petri nets are usually represented graphically according to the following conventions:
  - *Places* are represented by *circles*,
  - *transitions* by *bars*,
  - *input function* by arcs directed from places to transitions,
  - *output function* by arcs directed from transitions to places, and
  - *markings* by small filled circles called *tokens*.

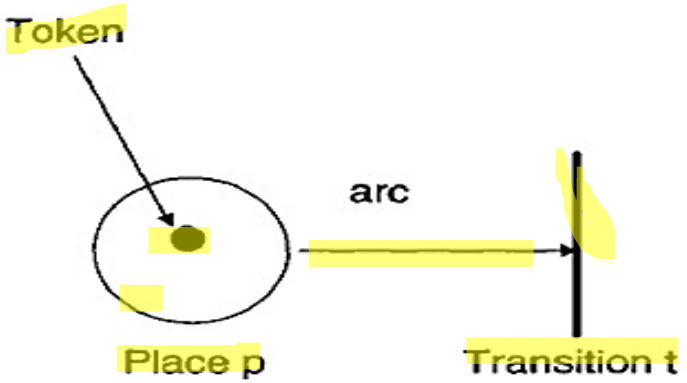


Fig 1: Basic Petri net components

Example

Fig 2 illustrates a simple Petri net with only one place and one transition.

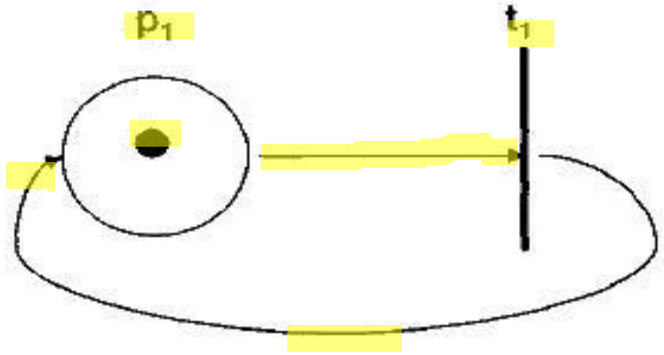


Fig 2: Example perpetual motion Petri net

- The former arc is input arc, while the later is an output arc.
- The placement of a token represents the active marking of the Petri net state.
- The Petri net shown in Fig 2 represents a net that will continue to cycle forever.

REPRESENTATION USING SET NOTATION

Using this notation, we can describe a PN as a 5-tuple, M= (P, T, I, O, MP)

- where P is the set of places, P = {P_1, P_2, … ,P_n}
- T is the set of transitions, T= {t_1, t_2, ... ,t_m}
- I $\subset$ P x T is the input function,
- O $\subset$ T x P is the output function, and
- MP $\subset$ P x Z^+ is a marking of the net,
- Here, Z^+ denotes the set of all nonnegative integers,
- I represents a bag of sets of input functions for all transitions,
- I = {I_{t1}, I_{t2}, …, I_{tm}}, mapping places to transitions,

- O represents a bag of sets of output functions for all transitions,
- O = $\{O_{t1}, O_{t2}, \ldots, O_{tm}\}$, mapping transitions to places; and
- *MP* represents the marking of places with tokens.
- The initial marking is referred to as MP_o.
- MP_o is represented as an ordered tuple of magnitude n, where n represents the number of places in our Petri net.
- Each place will have either no tokens or some integer number of tokens.

For example, the Petri net graph depicted in Fig 3 can be described using the previous notation as:

- *M = (P, T, I, O, MP)*
- *P = {p1, p2, p3, p4, p5}*
- *T = {t1, t2, t3, t4}*
- I(t1) = {p1}
- I(t2) = {p2, p3, p5}
- I(t3) = $\{p_3\}$
- I(t4) = $\{p_4\}$
- $O(t_1) = \{p_2, p_3, p_5\}$
- $O(t_2) = \{p_5\}$
- $O(t_3) = \{p_4\}$
- $O(t_4) = \{p_2, p_3\}$

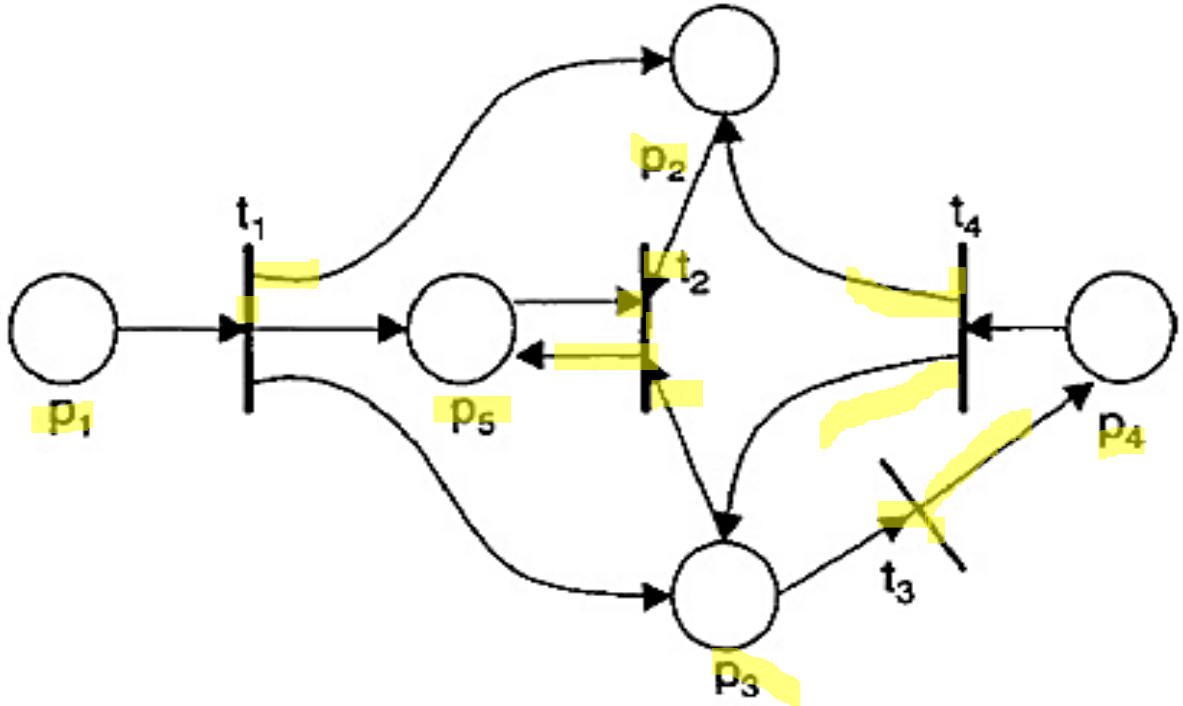


Fig 3: Petri net example

Dynamic Behaviour of Petri-Nets

- The dynamic behavior of a Petri net is described by the sequence of transition firings.
- The firing rules are as follows: Let *t* be a transition with incoming arcs from places ip_1 ,...,ip_K and outgoing arcs to places op_1,...:op_L for some K,L > 1.
- Then *t* can fire if and only if each of the K input places contains at least one token.
- As a result of firing, one token will be removed from each of the K input places, and one token will be added to each one of L output places.

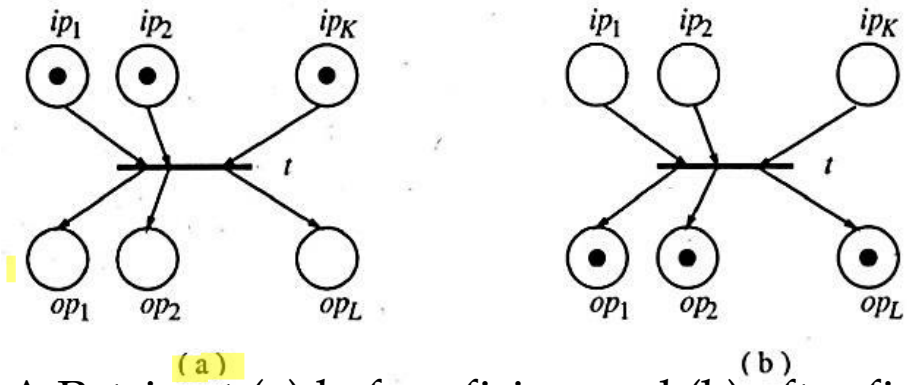


Fig 4: A Petri net (a) before firing and (b) after firing

DUAL OF A PETRI-NET

- The *dual* of a Petri net: transitions changed to places and places changed to transitions.

