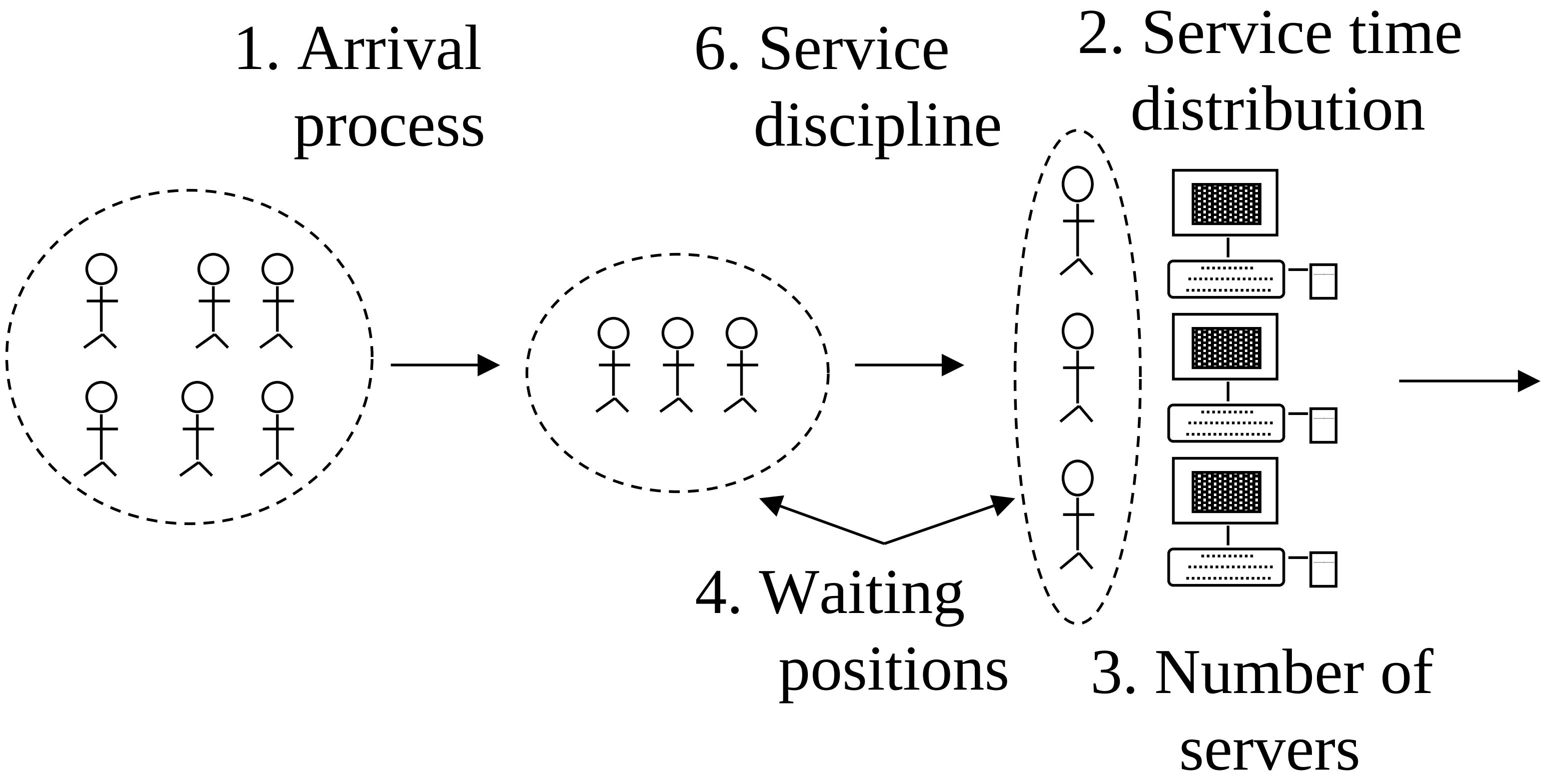
Introduction to Queueing Theory

- Queueing Notation
- Rules for All Queues
- Little's Law
- Types of Stochastic Processes

Queueing Models: What You will learn?