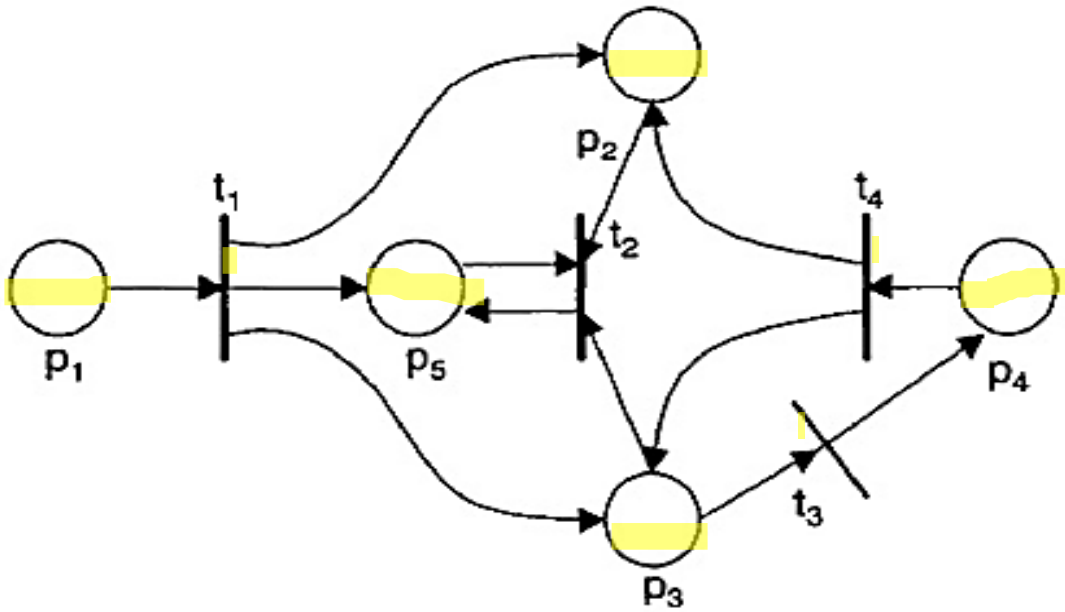


Fig 3: Petri Net Example

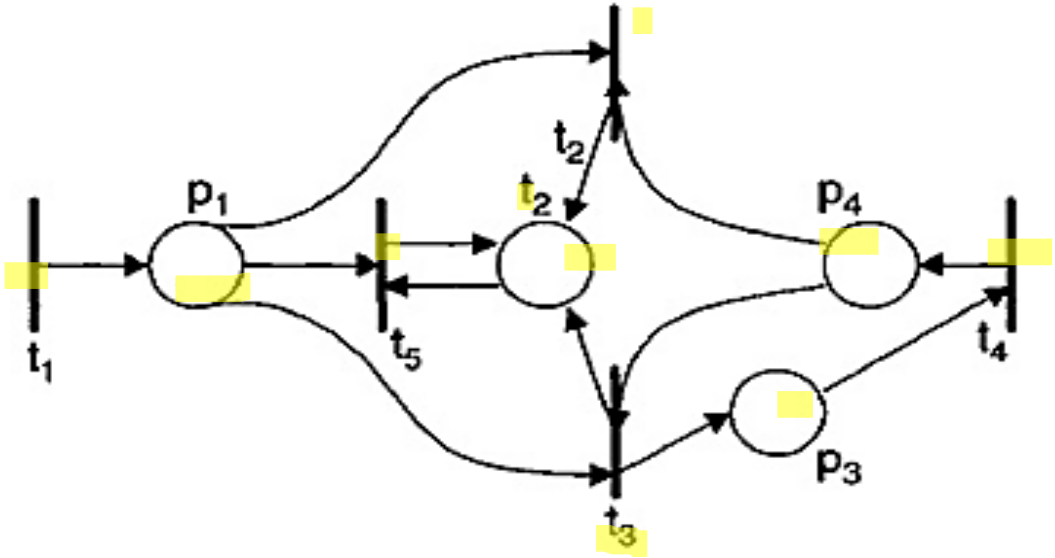


Fig 5: Dual of Petri Net from Fig 3

CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 28)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 27

Petri Nets - Introduction

Graphical & Set Notation Representation

Dynamic behavior of Petri-Nets

Dual of a Petri-Net

Chapter # 8 (Cont'd)

PETRI NET-BASED PERFORMANCE MODELING

Inverse of Petri Net

The *inverse* of a Petri net keeps all places and transitions the same and switches input functions with output functions.

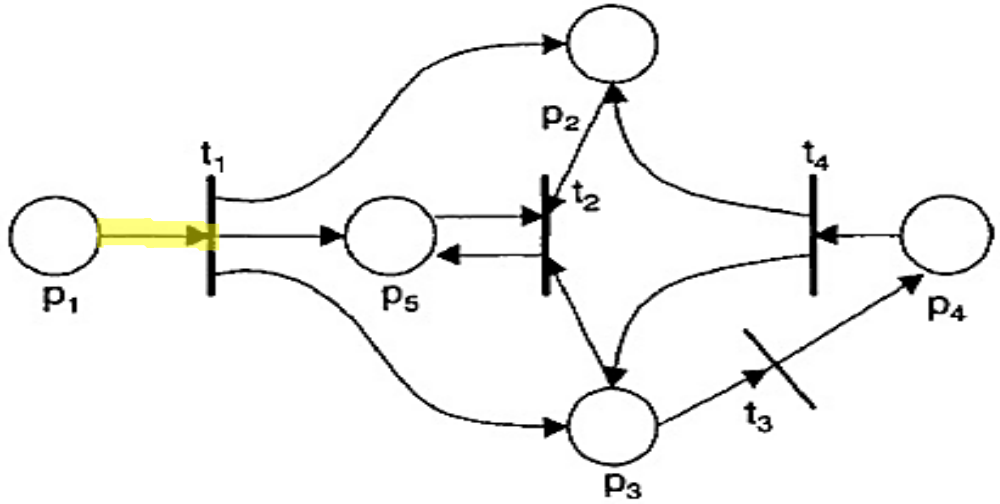


Fig 3: Petri net example

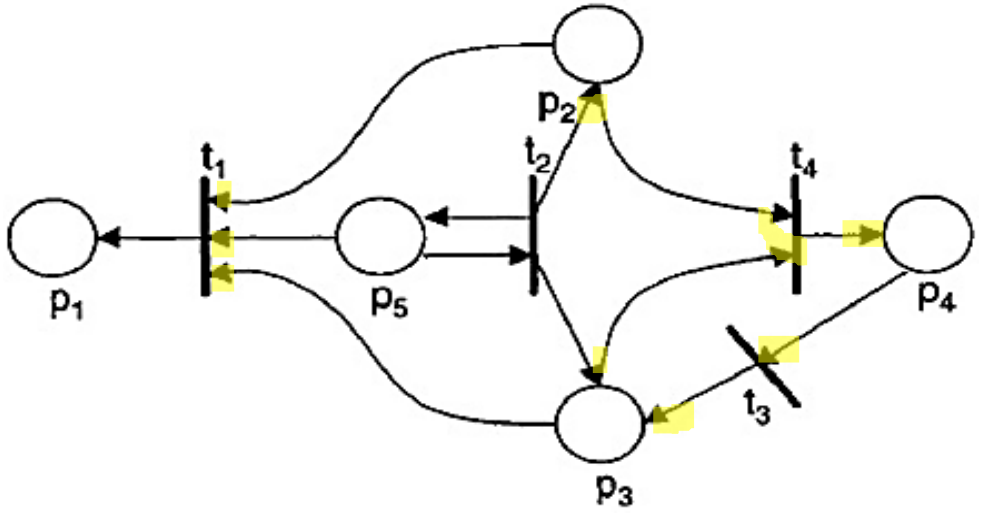


Fig 6: Inverse of Petri Net from Fig 3

Petri Nets as Multi-graph

Petri nets are defined also as *multi-graphs*, since a place can represent multiple inputs and/or outputs from or to a transition.

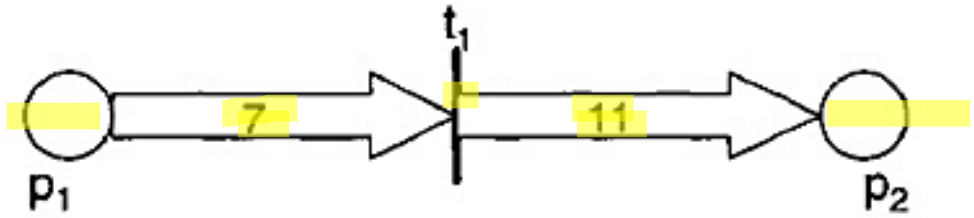


Fig 7: Multipath arc

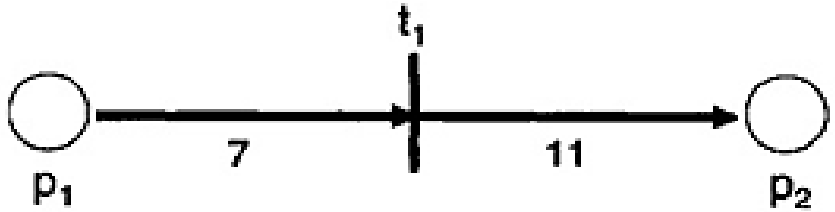


Fig 8: Multipath arc as bold line

State of a Petri Net

- Petri nets have a *state* defined by the cardinality of tokens and their distribution throughout the places in the Petri net.
- Marking represented as a function, μ (or MP), as follows:

$$\mu: P \rightarrow Z^+$$

- The marking, μ, can also be defined as an *n* vector.

$$\mu = (\mu_1 , \mu_2 , \mu_3, \ldots, \mu_n)$$

Where $n = |P|$ and each $\mu_i \in Z^+$, i = 0, …, n and $\mu(p_i) = \mu_i$.

- Therefore, the true representation of a marked Petri net is:

$$M= (P, T, I, O, \mu_t)$$

where μ_t represents state of Petri net at time *t,* where $t \in Z^+$.

- Set of all possible markings for a Petri net with n places
  - the set of all n vectors, N^n,
  - N represents all possible states and *n* the no. of places.
- The number of tokens that may be assigned to a place is unbounded.

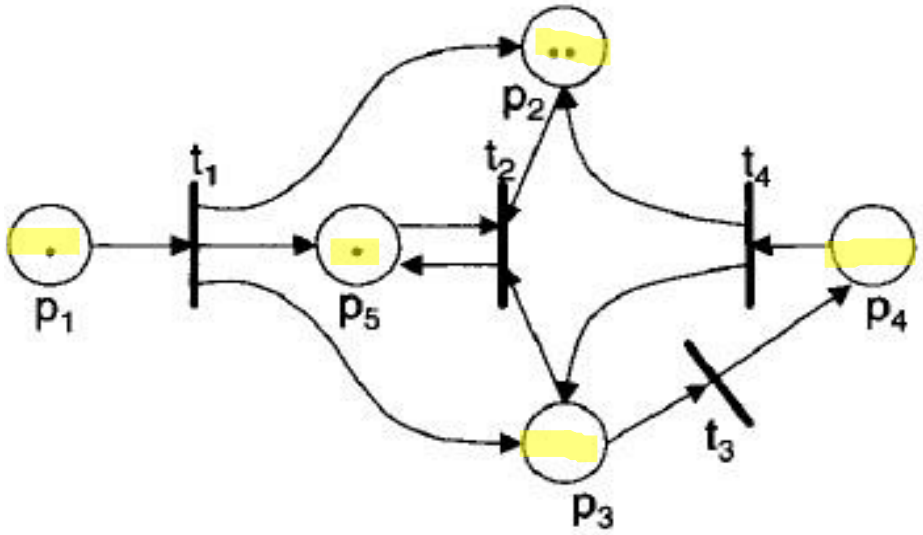


Fig 9: Marked Petri net

- The marking for the Petri net shown in Fig 9 represented as a vector would be $\mu_t = (1, 2, 0, 0, 1)$.

Classical Petri Net

- The classical PNs do not convey any notion of time.
- The exact moment of firing can be pictured as occurring as a clock signal in a computer system.

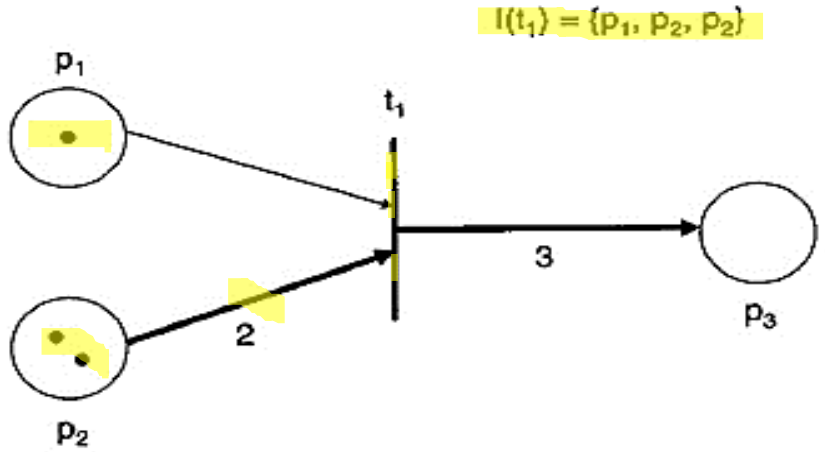


Fig 10: Enabled transition

Marking $\mu_o = (1,2,0)$

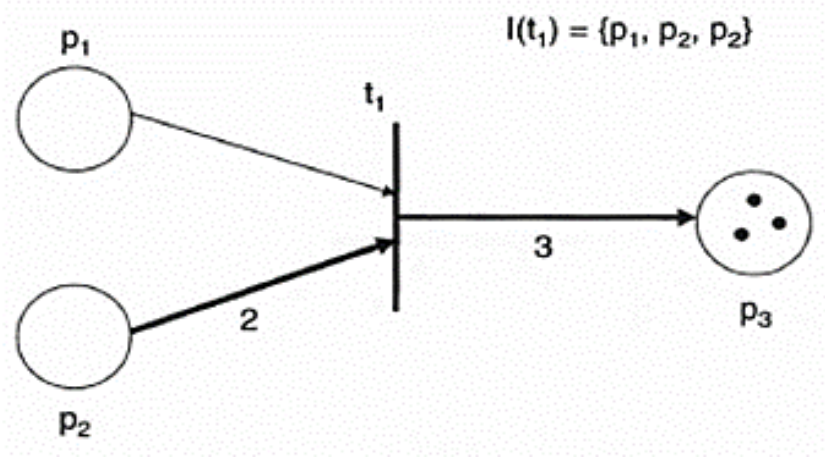


Fig 11: New Petri net state

Marking $\mu_1 = (0,0,3)$

- Input function $I(t_1) = \{P_1, P_2, P_2\}$ and Output function $O(t_1) = \{P_3, P_3, P_3\}$

State Space

- The collection of all possible states of a Petri net.

- Next-state function, δ applied to a Petri net state as follows:

$$\delta(\mu_i\{t\}) = \mu_{i+1}$$

- The set *{t}* represents the set of all enabled transitions within this Petri net.

- If a transition not enabled, then this function is undefined.

Petri Nets and the Modeling of Computer Systems

- PN used for modeling real systems sometimes referred to as *Condition/Events* nets.

- Places identify conditions of parts (working, idle, queuing, failed), and transitions describe the passage from one condition to another (end of a task, failure, repair ...).

- An event occurs (a transition fires) when all conditions satisfied.

- The number of tokens in a place used to identify the number of resources lying in the condition denoted by that place.

Concurrency (Parallelism)

- In reliability modeling, the PN of Fig 12 can represent two components C_1 and C_2 in parallel redundancy.
- p_1 & p_3 represent working condition, p_2 & p_4 the failed condition and t_1 & t_2 the event of failure of C_1 & C_2 respectively.

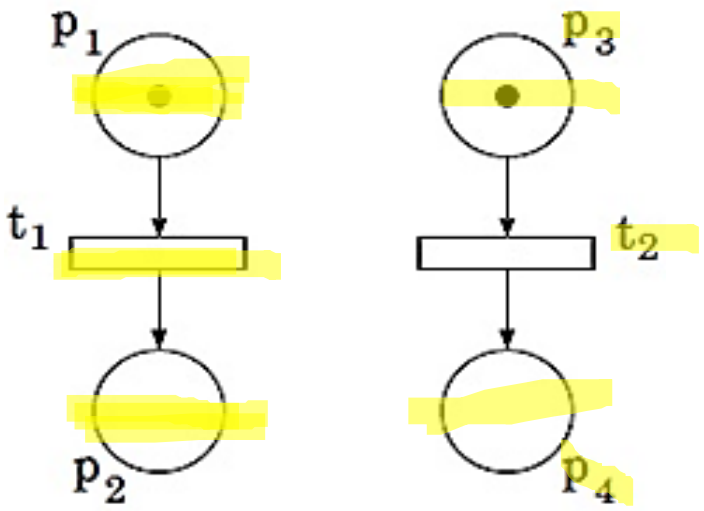


Fig 12: PN modeling two parallel activities

Synchronization

- Both the routines of a parallel program should be terminated before the program execution can proceed.
- The synchronization activity modeled in Fig 13 by means of t_3 whose firing requires a token both in p_2 and p_4.

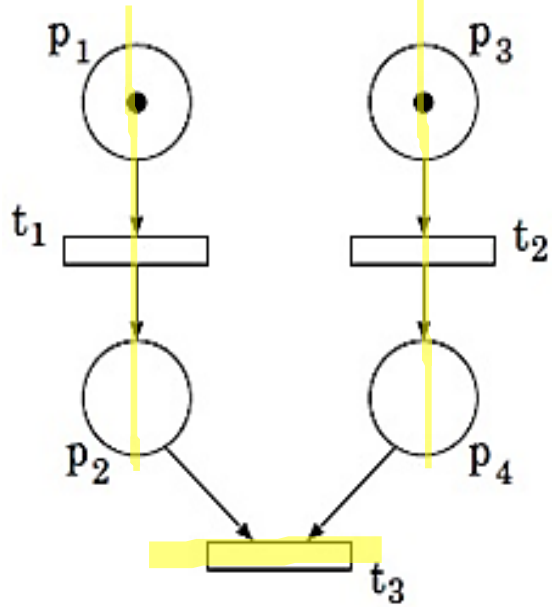


Fig 13: PN modeling two parallel activities with synchronization

Limited Resources

- This is a typical factor influencing the performance of computer systems.
- A PN representation of a buffer with limited size.

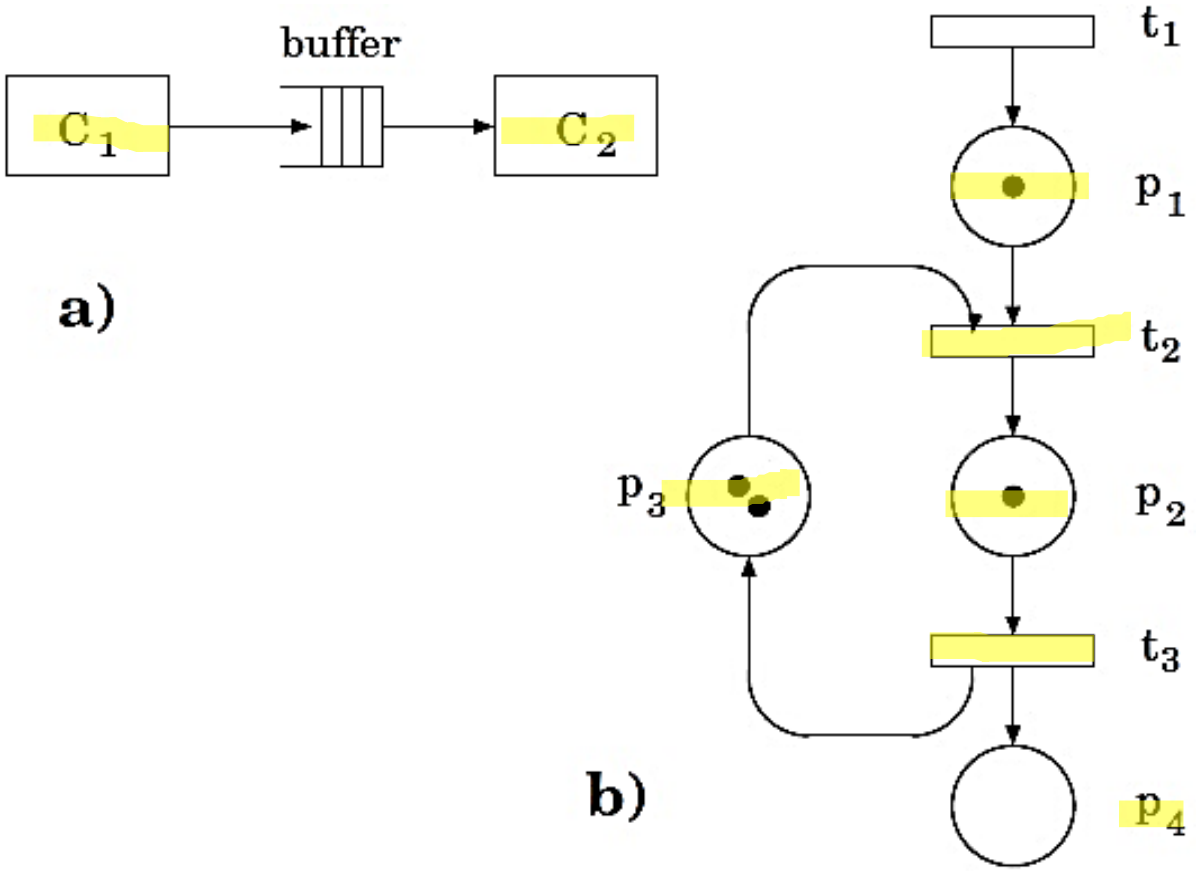


Fig 14: Block diagram and PN of a buffer with finite size.