- What are various types of queues.
- What is meant by an *M/M/m/B/K* queue?
- How to obtain response time, queue lengths, and server utilizations?
- How to represent a system using a network of several queues?
- How to analyze simple queueing networks?
- How to obtain bounds on the system performance using queueing models?
- How to obtain variance and other statistics on system performance?
- How to subdivide a large queueing network model and solve it?

Basic Components of a Queue



Kendall Notation *A/S/m/B/K/SD*

- *A*: Arrival process
- *S*: Service time distribution
- *m*: Number of servers
- *B*: Number of buffers (system capacity)
- *K*: Population size, and
- *SD*: Service discipline

Arrival Process

- Arrival times: $t_1, t_2, \ldots, t_j$
- Interarrival times: $\tau_j = t_j - t_{j-1}$
- τ_j form a sequence of Independent and Identically Distributed (IID) random variables
- Exponential + IID ⇒ Poisson
- Notation:
  - M = Memoryless = Poisson
  - E = Erlang
  - H = Hyper-exponential
  - G = General ⇒ Results valid for all distributions

Service Time Distribution

- Time each student spends at the terminal.
- Service times are IID.
- Distribution: M, E, H, or G
- Device = Service center = Queue
- Buffer = Waiting positions

Service Disciplines

- First-Come-First-Served (FCFS)
- Last-Come-First-Served (LCFS)
- Last-Come-First-Served with Preempt and Resume (LCFS-PR)
- Round-Robin (RR) with a fixed quantum.
- Small Quantum ⇒ Processor Sharing (PS)
- Infinite Server: (IS) = fixed delay
- Shortest Processing Time first (SPT)
- Shortest Remaining Processing Time first (SRPT)
- Shortest Expected Processing Time first (SEPT)
- Shortest Expected Remaining Processing Time first (SERPT).
- Biggest-In-First-Served (BIFS)
- Loudest-Voice-First-Served (LVFS)

Common Distributions

- M: Exponential
- E_k: Erlang with parameter k
- H_k: Hyper-exponential with parameter k
- D: Deterministic $\Rightarrow$ constant
- G: General $\Rightarrow$ All
- Memoryless:
  - Expected time to the next arrival is always $1/\lambda$ regardless of the time since the last arrival
  - Remembering the past history does not help.

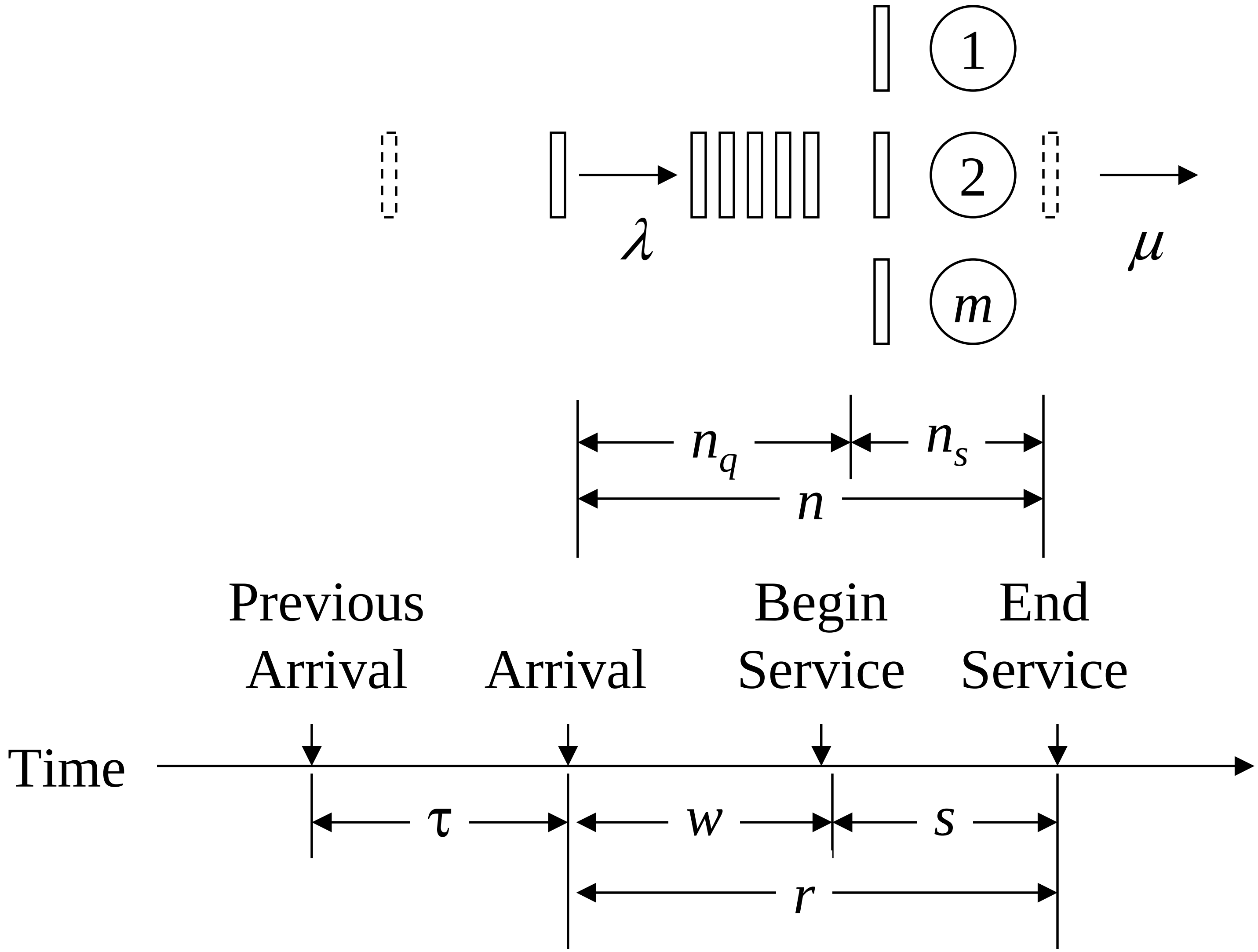
Example *M/M/3/20/1500/FCFS*

- Time between successive arrivals is exponentially distributed.
- Service times are exponentially distributed.
- Three servers
- *20* Buffers = *3* service + *17* waiting
- After *20*, all arriving jobs are lost
- Total of *1500* jobs that can be serviced.
- Service discipline is first-come-first-served.
- Defaults:
  - Infinite buffer capacity
  - Infinite population size
  - FCFS service discipline.
- $G/G/1 = G/G/1/\infty/\infty/FCFS$

Group Arrivals/Service

- Bulk arrivals/service
- $M^{[x]}$: x represents the group size
- $G^{[x]}$: a bulk arrival or service process with general inter-group times.
- Examples:
  - $M^{[x]}/M/1$: Single server queue with bulk Poisson arrivals and exponential service times
  - $M/G^{[x]}/m$: Poisson arrival process, bulk service with general service time distribution, and m servers.

Key Variables



Key Variables (cont)

- τ = Inter-arrival time = time between two successive arrivals.
- λ = Mean arrival rate = $1/E[\tau]$
  May be a function of the state of the system,
  e.g., number of jobs already in the system.
- s = Service time per job.
- μ = Mean service rate per server = $1/E[s]$
- Total service rate for m servers is $m\mu$
- n = Number of jobs in the system.
  This is also called **queue length**.
- Note: Queue length includes jobs currently receiving service as well as those waiting in the queue.

Key Variables (cont)

- n_q = Number of jobs waiting
- n_s = Number of jobs receiving service
- r = Response time or the time in the system
  = time waiting + time receiving service
- w = Waiting time
  = Time between arrival and beginning of service

Rules for All Queues

Rules: The following apply to *G/G/m* queues

1. Stability Condition:

$$\lambda < m\mu$$

   Finite-population and the finite-buffer systems are always stable.

2. Number in System versus Number in Queue:

   $n = n_q + n_s$

   Notice that n, n_q, and n_s are random variables.

   $E[n]=E[n_q]+E[n_s]$

   If the service rate is independent of the number in the queue,

   $Cov(n_q,n_s) = 0$

$$\mathrm{Var}[n] = \mathrm{Var}[n_q] + \mathrm{Var}[n_s]$$

Rules for All Queues (cont)

3. Number versus Time:
   If jobs are not lost due to insufficient buffers,
   Mean number of jobs in the system
   = Arrival rate $\times$ Mean response time
4. Similarly,
   Mean number of jobs in the queue
   = Arrival rate $\times$ Mean waiting time

This is known as **Little's law**.