The Bounded Buffer Producer/Consumer Problem

- A realistic situation is obtained by considering a buffer of *limited capacity* (Fig 15).

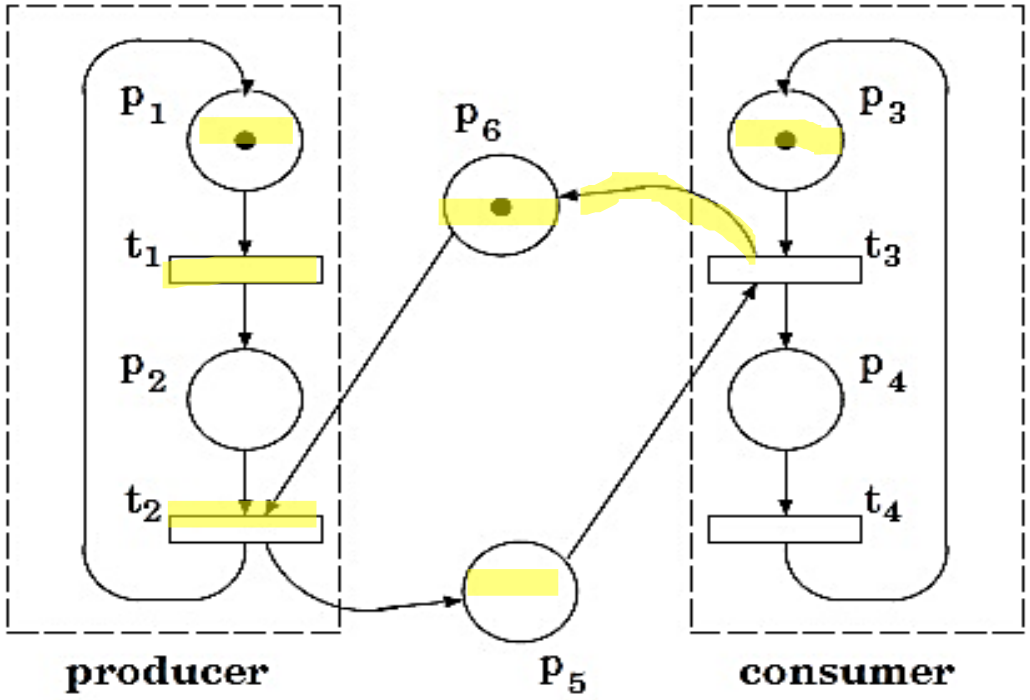


Fig 15: The producer/consumer problem with finite buffer

CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 29)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 28

Inverse of Petri-Net

Petri-Net as Multi-Graph

State of Petri-Net

Classical Petri-Net

Modeling of CS via Petri-Nets (Concurrency, Synchronization, Limited Resources)

Chapter # 8 (Cont'd)

PETRI NET-BASED PERFORMANCE MODELING

Mutual Exclusion (Conflict)

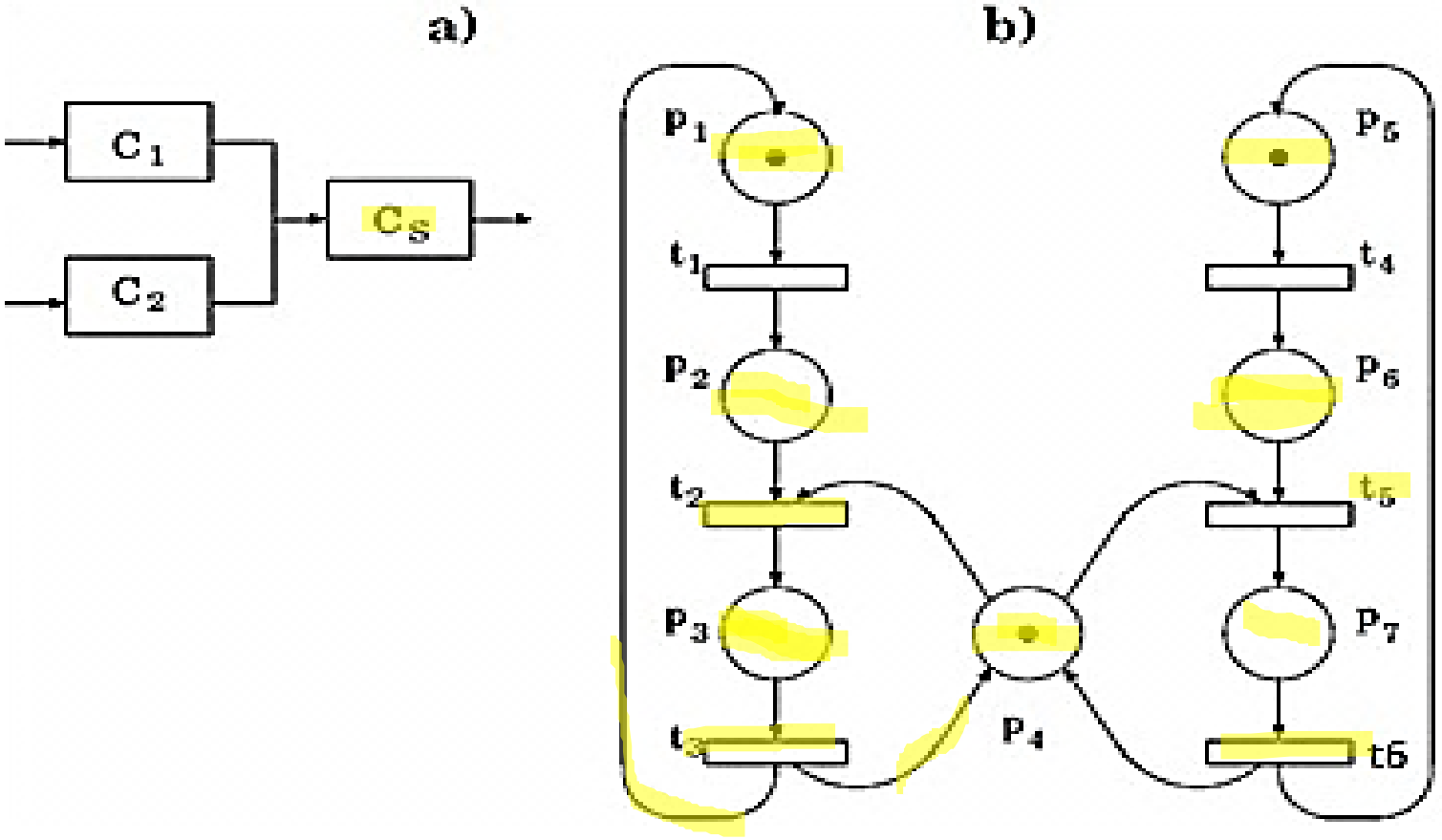


Fig 16: The Mutual Exclusion Problem

Logical Conditions

- It is often desirable to *model logical conditions*.
- e.g., to only fire a transition when there are more than *n* tokens in a place.

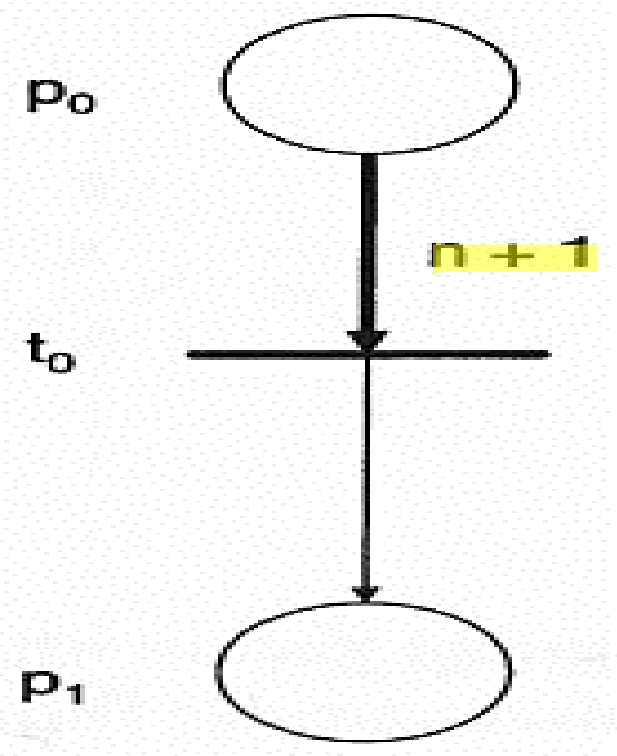


Fig 17: Petri net component to test condition greater than M

Inhibitor Arcs

- The inhibition function usually represented by *circle-headed arcs*.
- Modifies the enabling rules so that the transition fires only if p_j does not contain tokens.