5. Time in System versus Time in Queue

$$r = w + s$$

   r, w, and s are random variables.

$$E[r] = E[w] + E[s]$$

Rules for All Queues(cont)

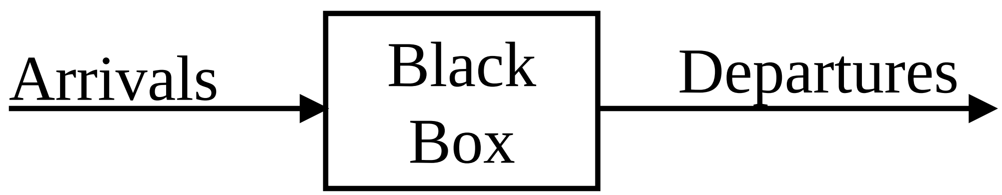
6. If the service rate is independent of the number of jobs in the queue,

$$Cov(w,s)=0$$

$$\mathrm{Var}[r] = \mathrm{Var}[w] + \mathrm{Var}[s]$$

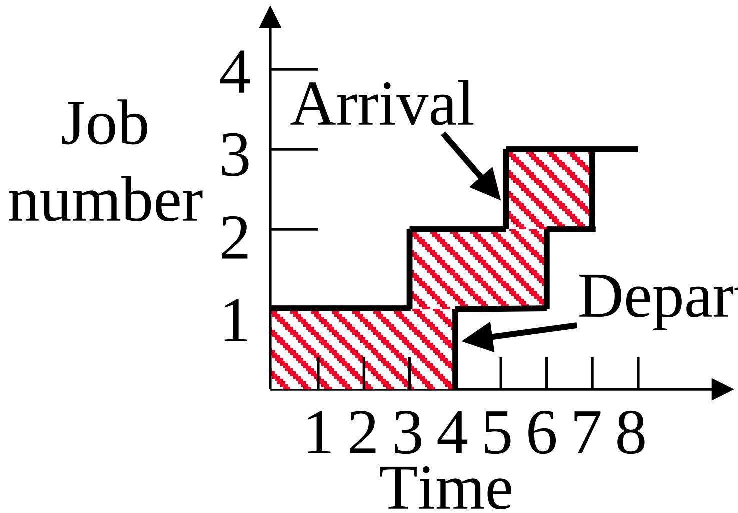
Little's Law

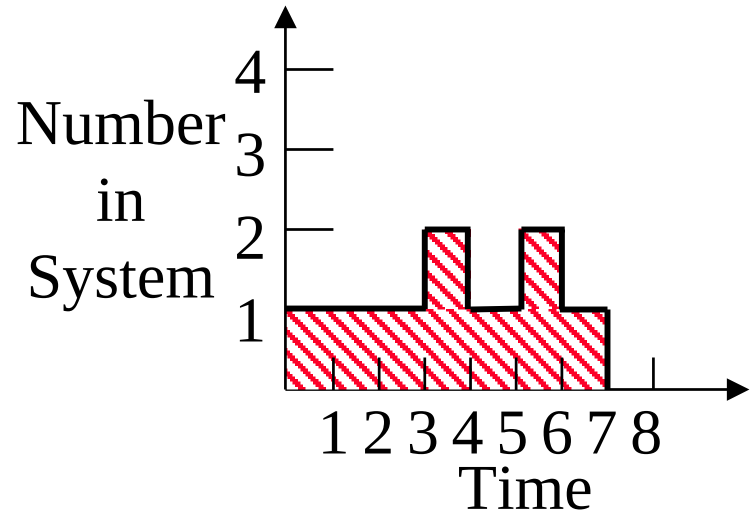
- Mean number in the system
  = Arrival rate × Mean response time
- This relationship applies to all systems or parts of systems in which the number of jobs entering the system is equal to those completing service.
- Named after Little (1961)
- Based on a black-box view of the system:



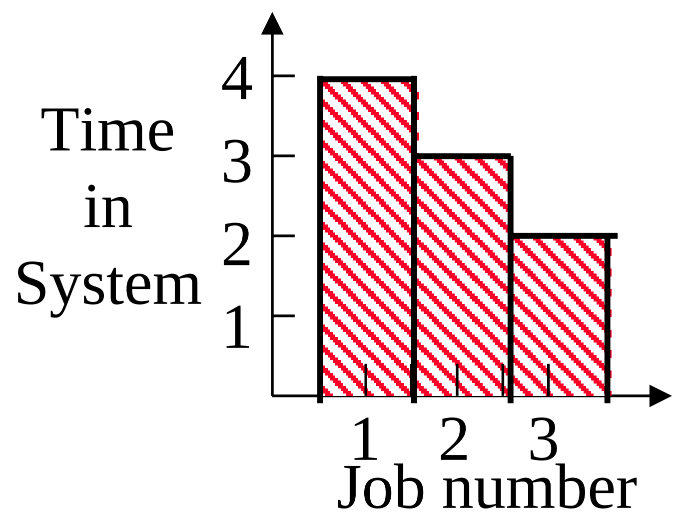
- In systems in which some jobs are lost due to finite buffers, the law can be applied to the part of the system consisting of the waiting and serving positions

Proof of Little's Law

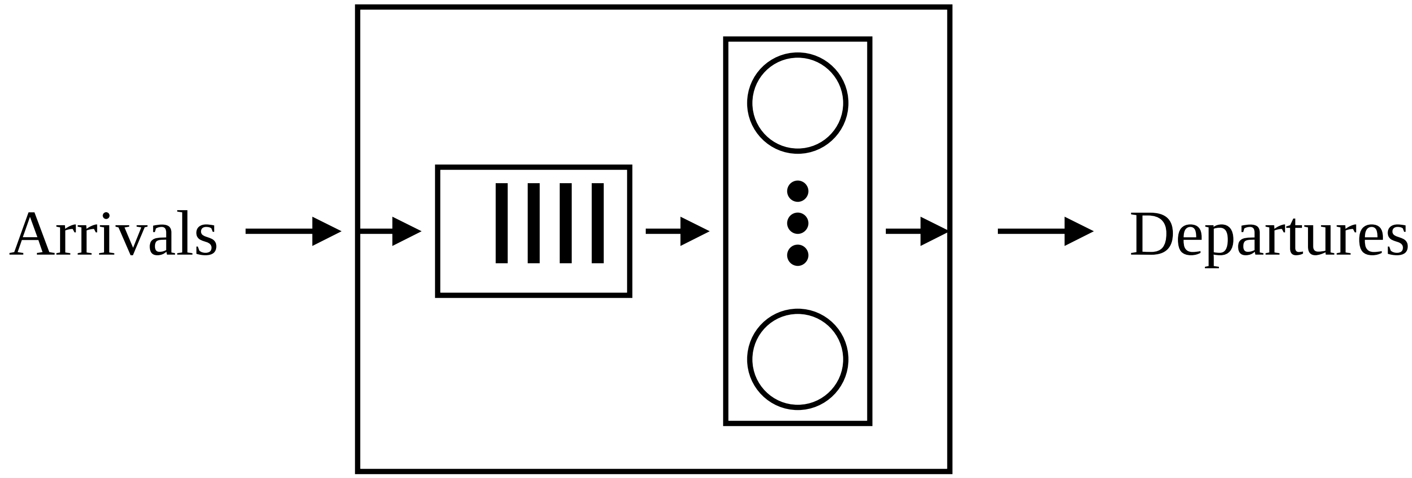




- If T is large, arrivals = departures = N
- Arrival rate = Total arrivals/Total time= N/T
- Hatched areas = total time spent inside the system by all jobs = J
- Mean time in the system= J/N
- Mean Number in the system
  $= J/T = \frac{N}{T} \times \frac{J}{N}$
  = Arrival rate× Mean time in the system



Application of Little's Law



- Applying to just the waiting facility of a service center
- Mean number in the queue = Arrival rate $\times$ Mean waiting time
- Similarly, for those currently receiving the service, we have:
- Mean number in service = Arrival rate $\times$ Mean service time

Example 30.3

- A monitor on a disk server showed that the average time to satisfy an I/O request was 100 milliseconds. The I/O rate was about 100 requests per second. What was the mean number of requests at the disk server?

- Using Little's law:

Mean number in the disk server

$= \text{Arrival rate} \times \text{Response time}$

$= 100 \text{ (requests/second)} \times (0.1 \text{ seconds})$

$= 10 \text{ requests}$

Stochastic Processes

- **Process**: Function of time
- **Stochastic Process**: Random variables, which are functions of time
- ***Example 1:***
  - *n(t)* = number of jobs at the CPU of a computer system
  - Take several identical systems and observe n(t)
  - The number *n(t)* is a random variable.
  - Can find the probability distribution functions for *n(t)* at each possible value of t.
- ***Example 2:***
  - *w(t)* = waiting time in a queue

Types of Stochastic Processes

- Discrete or Continuous State Processes
- Markov Processes
- Birth-death Processes
- Poisson Processes

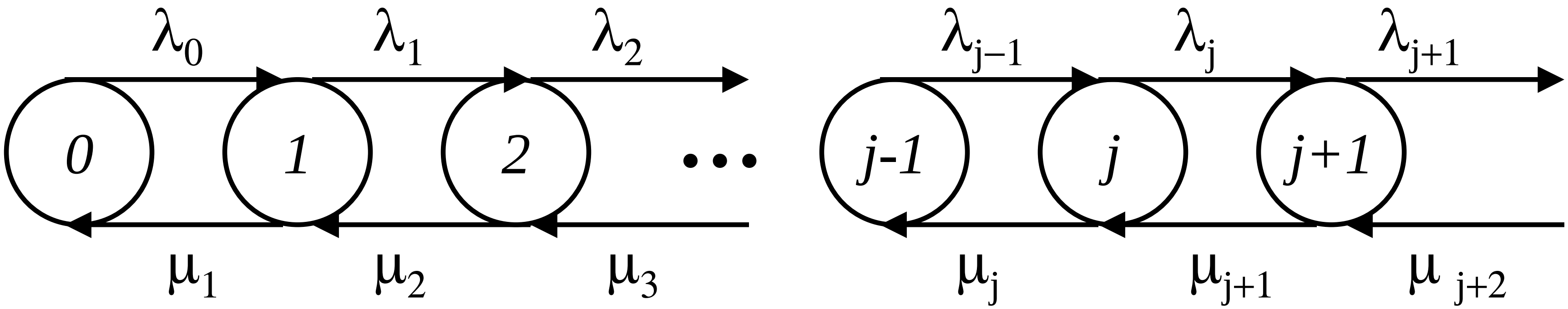
Discrete/Continuous State Processes

- Discrete = Finite or Countable
- Number of jobs in a system $n(t) = 0, 1, 2, ...$
- $n(t)$ is a discrete state process
- The waiting time $w(t)$ is a continuous state process.
- **Stochastic Chain**: discrete state stochastic process

Markov Processes

- Future states are independent of the past and depend only on the present.
- Named after A. A. Markov who defined and analyzed them in 1907.
- **Markov Chain**: discrete state Markov process
- Markov $\Rightarrow$ It is not necessary to know how long the process has been in the current state $\Rightarrow$ State time has a memoryless (exponential) distribution
- *M/M/m* queues can be modeled using Markov processes.
- The time spent by a job in such a queue is a Markov process and the number of jobs in the queue is a Markov chain.

Birth-Death Processes

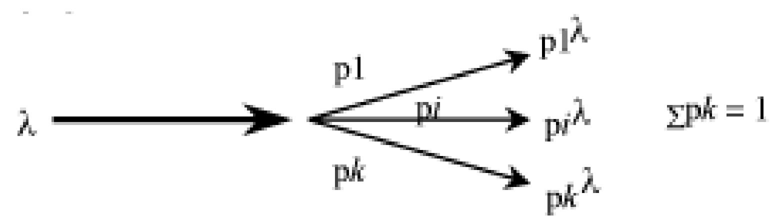


- The discrete space Markov processes in which the transitions are restricted to neighboring states
- Process in state *n* can change only to state *n+1* or *n-1*.
- Example: the number of jobs in a queue with a single server and individual arrivals (not bulk arrivals)