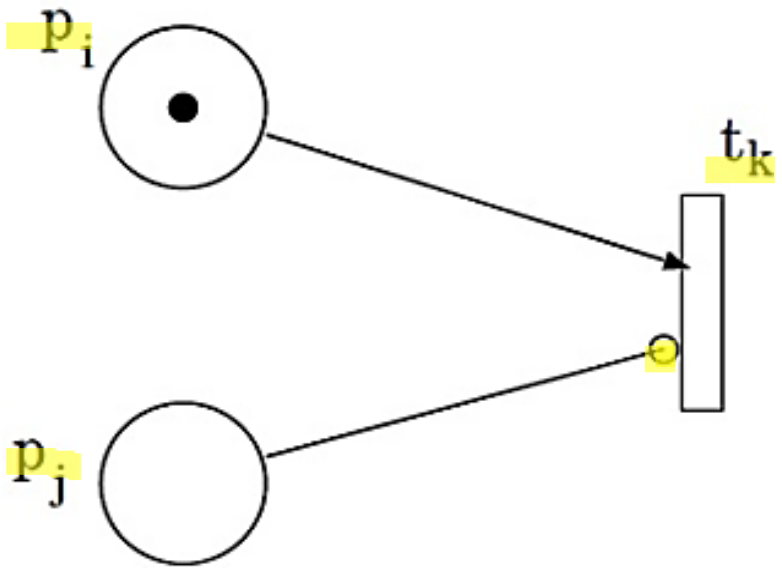


Fig 18: Inhibitor Arc

- To test for the condition of equal to some value but not greater than the value, we can use an inhibitor of arity $n + 1$ to block a transition if there are more than n tokens in place 0.

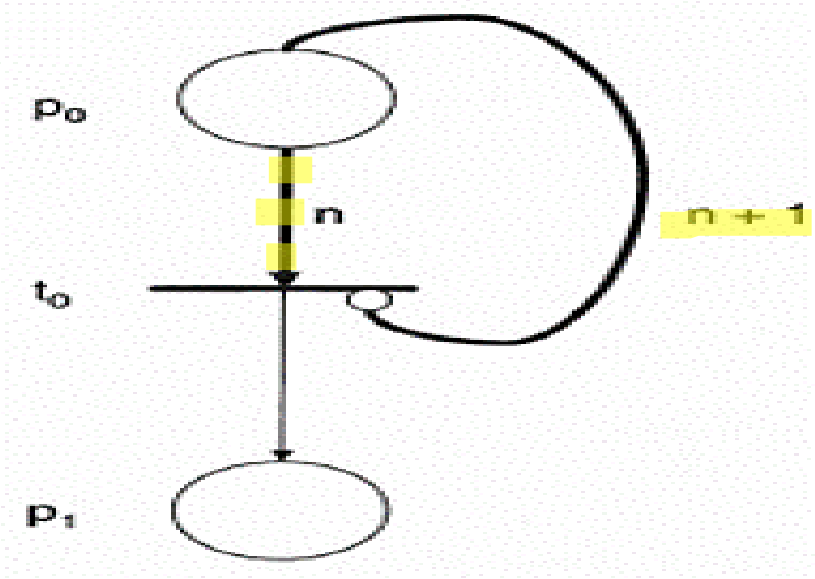


Fig 19: Petri net component to test condition equal but not greater than M

- If we wish to test for *less than n items* and remove the items, we could use the Petri net shown in Fig 21.

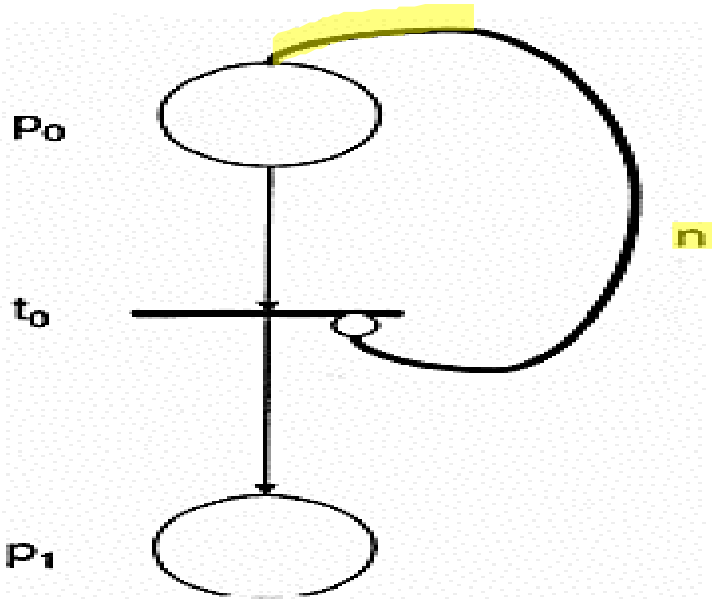


Fig 20

Modeling *conflict* and *concurrency*

- An initial marking, μ = (1,1,0,0,0), results in transitions t_1 and t_2 being enabled, the condition of concurrent transitions.
- If t_1 fires first, then we have two transitions enabled, t_2 and t_3. This then depicts a *conflict*.

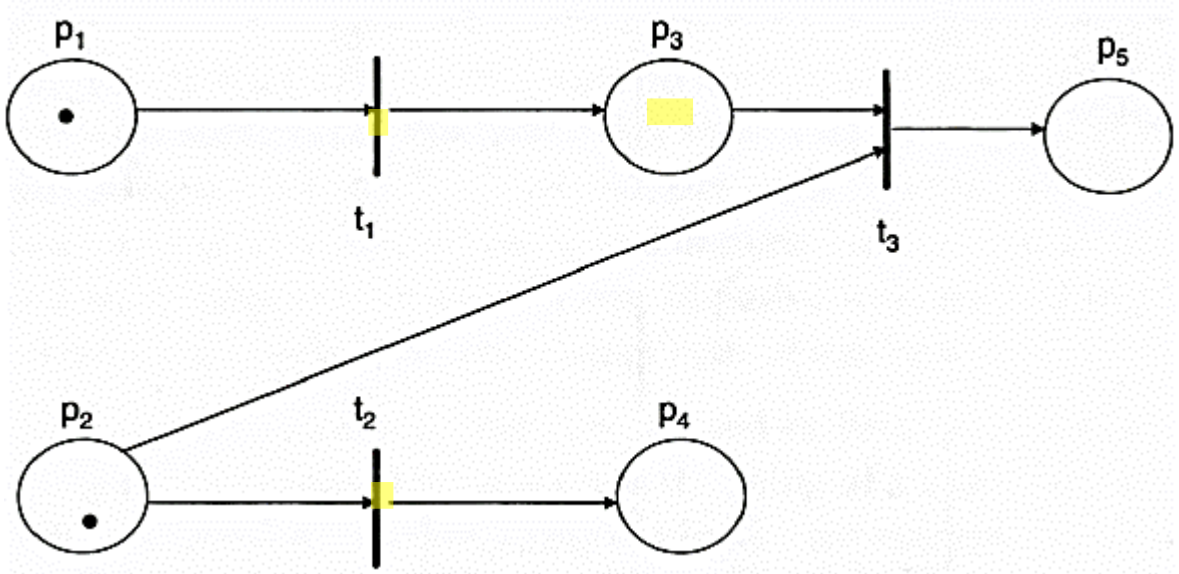


Fig 21: Petri net modeling *conflict* and *concurrency*

Reachability in Petri-Nets

- A Petri net state, **μ**, reachable from another state, **μ'**, if there is an integer number of intermediate steps from μ' to μ.
- e.g., μ_0 = (3, 0, 0, 0), and a target state μ' = (1,0,0,1).
- We can reach this target state in three firings of our net.

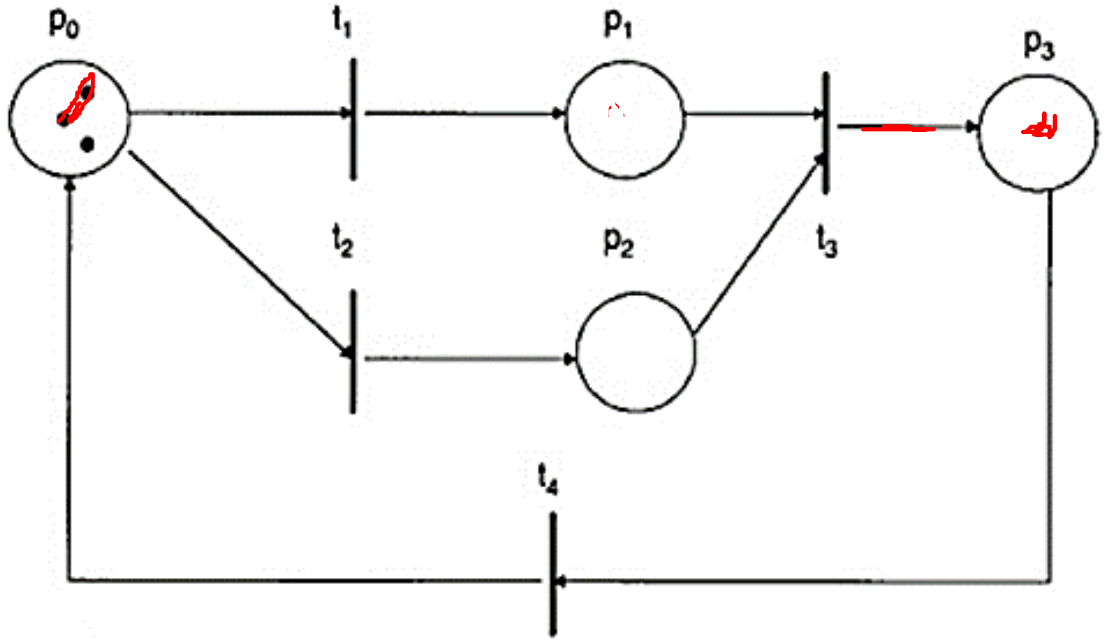


Fig 22: Petri net indicating *reachability*

Reversibility in Petri-Nets

- It is the property where, given some initial state, we can return back to this state, μ, in finite time.
- In Fig 23, μ_o, is not reversible, since we cannot get back to this state in a finite number of steps.

K-bounded Petri-Net:

- A Petri net defined to be k-place bounded if for all places, there are k or less tokens in each place for all possible states of the network.
- For example, Fig 23 is a three-bounded net.