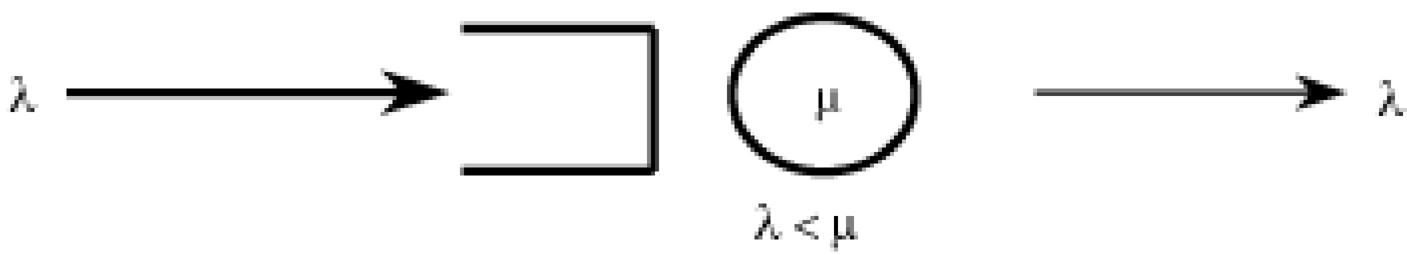
Poisson Processes

- Interarrival time *s* = IID and exponential
  ⇒ number of arrivals *n* over a given interval *(t, t+x)* has a Poisson distribution
  ⇒ arrival = Poisson process or Poisson stream
- Properties:
  - 1.Merging: $\lambda = \sum_{i=1}^{k} \lambda_i$

  - 2.Splitting: If the probability of a job going to *ith* substream is p_i, each substream is also Poisson with a mean rate of $p_i \lambda$

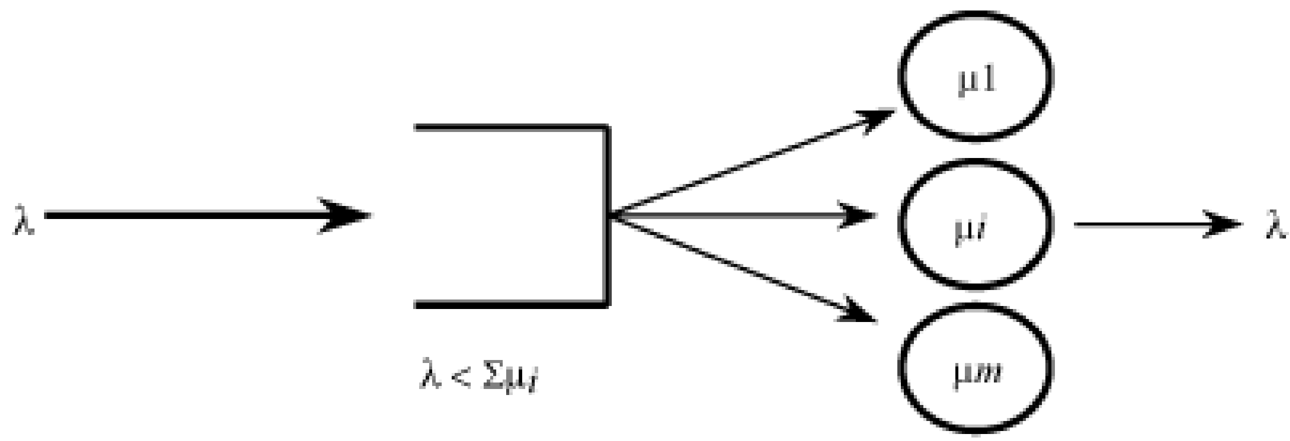


- 3.If the arrivals to a single server with exponential service time are Poisson with mean rate λ, the departures are also Poisson with the same rate λ provided $\lambda < \mu$.