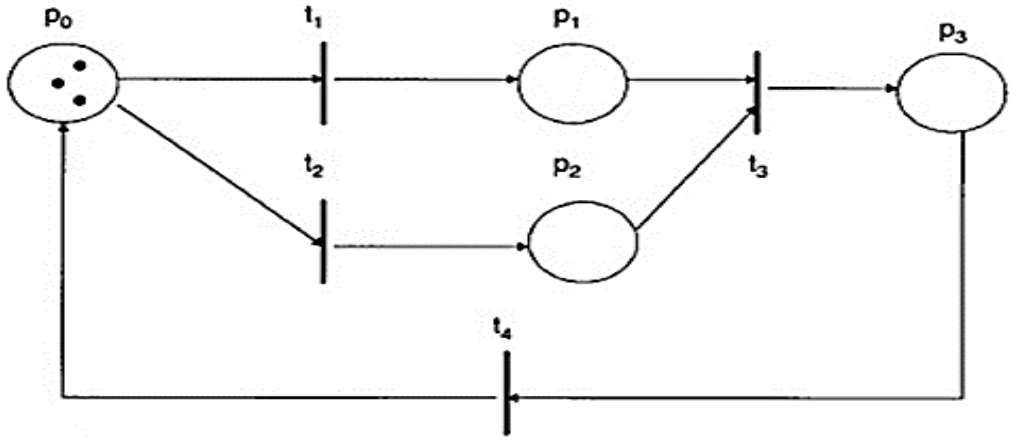


Fig 23: Petri net

Example Problem

Consider the Petri-Net in Fig 23 with the initial state as μ′ = (0,1,0,2).
Is it possible to return to this state after every *few* transitions ?
If is that so, provide the number of transitions. Is the state reversible ?

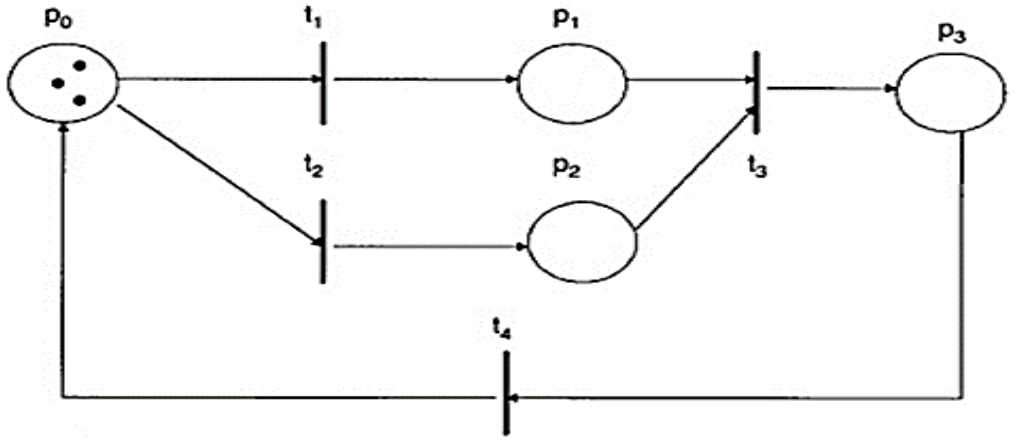


Fig 23: Petri net

Deadlocked Petri-net

- A Petri net is deadlocked if there are no transitions in the net that are enabled.
- Initial marking, $\mu_0 = (0,0,2,0)$.
- This marking results in no transitions being enabled.
- Conversely, a Petri net is considered *live* if there are any transitions enabled.

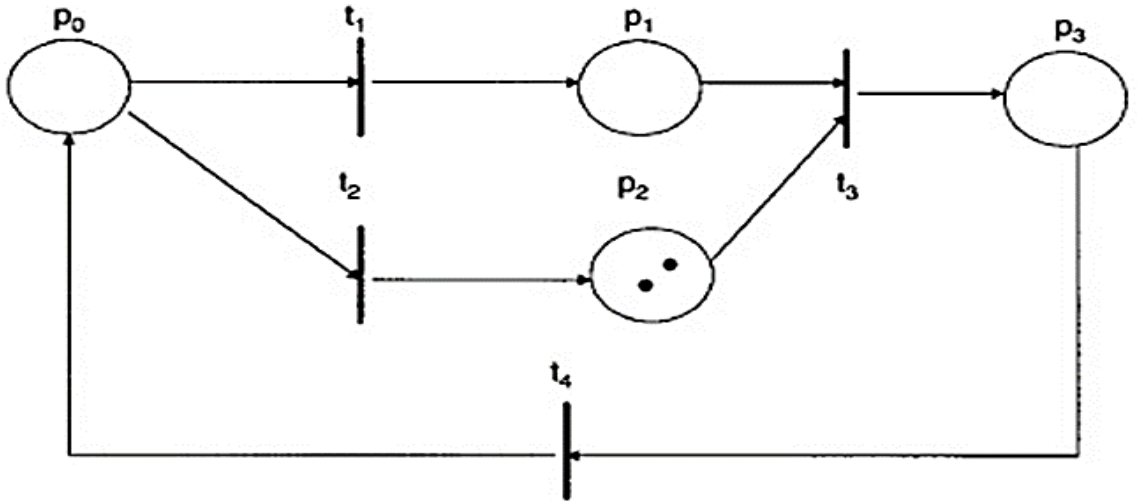


Fig 24: Deadlocked Petri-Net

Properties of Petri Nets

LIVENESS

- A transition is *live* if it is potentially firable in any marking of $R(M1)$.
- A transition is *dead* in M if it is not potentially firable; if the PN enters marking M the dead transition cannot fire any more.

SAFENESS

- A place is *safe* if the token count does not exceed 1 in any marking of $R(M1)$.
- A PN is safe if each place is safe.

Properties of Petri Nets (Cont'd)

BOUNDEDNESS

- A simple generalization of safeness.
- A PN is *k*-bounded if each place is *k*-bounded.

CONSERVATION

- A PN is strictly conservative if the total number of tokens is constant in each marking of $R(M1)$.

CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 30)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 29

Modeling of CS via Petri-Nets
(Mutual Exclusion, Logical Conditions, Conflict & Concurrency)

Reachability & Reversibility in Petri-Nets

Deadlocked Petri-Net

Properties of Petri-Net

Chapter # 8 (Cont'd)

PETRI NET-BASED PERFORMANCE MODELING

Example Problem 1

Consider the Petri-Network in Fig 25.

a) Provide the sets of places, transitions & initial marking.
b) Provide the sets of all the input and output functions w.r.t the transitions given in the figure.
c) Suppose at time T_1, transition t_1 is able to fire. What would be the marking M_1?
d) Identify M_2, M_3, M_4 and M_5 at times T_2, T_3 and T_4.

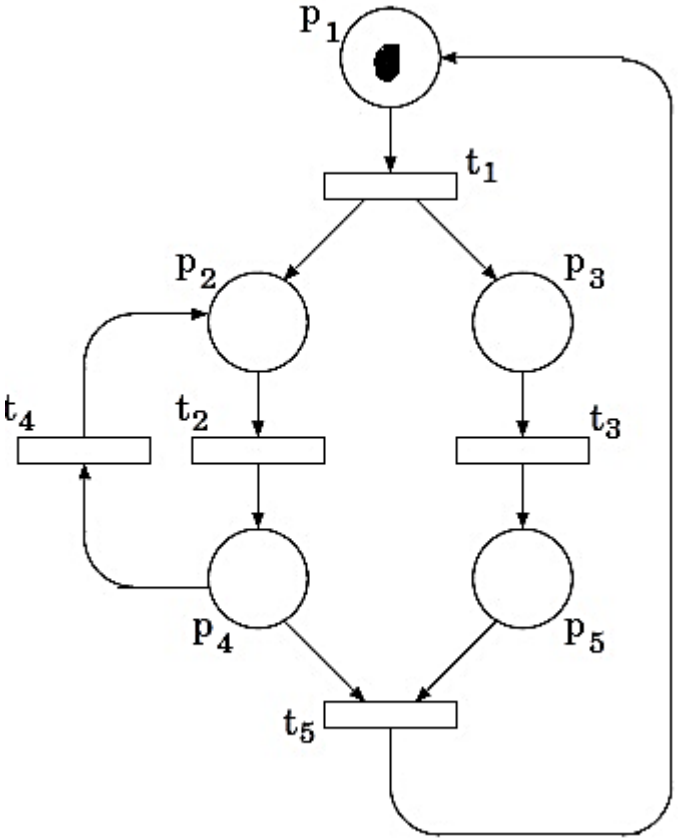
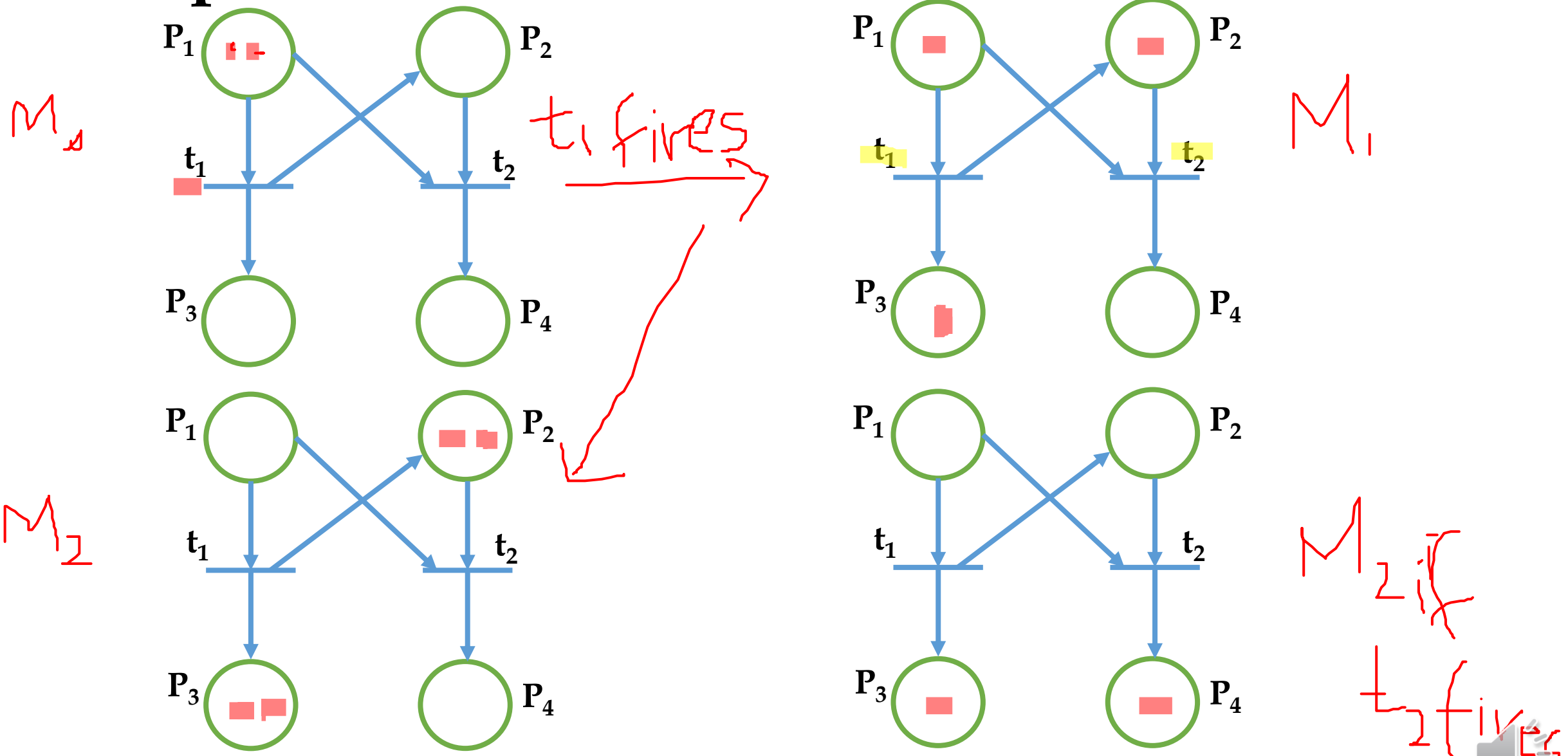


Fig 25

Example Problem 2



MATRIX ANALYSIS

- The input and output functions of a PN can be equivalently defined using a matrix notation.

- Let D^- denotes the input matrix. $\boldsymbol{D^-}$ is a (n_t *x* n_p) matrix, whose generic element $\boldsymbol{d_{ij}^-}$ is equal to the number of arcs connecting place $\boldsymbol{p_j}$ with transition $\boldsymbol{t_i}$.

- Similarly the output matrix $\boldsymbol{D^+}$ is a (n_t *x* n_p) matrix, whose generic element $\boldsymbol{d_{ij}^+}$ is equal to the number of arcs connecting transition $\boldsymbol{t_i}$ with place $\boldsymbol{p_j}$.

- The *incidence matrix D* is defined by the following relation:

$$\boldsymbol{D = D^+ - D^-}$$

Example

Following are the matrices D^-, $D+$ and D for the PN of Fig 26:

$$D^- = \begin{array}{c|ccccccc} & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \\ t_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ t_2 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ t_3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ t_4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ t_5 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ t_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

$$D^+ = \begin{array}{c|ccccccc} & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \\ t_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ t_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ t_3 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ t_4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ t_5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ t_6 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}$$

$$D = \begin{array}{c|ccccccc} & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \\ t_1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ t_2 & 0 & -1 & 1 & -1 & 0 & 0 & 0 \\ t_3 & 1 & 0 & -1 & 1 & 0 & 0 & 0 \\ t_4 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ t_5 & 0 & 0 & 0 & -1 & 0 & -1 & 1 \\ t_6 & 0 & 0 & 0 & 1 & 1 & 0 & -1 \end{array}$$

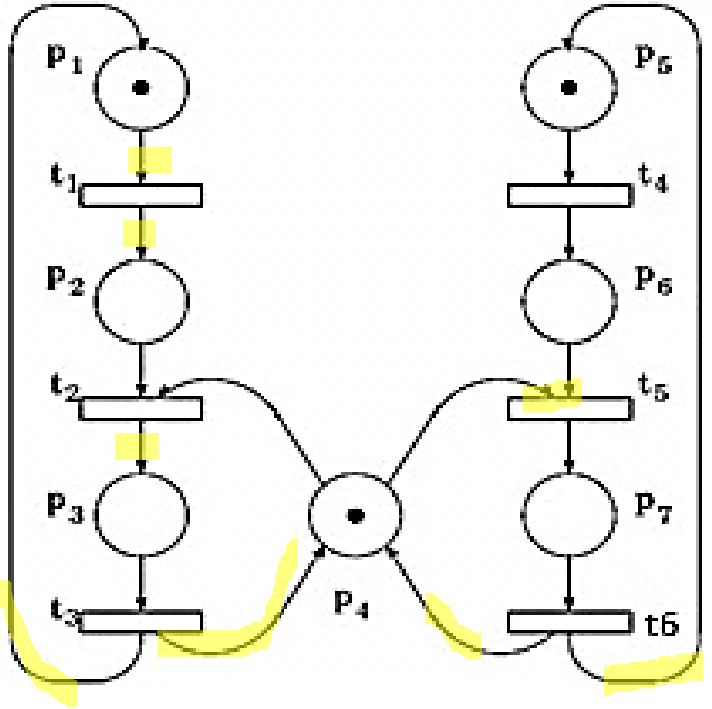


Fig 26

- Introducing the vector $\underline{e}_j$ which is a n_t-dimensional row vector with all the entries equal to 0 except entry j equal to 1.

- With this notation the execution rules of a PN becomes:
  - a transition t_j is enabled in marking M iff $\boldsymbol{M \geq \underline{e}_j D^-}$ (note that $\underline{e}_j D^-$ is the *j-th* row of D^-);
  - firing of t_j in M produces a marking M' given by:

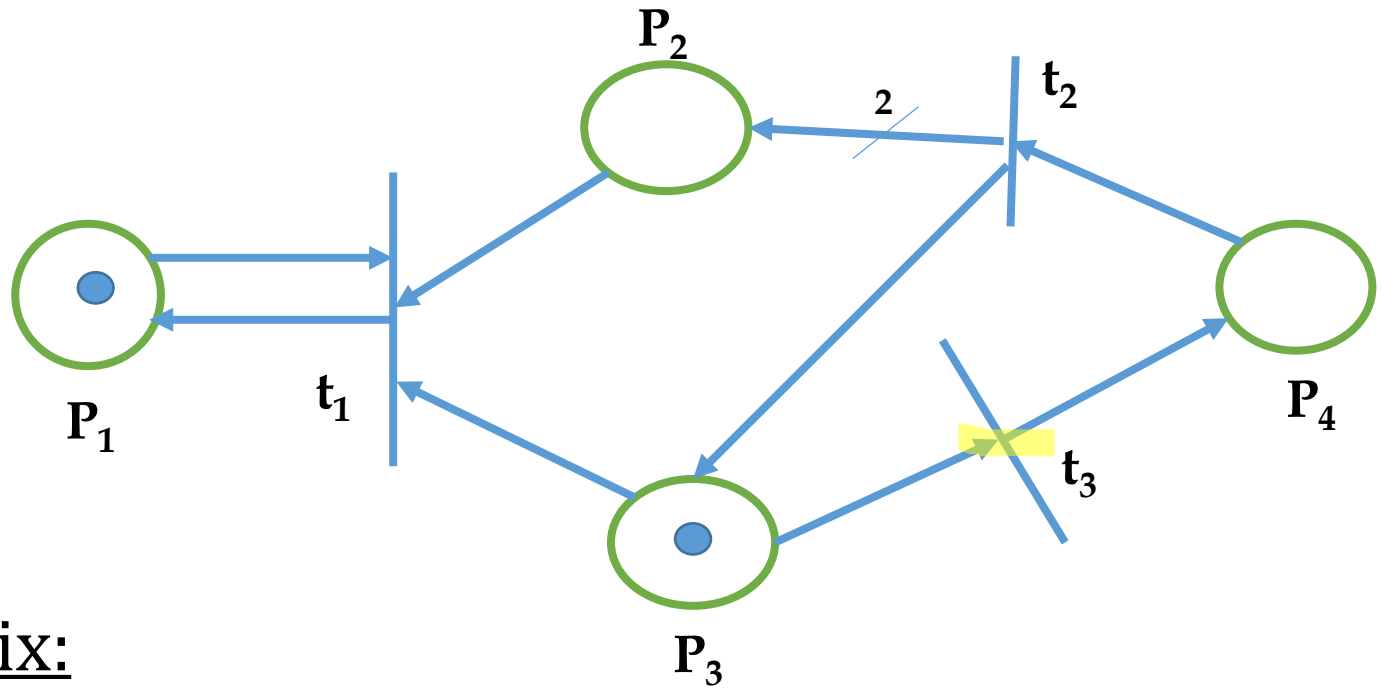
$$\boldsymbol{M' = M - \underline{e}_j D^- + \underline{e}_j D^+ = M + \underline{e}_j D}$$

- Given a PN with initial marking $\boldsymbol{M_1}$ and a firing sequence $t_i \rightarrow t_j \rightarrow t_k \rightarrow t_j \rightarrow t_i$, the marking obtained at the end of the sequence is given by the following matrix equation:

$$\boldsymbol{M_{fin} = M_1 + (\underline{e}_i + \underline{e}_j + \underline{e}_k + \underline{e}_j + \underline{e}_i)D}$$

Example Problem 3

Q) Consider the figure, find M' if t_3 is enabled to fire.