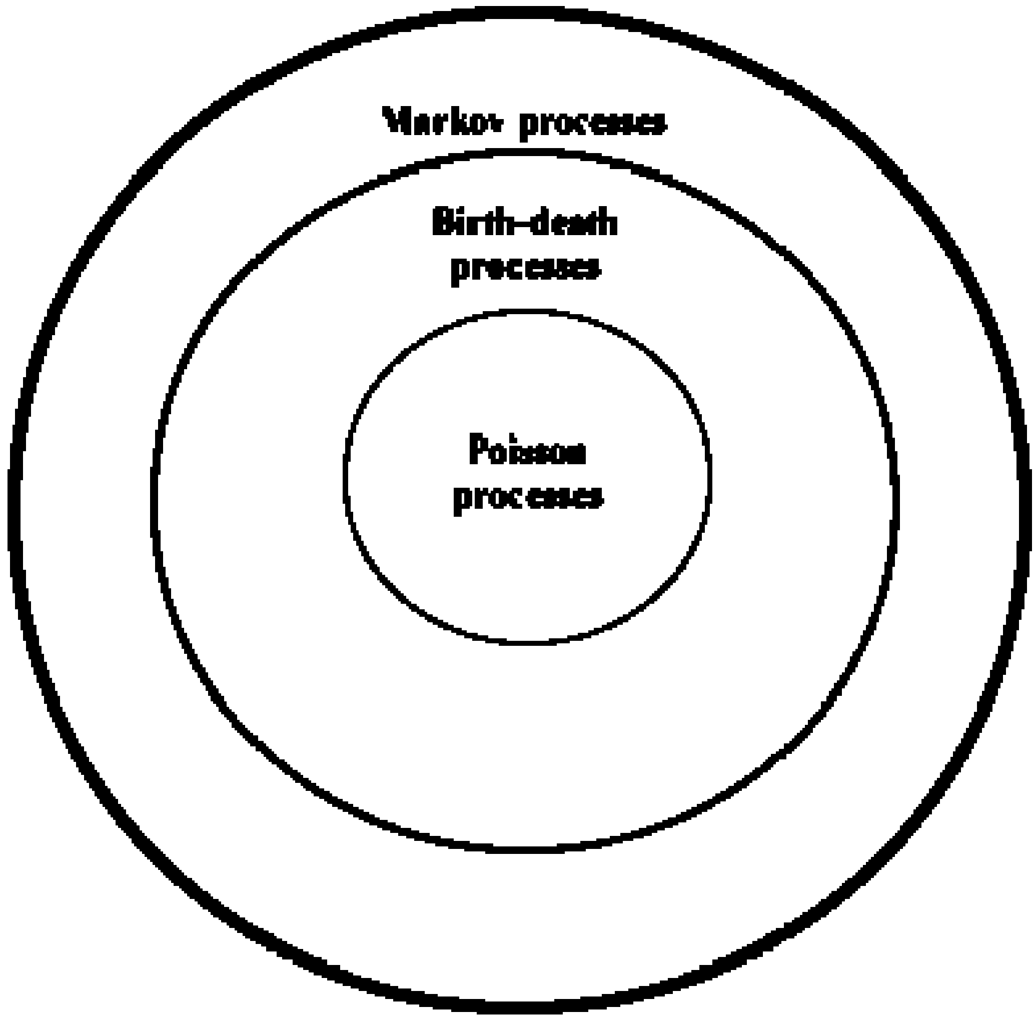


Poisson Process(cont)

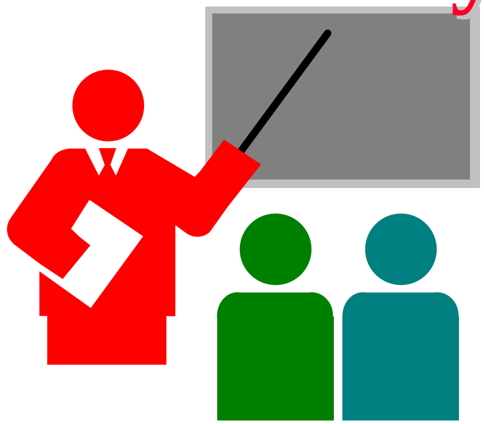
- 4. If the arrivals to a service facility with m service centers are Poisson with a mean rate λ, the departures also constitute a Poisson stream with the same rate λ, provided $\lambda < \sum_i \mu_i$. Here, the servers are assumed to have exponentially distributed service times.



Relationship Among Stochastic Processes



Summary



- Kendall Notation: A/S/m/B/k/SD, M/M/1
- Number in system, queue, waiting, service
  Service rate, arrival rate, response time, waiting time, service time
- Little's Law: Mean number in system = Arrival rate X Mean time spent in the system
- Processes: Markov $\Rightarrow$ Memoryless,
  Birth-death $\Rightarrow$ Adjacent states
  Poisson $\Rightarrow$ IID and exponential inter-arrival

Homework

- Submit answer to Exercise 30.4

30.4 During a one-hour observation interval, the name server of a distributed system received *10,800* requests. The mean response time of these requests was observed to be one-third of a second. What is the mean number of queries in the server? What assumptions have you made about the system? Would the mean number of queries be different if the service time was not exponentially distributed?