Fig

Input or Pre-Incidence Matrix:

$$D^- = \begin{vmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

Output or Post-Incidence Matrix:

$$D^+ = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Incidence Matrix:

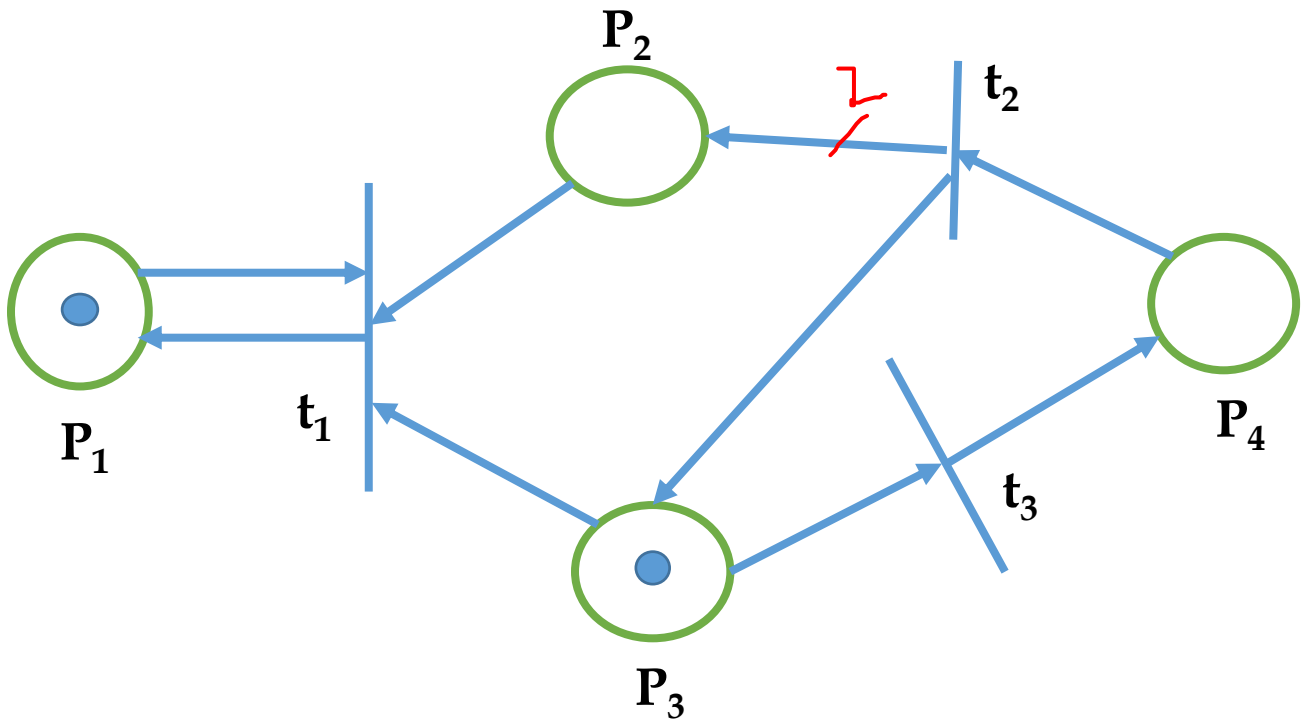
$$D = \begin{vmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{vmatrix}$$

Initial Marking:

$M_0 = [1\ 0\ 1\ 0]$

From the Fig, t_3 is enabled:

$e_j = (0, 0, 1)$



Fig

$$M' = M + e_j D$$

$$M' = [1\ 0\ 1\ 0] + [0\ 0\ 1] \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

$$M' = [1\ 0\ 0\ 1]$$

Q) Let the firing sequence be: $t_3 t_2 t_3 t_2 t_1$

$\Longrightarrow\ e_j = [1\ 2\ 2]$

If $M = [1\ 0\ 1\ 0]$,

$$M' = [1\ 0\ 1\ 0] + [1\ 2\ 2] \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

$$M' = [1\ 3\ 0\ 0]$$

Timed Petri Nets

- An activity in a real system takes finite time to perform its operation.
- e.g., to read a file from a disk, to execute a program, or to communicate with some other machine.
- Adding time provides Petri net modeler with another *powerful tool* to study the performance of computer systems.
- Time can be associated with transitions, selection of paths, waiting in places, inhibitors, and with any other component of the Petri net.
- The most typical way that time used in Petri nets is with transitions.
  - the firing of a transition can be viewed as the execution of an event being modeled – e.g., a CPU execution cycle.
- These timed transitions are represented graphically as a *rectangle or thick bars*.

The Semantics of the Firing

- When a transition becomes enabled, its clock timer is set and begins to count down.
- Once the timer reaches 0, the transition fires.
- In Fig 27, when token arrives at p_1, the timer for t_1 set to τ_1 and begins to count down.
- The decrement of the timer must be at a constant fixed speed for all transitions in the Petri net model.

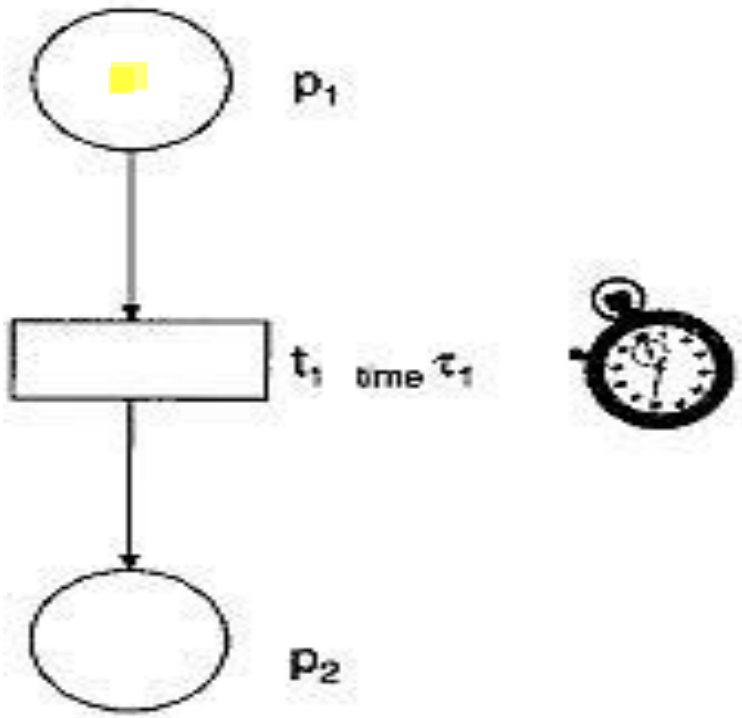


Fig 27: Timed Petri net

- A consideration to think about
  - what occurs when a transition becomes non-enabled due to the initial enabling token being used to ultimately fire a competing transition.
- This condition shown in Fig 28.
- If we assume the time for $\mathbf{t_1}$ is less than that for $\mathbf{t_2}$, then, when $\mathbf{p_1}$ receives a token, the two timers would begin counting down.
- At some time (τ_1) in the future, the timer for $\mathbf{t_1}$ would reach its zero value, resulting in the firing of $\mathbf{t_1}$.
- Since the token enabling $\mathbf{t_2}$ is now gone, $\mathbf{t_2}$ is no longer enabled and, therefore, its timer (τ_2) would stop ticking down.
- The question now is what to do with transition $\mathbf{t_2}$'s timer.

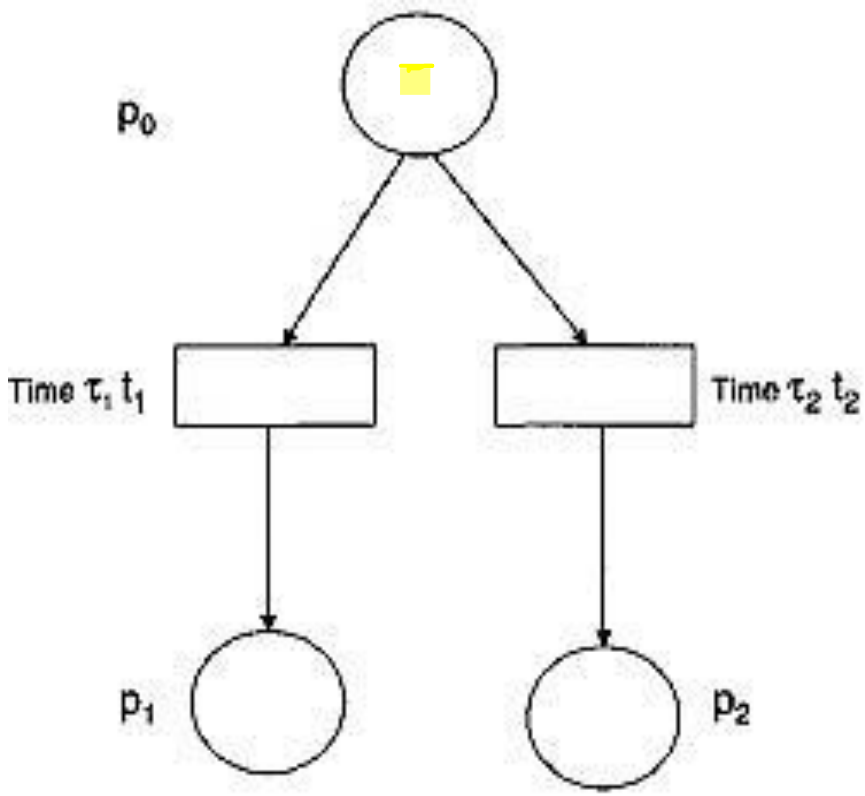


Fig 28: Timed Petri net with conflict

Two possibilities

1) The first is to simply reset the timer on the next cycle.

- In this case, unless place $\mathbf{p_1}$ has a state where it has more than one token present, transition $\mathbf{t_2}$ will never fire.

2) Allow transition t_2's timer to maintain the present clock timer value $\mathbf{(\tau_2 - \tau_1)}$.

- When the next token is received at p_1, if the remaining time in t_2's timer is less than t_1's timer, then t_2 will fire, leaving t_1 with the remaining time $\mathbf{(\tau_1 - (\tau_2 - \tau_1))}$.

The choice of which protocol to use will depend on the system one wishes to model.